УТВЕРЖДЕН

КПДА.10964-01 32-ЛУ

ЗАЩИЩЕННАЯ ОПЕРАЦИОННАЯ СИСТЕМА РЕАЛЬНОГО ВРЕМЕНИ «НЕЙТРИНО»

Руководство системного программиста

(администратора)

КПДА.10964-01 32

Листов 443

2016

СОДЕРЖАНИЕ

Предисловие

Содержание руководства

С помощью данного руководства вы узнаете:

- как использовать среду исполнения ЗОСРВ «Нейтрино» вне зависимости от типа компьютера, на котором она запущена (встраиваемая система или настольный ПК). Данное руководство служит дополнением к руководству «Описание программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01 и дает ответы на вопрос: "Как это сделать?". Из этого руководства вы узнаете, как можно работать с ЗОСРВ «Нейтрино» с помощью командной строки или через оболочку Photon;
- как выполнять такие операции по системному администрированию как настройка учетных записей, управление безопасностью, запуск компьютера с ЗОСРВ «Нейтрино» и др.

Данное руководство предназначено для программистов, которые разрабатывают приложения для среды исполнения ЗОСРВ «Нейтрино», а также для производителей оборудования и для других пользователей ОС, которым необходимо предоставить своим конечным пользователям документацию на ЗОСРВ «Нейтрино», применяемую в их продукте.

Примечание.

- предполагается, что ЗОСРВ «Нейтрино» уже установлена и работает на вашем компьютере.
- в состав вашей системы необязательно могут входить все элементы, описанные в данном руководстве. Это зависит от установленного у вас программного обеспечения. Например, некоторые утилиты входят в состав комплекта разработчика, другие же могут быть включены в соответствующий пакет поддержки процессорных плат.
- отключите функцию PnP-aware OS в BIOS вашего компьютера.

В таблице П1 содержится краткое описание разделов руководства.

Таблица П1

Тема	Раздел
Сравнение ЗОСРВ «Нейтрино» с другими операционными системами	1. Знакомство с ЗОСРВ «Нейтрино»
Начало и завершение сеанса, завершение работы системы ЗОСРВ «Нейтрино»	2. Вход в систему, выход из нее и завершение работы системы
Добавление пользователей в систему, управление паролями и т. д.	3. Управление учетными записями пользователей
Основы работы с клавиатурой, командной строкой и оболочкой (интерпретатор команд)	4. Командная строка
Использование в ЗОСРВ «Нейтрино» графического интерфейса пользователя	5. Графическая оболочка Photon microGUI
Файлы, каталоги и разрешения	6. Работа с файлами
Как редактировать файлы	7. Редакторы
Управление процессом загрузки вашего компьютера	8. Управление запуском ЗОСРВ «Нейтрино»
Настройка параметров вашей оболочки, установка времени и т. д.	9. Настройка среды
Создание собственных команд	10. Написание сценариев командного интерпретатора
Файловые системы, которые поддерживаются в ЗОСРВ «Нейтрино»	11. Работа с файловыми системами
Получение доступа к другим компьютерам, используя собственные сетевые средства ЗОСРВ «Нейтрино»	12. Прозрачная распределенная обработка с помощью протокола Qnet
Работа с сетью TCP/IP	13. Сеть TCP/IP
Добавление принтеров в систему и их использование	14. Печать
Добавление в систему USB-устройств, терминалов, видеокарт и другого оборудования	15. Подключение оборудования
Добавление встраиваемых HTTP-сервисов и динамического контента к встраиваемым Web-приложениям	16. Настройка встраиваемого Web-сервера
Отслеживание изменений в вашем программном обеспечении и в других файлах	17. Использование CVS
Резервное копирование и восстановление файлов	18. Резервное копирование и восстановление данных

Таблица П1

Тема	Раздел
Как сделать систему ЗОСРВ «Нейтрино» более безопасной	19. Обеспечение безопасности системы
Анализ и повышение производительности вашего компьютера	20. Точная настройка системы
Сколько процессов, файлов и т. д. может поддерживаться в вашей системе	21. Лимиты системы
Как получить помощь	22. Техническая поддержка

В руководстве используются следующие типы шрифтов для различных элементов текста (табл. П2).

Таблица П2

Элемент текста	Пример
Фрагмент кода	if (stream == NULL)
Параметр команды	-lR
Команда	make
Переменная окружения	**PATH**
Файлы и каталоги	/dev/null
Имя функции	exit()
Клавиша	<Enter>
Комбинация клавиш	<Ctrl>+<Alt>+<Delete>
Ввод с клавиатуры	something you type
Программный вывод	login:
Программная константа	NULL
Программный тип данных	unsigned short
Программные литералы	0xFF, "message string"
Заменяемые имена	stdin
Элементы интерфейса	**Cancel**

Для обозначения последовательного выбора пунктов меню используется стрелка (→), например: Perspective→Show View.

Замечание для пользователей Windows

В документации, посвященной программным продуктам ЗОСРВ «Нейтрино», символ прямого слэша (/) используется в качестве разделителя во всех именах путей, включая те из них, которые относятся к Windows-файлам.

Кроме того, следует иметь в виду, что в большинстве случаев применяются правила, принятые в файловых системах POSIX/UNIX.

Техническая поддержка

Информация о технической поддержке доступна на сайте www.kpda.ru.

1. Знакомство с ЗОСРВ «Нейтрино»

1.1. Отличия ЗОСРВ «Нейтрино» от других операционных систем

В данном разделе описаны отличия ЗОСРВ «Нейтрино» от ОС UNIX и Microsoft Windows с точки зрения пользователя, а не разработчика. Более подробные сведения об архитектуре ЗОСРВ «Нейтрино» и философии, лежащей в ее основе, представлены в документе «Описание применения. Часть 1. Системная архитектура» КПДА.10964-01 31 01.

UNIX

Если вы знакомы с UNIX-подобными операционными системами, вам будет очень легко разобраться в ЗОСРВ «Нейтрино». В основе рассматриваемой операционной системы лежит микроядро, которое работает в окружении группы процессов (включая, например, командный интерпретатор Korn Shell ksh, см. раздел 4). Каждый процесс имеет свой идентификатор (process ID, PID) и содержит в себе один или несколько потоков.

Примечание. Определить версию релиза ядра вашей системы вы можете с помощью команды uname -a. Подробнее см. в «Описание программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

ЗОСРВ «Нейтрино» – это многопользовательская ОС, т. е. она одновременно поддерживает любое количество пользователей. Пользователи организуются в группы, каждая из которых имеет свой набор разрешений на доступ к файлам и каталогам (более подробные сведения см. в разделе 3).

ЗОСРВ «Нейтрино» соответствует различным промышленным стандартам, в том числе POSIX (командный интерпретатор и утилиты) и TCP/IP, что облегчает задачу переноса существующего кода и скриптов в ЗОСРВ «Нейтрино».

Командная строка в ЗОСРВ «Нейтрино» выглядит так же, как и в UNIX. Поддерживаются многие привычные утилиты (grep, find, ls, awk), которые можно объединять при помощи неименованных программных каналов (pipe). Также

возможно перенаправлять потоки ввода и вывода, проверять коды возврата и т. д. Многие утилиты повторяются в UNIX и ЗОСРВ «Нейтрино», однако некоторые утилиты в ЗОСРВ «Нейтрино» имеют другое имя или синтаксис (табл. 1.1).

Таблица 1.1

UNIX	**ЗОСРВ «Нейтрино»**	**См. также**
adduser	passwd	3. Управление учетными записями пользователей
at	cron	«Описание программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01
dmesg	slogger, sloginfo	«Описание программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01
fsck	chkfsys, chkdosfs	18. Резервное копирование и восстановление данных
ifconfig eth0	ifconfig en0	13. Сеть TCP/IP
lp	lpr	14. Печать
lpc	lprc	14. Печать
lpq, lpstat	lprq	14. Печать
lprm, cancel	lprrm	14. Печать
man	use	4. Командная строка
pg	less, more	4. Командная строка

Подробное описание каждой команды можно найти в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

Microsoft Windows

ЗОСРВ «Нейтрино» и ОС Windows имеют разные архитектуры, однако, с точки зрения пользователя, основное различие между ними заключается в способе запуска программ. Многое из того, что пользователь Windows делает посредством графического пользовательского интерфейса (Graphical User Interface, GUI), пользователь ЗОСРВ «Нейтрино» выполняет с помощью утилит командной строки, конфигурационных файлов и скриптов. Тем не менее, в ЗОСРВ «Нейтрино» реализована мощная интегрированная среда разработки (Integrated Development Environment, IDE), которая позволяет создавать, тестировать и отлаживать программы и встраиваемые системы.

Приведем еще несколько основных отличий:

- в ЗОСРВ «Нейтрино» и DOS применяются разные символы конца строки: в ЗОСРВ «Нейтрино» используется символ перехода на новую строку, а в DOS — символ возврата каретки и символ перевода строки. Если необходимо перенести текстовый файл из одной ОС в другую, вы можете воспользоваться утилитой textto для конвертирования файла. Например, для того чтобы конвертировать символы конца строки в формат ЗОСРВ «Нейтрино», введите следующую команду:

textto -l my_file

Для того чтобы конвертировать символы конца строки в формат DOS, введите команду:

textto -l my_file

- в ЗОСРВ «Нейтрино» используется прямой слэш (/) вместо обратного слэша (\) в качестве разделителя в именах путей.
- в ЗОСРВ «Нейтрино» нельзя использовать DOS-команды, однако вместо них существует множество эквивалентов. Более подробные сведения см. в подразд. "ЗОСРВ «Нейтрино» для пользователей MS-DOS" раздела 4.

Ограничения

Хотя ЗОСРВ «Нейтрино» вполне может служить в качестве настольной ОС, компания ООО «СВД Встраиваемые Системы» не поставляет с ней настольные

приложения, вроде текстовых редакторов, электронных таблиц или почтовых клиентов. Некоторые приложения такого рода предлагаются третьими сторонами.

1.2. В чем уникальность ЗОСРВ «Нейтрино»?

ЗОСРВ «Нейтрино» состоит из микроядра и различных процессов. Каждый процесс, в том числе драйверы устройств, выполняется в своем собственном виртуальном пространстве памяти (рис. 1.1).

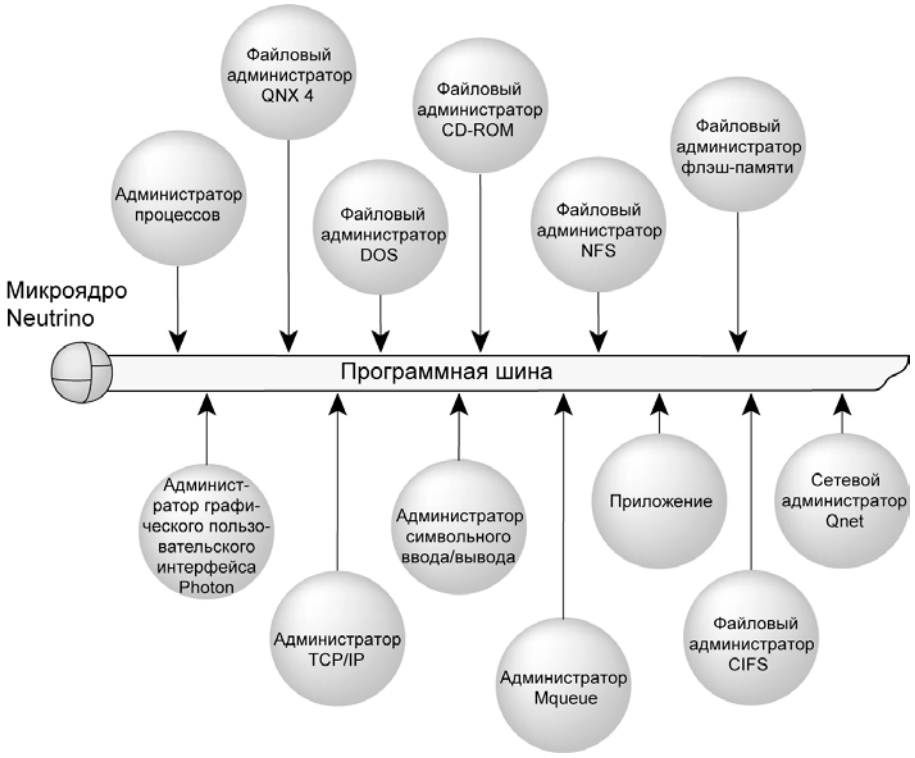


Рис. 1.1. Архитектура ЗОСРВ «Нейтрино»

Преимущество использования виртуальной памяти в том, что один процесс не может повредить адресное пространство другого процесса. Более подробные сведения можно найти в главе 1 документа «Описание применения. Часть 1. Системная архитектура» КПДА.10964-01 31 01.

Главной особенностью ЗОСРВ «Нейтрино» является ее микроядерная архитектура, вокруг которой строится инфраструктура администраторов ресурсов (см. далее). Драйверы имеют точно такой же статус, как и другие пользовательские приложения, поэтому их отладку можно выполнять с помощью тех же

высокоуровневых инструментов IDE, которые используют исходный текст (source-aware) и точки останова (breakpointing) и которые обычно применяются для отладки пользовательских приложений.

Это также означает, что:

- процесс отладки драйвера не затрагивает ядро;
- сбой драйвера не приведет к сбою всей ОС;
- останавливать и перезапускать драйвер, как правило, можно без перезапуска системы.

Чтобы убрать обработчики прерываний (которые, как правило, представляют собой наиболее сложные участки кода), разработчики обычно могут переместить код, предназначенный для взаимодействия с оборудованием, на уровень потока приложения. Это большие преимущества по отладке, которые снимают все возможные ограничения. Из этого видно, что ЗОСРВ «Нейтрино» имеет огромные преимущества относительно монолитных систем.

Более того, благодаря модульности ЗОСРВ «Нейтрино», в реализованную систему можно включать избыточные компоненты в виде простого, но весьма эффективного администратора высокой готовности (High Availability (HA) manager), который, в отличие от более монолитных архитектур, значительно упрощает создание систем высокой степени отказоустойчивости. Конечно, простота, с которой рабочие устройства могут быть отображены в пространство путевых имен POSIX, также является привлекательным преимуществом ЗОСРВ «Нейтрино».

С точки зрения разработчиков, системных администраторов, пользователей, ценные качества ЗОСРВ «Нейтрино» состоят и в том, что она основана на стандарте POSIX, обеспечивает высокую скорость отклика в реальном времени (благодаря возможности ограничения максимального объема нереентерабельного кода) и имеет отказоустойчивое микроядро.

Предупреждение.

Некоторые системы x86 могут функционировать в режиме SMM (System Management Mode, режим управления системой), при котором BIOS устанавливает особый код, выполняющийся при возникновении SMI (System Management Interrupt, прерывание управления системой). Прерывания SMI могут быть сгенерированы материнской платой или периферийным оборудованием и не может маскироваться операционной системой. При входе в режим SMM обычные действия — включая ОС — приостанавливаются, и обработчик SMI выполняется с высоким приоритетом. Избегайте использования систем, в которых SMM не может быть отключен, поскольку этот режим может негативно отразится на характеристиках реального времени ЗОСРВ «Нейтрино». Операционная система не может ни повлиять на задержки, вносимые SMM, ни даже определить факт перехода системы в SMM.

Микроядерная архитектура ЗОСРВ «Нейтрино» позволяет масштабировать программный код в соответствии с требованиями очень компактных встраиваемых систем с минимальными ресурсами, и в то же время ЗОСРВ «Нейтрино» является достаточно мощной, чтобы работать в качестве настольной ОС. ЗОСРВ «Нейтрино» поддерживает различные аппаратные платформы, в том числе x86, ARM, MIPS и PPC, а также позволяет реализовывать симметричную многопроцессорность (symmetric multiprocessing, SMP) и многопроцессорность с привязкой (bound multiprocessing, BMP) на многоядерных системах с количеством процессоров до 32.

Кроме того, в ЗОСРВ «Нейтрино» применяется сетевой протокол Qnet, который обеспечивает прозрачную распределенную обработку по сети, т. е. возможность доступа со своей машины к файлам и процессам на любой другой машине, находящейся в сети.

Администраторы ресурсов

Администратор ресурсов (resource manager) — это серверная программа, которая принимает сообщения от других программ и может взаимодействовать с оборудованием. Все драйверы устройств и файловые системы в ЗОСРВ «Нейтрино» реализуются как администраторы ресурсов.

Администраторы ресурсов служат для обеспечения интерфейса с различными типами устройств. В их число могут входить как физические устройства (например, последовательные порты, параллельные порты, сетевые карты, диски), так и виртуальные (например, /dev/null, сетевая файловая система, псевдотерминалы).

Взаимодействие между администратором ресурса и клиентской программой, использующей данный ресурс, осуществляется посредством гибкого механизма пространства путевых имен (pathname-space mapping). Этот механизм служит для ассоциирования путевых имен с администратором ресурсов. Для этого администратор ресурсов уведомляет администратор процессов (process manager) о том, что он будет отвечать за обработку запросов к заданной точке монтирования (или ниже нее в случае файловых систем). Таким образом, администратор процессов устанавливает связь между службами (т. е. функциями, предоставляемыми администраторами ресурсов) и именами путей.

Как только администратор ресурсов установил свой префикс путевого имени, он начинает получать сообщения при каждой попытке клиентской программы сделать по этому путевому имени вызов open(), read(), write() и т. д.

Более подробные сведения см. в главе 8 руководства «Описание применения. Часть 1. Системная архитектура» КПДА.10964-01 31 01.

2. Вход в систему, выход из нее и завершение работы системы

ЗОСРВ «Нейтрино» – это многопользовательская операционная система. Несколько пользователей могут одновременно входить и работать в системе, при этом защита пользователей осуществляется с помощью механизма владения ресурсами (resource ownership) и прав доступа (permissions).

В зависимости от конфигурации система загружается либо в графическом режиме (графическая оболочка Photon), либо в текстовом режиме, после чего у пользователя запрашивается имя и пароль. Более подробную информацию вы получите в разделе 8.

Примечание. Систему можно сконфигурировать таким образом, чтобы пользователю не требовалось выполнять процедуру входа.

2.1. root или не root?

При первой установке ЗОСРВ «Нейтрино» автоматически создается единственная учетная запись пользователя с именем root. Этот пользователь может выполнять в системе любые действия (по терминологии Windows, он имеет привилегии администратора, а в UNIX-подобных операционных системах учетную запись root называют суперпользователем (superuser).

По умолчанию для учетной записи root пароль не установлен. Для защиты системы следует:

- создать надежный пароль сразу после установки ОС;
- создать другую учетную запись (см. раздел 3) для повседневной работы (это поможет предотвратить случайное изменение или удаление системного программного обеспечения, поскольку некоторые действия, например запуск драйверов, выполнение задач по системному администрированию или профилированию приложений, можно выполнить только под учетной записью root).

Приглашение командной строки, заданное по умолчанию, указывает используемый идентификатор пользователя:

- для пользователя root отображается символ диеза (#);
- для остальных пользователей отображается символ доллара ($).

Более подробные сведения об изменении приглашения см. “.kshrc” в приложении.

2.2. Вход в систему

Графический режим

Если система настроена на запуск графической оболочки Photon, автоматически запускается утилита phlogin2 или phlogin, отображающая диалоговое окно для входа в систему. В этом окне нужно ввести имя пользователя (или щелкнуть мышью по пиктограмме нужного пользователя), ввести пароль и затем нажать кнопку Login.

Текстовый режим

Если система настроена на загрузку в текстовом режиме, автоматически запускается утилита login, которая запрашивает имя пользователя и пароль.

Примечание. Если вы введете неверное имя пользователя, система не сообщит об этом и все равно запросит пароль (это сделано для повышения безопасности системы).

Текстовый режим на компьютере с архитектурой x86 может быть реализован на физической консоли посредством утилиты devc-con или devc-con-hid. Кроме того, соединение с целевой системой может осуществляться через последовательный порт или посредством протокола TCP/IP.

После входа в систему

После входа в систему автоматически запускается сценарий /home/имя_пользователя/.profile. Он позволяет настроить рабочее окружение пользователя (working environment), не влияя на настройки других пользователей. Более подробные сведения см. в разделе 9.

Изменить пароль можно с помощью команды passwd. Эта команда запрашивает текущий и новый пароль (см. подразд. "Управление собственной учетной записью" раздела 3).

Чтобы войти в систему в качестве другого пользователя, введите команду login в командной строке и затем введите имя пользователя и пароль.

Примечание. Утилита su позволяет временно переключаться на другого пользователя. Эта утилита не запускает профиль заданного пользователя и не вносит значительных изменений в окружение (При вызове утилиты su профиль нового пользователя можно запустить, указав перед именем этого пользователя символы дефиса и пробела). Более подробные сведения см. в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

Для определения своего текущего имени пользователя может использоваться команда id.

2.3. Выход из системы

Выход из графического режима

Чтобы выйти из графической оболочки Photon, выполните следующие действия:

- выберите команду Log Out в меню Launch или Desktop либо введите команду phshutdown в командной строке. На экране появится диалоговое окно завершения работы;
- выберите Logout (End Photon session) и нажмите кнопку Ok. Если система настроена на запуск в графической оболочке Photon, на экране снова появится диалоговое окно утилиты phlogin2 или phlogin. Если же вы запустили графическую оболочку Photon вручную из текстового режима, система вернется в этот режим.

Даже если запуск графической оболочки Photon происходит автоматически, вы можете завершить сеанс работы с Photon и перейти в текстовый режим. Для этого выполните следующие действия:

- в диалоговом окне входа в систему щелкните мышью Shutdown. На экране появится диалоговое окно завершения работы.
- выберите Exit to text mode и нажмите кнопку Ok.

Если вы запустите сеанс работы с терминалом из графической оболочки Photon (например, щелкнув мышью **Terminal** на системной панели, утилита pterm запустит командный интерпретатор от имени текущего пользователя Photon. Как и в текстовом режиме, вы можете переключаться между пользователями, однако при выполнении команды выхода окно утилиты pterm закрывается.

Выход из текстового режима

Чтобы выйти из системы, находящейся в текстовом режиме, введите в командной строке команду logout. Вы также можете выйти из системы с помощью завершения командного интерпретатора. Для этого нужно ввести команду exit или нажать комбинацию клавиш <Ctrl>+<D>.

2.4. Завершение работы и перезагрузка системы

ЗОСРВ «Нейтрино» редко требуется перезагружать целиком. В случае сбоя драйвера или другого системного процесса, как правило, его можно перезапустить отдельно.

Примечание. Для завершения работы системы с ЗОСРВ «Нейтрино» нельзя просто выключать компьютер из электросети, т. к. процессы могут завершиться некорректно, а данные, находящиеся в кэше файловой системы, могут не записаться на диск. Информация о том, как снизить подобный эффект, приведена в подразд. "Файловые системы" раздела 20.

Для завершения работы или перезагрузки системы в текстовом режиме используйте команду shutdown. Однако эта команда доступна только для привилегированного пользователя root. Эта утилита позволяет выполнять следующие операции:

- задавать узел, работу которого требуется завершить (по умолчанию текущий узел);

- задавать тип завершения работы (по умолчанию выполняется перезагрузка);
- ускоренно завершать работу;
- выводить список действий, выполняемых при завершении работы (т. е. подробный отчет).

В графической оболочке Photon вы можете выполнить команду phshutdown в командной строке или выбрать пункт Shutdown в меню Launch или Desktop. По умолчанию эти действия могут выполнять любые пользователи (не только root).

Перед тем как завершить работу системы, команды shutdown и phshutdown посылают сигнал SIGTERM всем выполняемым процессам для их корректного завершения. Более подробные сведения об этих утилитах см. в документе "Описание программы" КПДА.10964-01 13.

Внимание! Использование средств защиты информации от несанкционированного доступа, разработанных в соответствии с требованиями российских руководящих документов описано в отдельном документе «Описание применения. Часть 2. Комплекс средств защиты» КПДА.10964-01 31 02.

3. Управление учетными записями пользователей

В данном разделе мы рассмотрим, как работают учетные записи пользователей, как с помощью утилиты passwd пользователи могут изменять пароль своей учетной записи, а системные администраторы – создавать и обслуживать учетные записи пользователей посредством редактирования файлов базы данных учетных записей.

Примечание. Во встраиваемой системе разработчик может удалить файлы, относящиеся к учетным записям, отключить вход в систему и обращение к пользователям и группам по именам. При этом система остается полностью многопользовательской и поддерживает выполнение программ и владение системными ресурсами множеством пользователей с различными идентификаторами. Если ваша система настроена именно таким образом, большая часть материала в этом разделе может вам не понадобиться.

3.1. Для чего служат учетные записи пользователей?

Пользовательская учетная запись связывает текстовое имя пользователя с числовым идентификатором пользователя, идентификатором группы, паролем для входа в систему, полным именем, домашним каталогом и начальным командным интерпретатором (login shell). Эти данные хранятся в файлах /etc/passwd и /etc/shadow, которые используются утилитами входа в систему и другими приложениями, нуждающимися в информации учетных записей.

Примечание. Имена пользователей и пароли должны создаваться с учетом регистра.

Учетные записи позволяют выполнять следующие действия:

- для пользователей – входить в систему под своим именем и паролем, запуская сеанс работы с соответствующим идентификатором пользователя и группы;
- создавать собственные окружения;

- для приложений – определять имя пользователя и учетные данные по идентификатору пользователя и группы, если они указаны в файлах /etc/passwd и /etc/group (например, команда ls -l отображает имена, а не идентификаторы пользователей и групп, владеющих файлами);
- для утилит и приложений – принимать в качестве входных данных имена пользователей вместо числовых идентификаторов;
- для командных интерпретаторов – преобразовывать пути ~имя_пользователя в фактические путевые имена на основе информации о домашних каталогах пользователей, хранимой в учетных записях.

Группы (groups) используются для связывания аналогичных прав доступа с группами пользователей. Список членов группы определяется записями в файлах /etc/passwd и /etc/group, а права доступа для членов группы – идентификатором группы выполняющейся программы, идентификатором группы и правами доступа к файлам и каталогам.

При входе в систему пользователь считается принадлежащим к группе, указанной в файле /etc/passwd. Переключиться в другую группу можно с помощью утилиты newgrp.

Учетные записи и идентификаторы пользователей: вход в систему, считывание и права доступа

После входа в систему возможности доступа ваших программ к ресурсам системы и выполнения операций (например, передачи сигналов другим процессам) определяются числовым идентификатором пользователя, от имени которого выполняются программы. Текстовые имена используются только утилитами и приложениями, которым требуется выполнять преобразование между именами и числовыми идентификаторами.

Примечание. Изменение имени пользователя, группы, идентификатора и прочих параметров в базе данных учетных записей не влияет на доступ к файлам до тех пор, пока пользователь не войдет в систему заново.

Пользователь root (с идентификатором 0) имеет право выполнять почти любые действия с файлами независимо от их владельцев и установленных разрешений. Более подробные сведения см. в подразд. "Владение файлами и права доступа" раздела 6.

Примечание. Когда командный интерпретатор обрабатывает путевое имя ~имя_пользователя, он считывает домашний каталог пользователя из файла /etc/passwd. Если вы удалите или измените учетную запись пользователя, то командный интерпретатор, ранее использовавший путевое имя ~имя_пользователя для доступа к домашнему каталогу, может попытаться воспользоваться устаревшей информацией, поскольку командный интерпретатор кэширует такого рода данные.

Новые командные интерпретаторы заново считывают данные из файла /etc/passwd. Это может вызывать проблемы в тех случаях, когда командный сценарий, использующий ~имя_пользователя, запускает другой сценарий, который использует эту же возможность для определения полного пути. Эти сценарии будут оперировать разными путями, если информация о домашнем каталоге изменилась с момента первого считывания.

Что происходит при входе в систему?

Как правило, сеанс работы с компьютером начинается с процедуры входа в систему (см. раздел. 2). Действия системы после процедуры входа зависят от конфигурации учетной записи.

При входе пользователя система создает для него сеанс, лидером которого является процесс с идентификатором пользователя и идентификатором группы по умолчанию, заданными в файле /etc/passwd.

Идентификаторы пользователя и группы определяют разрешения, которые имеет этот процесс для доступа к файлам и системным ресурсам. Кроме того, если процесс создает какие-либо файлы или каталоги, то они принадлежат данному пользователю и группе. Каждый новый процесс, запускаемый пользователем, наследует идентификатор пользователя и идентификатор группы от родительского

процесса. Более подробные сведения см. в подразд. "Владение файлами и права доступа" раздела 6.

Примечание. Более подробные сведения о сессиях и группах процессов можно найти в IEEE Std 1003.1-2001 Standard for Information Technology Portable Operating System Interface.

Вход в систему в текстовом режиме (утилита login) происходит иначе, чем в графическом режиме (утилита phlogin2 или phlogin):

- при входе в систему с помощью утилиты login происходит переход в каталог, указанный в переменной окружения HOME. Эта утилита также записывает в переменную LOGNAME имя пользователя, а в переменную SHELL – начальную оболочку, указанную в учетной записи. Затем запускается начальная оболочка, в качестве которой, как правило, служит командный интерпретатор (/bin/sh), однако в его роли может также выступать любое приложение, запускаемое при входе пользователя в систему;
- при входе в систему с помощью команды phlogin2 или phlogin в графической оболочке Photon также происходит переход в каталог, указанный в переменной окружения HOME, а значения LOGNAME и SHELL задаются в соответствии с именем пользователя и его начальной оболочкой.

Однако при входе в графическом режиме начальная оболочка запускается не в виде интерактивной программы, а с аргументами -c /usr/bin/ph.

Предупреждение. Если в качестве начальной оболочки используется не /bin/sh или /bin/ksh, то, вероятно, вы не сможете войти в систему с помощью утилит phlogin2 и phlogin.

Команда ph запускает среду графической оболочки Photon. Из этой среды вы можете запустить в окне утилиты pterm командный интерпретатор, определенный значением переменной окружения SHELL.

3.2. База данных учетных записей

База данных учетных записей состоит из файлов, перечисленных с соответствующими разрешениями доступа в табл. 3.1.

Таблица 3.1

Файл	Владелец	Группа	Разрешения
/etc/passwd	root	root	rw- r-- r--
/etc/group	root	root	rw- r-- r--
/etc/shadow	root	root	rw- --- ---
/etc/.pwlock	root	root	rw- r-- r--

Обратите внимание на неограниченный доступ на чтение файла /etc/passwd, что позволяет стандартным утилитам получать информацию о пользователях. Однако шифрованные пароли хранятся не в этом файле, а в /etc/shadow, доступ к которому имеет только пользователь root. Таким образом, пароли защищаются от дешифрации.

Примечание. Для обеспечения безопасности не изменяйте вышеописанные разрешения.

Файл /etc/passwd

Все строки в файле /etc/passwd имеют следующий формат:

```
username:has_pw:userid:group:comment:homedir:shell
```

Компоненты строки разделяются двоеточиями. Далее приводятся их описания:

- username — имя пользователя, которое может содержать любые символы за исключением двоеточия (:). Кроме того, следует избегать специальных символов командного интерпретатора. Более подробные сведения см. в подразд. "Применение кавычек со специальными символами" раздела 4.
- has_pw — это поле должно либо быть пустым, либо содержать значение x. Первый случай означает, что у пользователя нет пароля, а второй — что шифрованный пароль хранится в файле /etc/shadow.
- userid — числовой идентификатор пользователя.

- group — числовой идентификатор группы.
- comment — поле для свободного комментария, в котором, как правило, сообщают реальное имя пользователя. В комментарии запрещено использовать двоеточие.
- homedir — домашний каталог пользователя.
- shell — первая команда, которая должна быть выполнена после утилиты login (по умолчанию /bin/sh).

Примечание. Программе login нельзя передавать аргументы.

Приведем пример записи в файле /etc/passwd:

```
fred:x:290:120:Fred L. Jones:/home/fred:/bin/sh
```

Файл /etc/group

Все строки в файле /etc/group имеют следующий формат:

```
groupname:x:group_ID:[username[,username]...]
```

Компоненты строки разделяются двоеточиями. Далее приводятся их описания:

- groupname — имя группы. Как и имя пользователя, оно может содержать любые символы за исключением двоеточия (:), однако следует избегать специальных символов командного интерпретатора. Более подробные сведения см. в подразд. "Применение кавычек со специальными символами" раздела 4.
- x — пароль группы. Операционная система ЗОСРВ «Нейтрино» не поддерживает пароли для групп.
- group_ID — числовой идентификатор группы.
- username[,username]... — имена принадлежащих к данной группе пользователей, взятые из учетных записей. Разделяются запятыми (,).

Пример записи:

```
techies:x:123:michel,ali,sue,jake
```

Файл /etc/shadow

Все строки в файле /etc/shadow имеют следующий формат:

```
username:password:0:0
```

Компоненты строки разделяются двоеточиями. Далее приводятся их описания.

username - имя пользователя для входа в систему.

password - шифрованный пароль пользователя.

Файл /etc/.pwlock

Утилита passwd создает файл /etc/.pwlock для того, чтобы сообщить другим запущенным ее экземплярам, что файл паролей в данный момент изменяется. По завершении своей работы утилита passwd удаляет файл блокировки.

Если системному администратору необходимо отредактировать файлы учетной записи, ему следует выполнить следующие действия:

- заблокировать базу данных паролей: создать файл /etc/.pwlock (если он уже существует, то дождаться его удаления).
- открыть нужные файлы с помощью какого-либо текстового редактора и внести необходимые изменения.
- разблокировать базу данных паролей посредством удаления файла /etc/.pwlock.

3.3. Управление собственной учетной записью

Обычные пользователи (т. е. не root) могут менять свой пароль, а также настраивать окружение посредством изменения конфигурационных файлов, расположенных в домашнем каталоге (см. раздел 9).

Изменение пароля

Для изменения пароля служит утилита passwd, а в графической оболочке Photon также может использоваться утилита phuser. Обе утилиты запрашивают текущий пароль и затем предлагают ввести новый. Для проверки ошибок новый

пароль вводится дважды. С помощью утилиты phuser вы можете также "привязать" к вашему пользователю пиктограмму.

В зависимости от правил задания паролей, установленных системным администратором, утилита passwd может потребовать, чтобы вводимый пароль имел определенную длину или содержал определенные элементы (например, сочетание букв, цифр и знаков препинания). Если пароль не соответствует установленному критерию, утилита passwd запросит ввести другой пароль.

Если к системе имеют доступ другие пользователи (физически, через Интернет или коммутируемое соединение), пароль следует выбрать таким образом, чтобы он гарантировал защиту от несанкционированного доступа. Для этого пароль должен отвечать следующим требованиям:

- иметь длину не менее 5 символов;
- состоять из нескольких слов или чисел и включать знаки препинания и пробелы;
- не использоваться в других системах (многие системы, в особенности Web-сайты, хранят и передают пароли в нешифрованной форме; это позволяет людям, имеющим доступ к системе, видеть ваш пароль как обычный текст);
- включать буквы как в нижнем, так и в верхнем регистре;
- не содержать слов, фраз и чисел, которые могут быть угаданы другими людьми (например, имена членов вашей семьи, клички домашних животных, номера автомашин, даты рождения).

Более подробные сведения об обеспечении безопасности систем см. в разделе 19.

Если вы забыли пароль

Если вы забыли пароль, попросите системного администратора (пользователя root) назначить новый пароль для вашей учетной записи. root — единственный пользователь, который может сделать это.

Вообще никто не может получить ваш старый пароль из файла /etc/shadow. Если пароль короткий или состоит из одного слова, системный администратор (или

злоумышленник) может без труда подобрать его, поэтому лучше задать новый пароль.

Если вы системный администратор и забыли пароль пользователя root, вам следует найти альтернативный способ доступа к файлам /etc/passwd и /etc/shadow, чтобы переустановить пароль. Например, вы можете использовать такие способы:

- загрузите систему с другого диска или устройства, позволяющего вам войти как пользователь root (например, с установочного компакт-диска), и установите пароль вручную;
- откройте нужные файлы из учетной записи root на другой машине ЗОСРВ «Нейтрино» через сеть Qnet (более подробные сведения см. в разделе 12);
- удалите из системы носитель, содержащий файлы /etc/passwd и /etc/shadow, и подключите его к другой машине ЗОСРВ «Нейтрино», на которой вы можете отредактировать их;
- в случае встроенной системы постройте новый образ с новыми файлами /etc/passwd и /etc/shadow и затем перенесите его в целевую систему.

3.4. Управление другими учетными записями

Системный администратор должен добавлять и удалять учетные записи пользователей и их группы, управлять паролями и разрешать проблемы, связанные с работой пользователей. Для этого следует войти в систему как root, поскольку другие пользователи не имеют разрешение на изменение файлов /etc/passwd, /etc/shadow и /etc/group.

Предупреждение.

Изменять пароль существующего пользователя с помощью утилиты passwd можно в любое время. Однако любые другие изменения базы данных учетных записей во время использования системы могут быть небезопасны. Следующие операции по изменению учетных данных пользователя могут привести к некорректной работе приложений и утилит:

- добавление нового пользователя при помощи утилиты passwd или посредством ручного редактирования файла /etc/passwd;
- задание пароля для учетной записи, ранее не имевшей его;
- редактирование файлов /etc/passwd и /etc/group.

Если существует вероятность использования утилиты passwd или обновления файлов базы данных учетных записей во время их редактирования, следует предварительно заблокировать базу данных паролей, создав файл /etc/.pwlock.

Как описано ранее, для изменения пароля учетной записи следует пользоваться утилитой passwd. Однако для решения следующих задач вам понадобится текстовый редактор:

- изменение имени, полного имени, идентификатора пользователя, идентификатора группы, домашнего каталога и начального командного интерпретатора для существующего пользователя;
- создание новой учетной записи, не допустимой по конфигурации утилиты passwd;
- удаление учетной записи;
- создание и удаление группы;
- изменение списка членов группы.

В графической оболочке Photon вы можете воспользоваться командой phuser, которая обеспечивает графический интерфейс с утилитой passwd и позволяет выбирать пиктограмму и оболочку пользователя, а также редактировать группы.

Примечание. Изменения файлов учетной записи вручную не проверяются на соответствие правилам, установленным в файле конфигурации утилиты passwd. (Более подробные сведения можно найти в информации о файле /etc/default/passwd в описании команды passwd в «Описание программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.)

Добавление пользователей

Чтобы добавить пользователя:

- войдите в систему под учетной записью пользователя root;

- затем запустите утилиту passwd или phuser (при использовании графической оболочки Photon): passwd новое_имя_пользователя

Примечание. Длина имени пользователя не должна превышать 14 символов, в противном случае пользователь не сможет войти в систему.

Если вы задаете уже зарегистрированное ранее имя, утилита passwd считает, что вы хотите изменить его пароль. Если вы действительно хотите выполнить эту операцию, введите новый пароль и подтвердите его. В противном случае нажмите комбинацию клавиш <Ctrl>+<C>, чтобы завершить утилиту passwd без сохранения изменений.

Если имя пользователя еще не зарегистрировано, утилита passwd предложит ввести данные для учетной записи (список групп пользователя, домашний каталог и начальный командный интерпретатор). Конфигурационный файл /etc/default/passwd определяет правила, по которым устанавливаются данные по умолчанию для новых учетных записей. (Более подробные сведения см. в описании команды passwd.)

Утилита passwd запрашивает следующие данные.

- User id# (значение по умолчанию)

Введите числовой идентификатор нового пользователя. По умолчанию никакие два пользователя не могут иметь одинаковый идентификатор, иначе приложения не смогут однозначно определить имя пользователя, соответствующее этому идентификатору.

- Group id # (значение по умолчанию)

Введите числовой идентификатор группы, к которой будет принадлежать пользователь после входа в систему.

Примечание. Утилита passwd не добавляет нового пользователя в запись группы в файле /etc/group. Это можно сделать вручную с помощью текстового редактора. (Более подробные сведения см. в подразд. "Определение групп" далее в этом разделе.)

- Real name ()

Введите реальное имя пользователя. Реальное имя редко используется системными утилитами, однако оно может потребоваться для некоторых приложений (например, электронной почты).

- Home directory (/home/имя_пользователя)

Введите путевое имя домашнего каталога пользователя. Как правило, оно имеет вид /home/имя_пользователя. Утилита passwd автоматически создает заданный каталог. Если он уже существует, то по умолчанию предлагается выбрать другое путевое имя. Для отключения этой возможности см. сведения о файле /etc/default/passwd в описании команды passwd.

- Login shell (/bin/sh)

Программа, запускаемая при входе пользователя в систему. Как правило, это командный интерпретатор (/bin/sh), обеспечивающий интерактивную командную строку.

Примечание. Вместо начальной оболочки можно задать любую программу, однако ей нельзя передавать аргументы командной строки. Кроме того, вход в систему с использованием графических утилит phlogin2 и phlogin невозможен, если начальная программа несовместима со стандартом POSIX.

Задать начальную программу можно не только в учетной записи, но и в файле .profile, находящемся в домашнем каталоге каждого пользователя. Командный интерпретатор /bin/sh запускает этот профиль автоматически в начале работы. Более подробные сведения см. в разделе 9.

- New password:

Укажите пароль учетной записи и подтвердите его повторным набором.

Удаление учетных записей

Чтобы удалить учетную запись пользователя, выполните следующие действия.

- заблокируйте базу данных учетных записей. Если файл /etc/.pwlock не существует, создайте его, в противном случае дождитесь его удаления;

- удалите информацию об учетной записи из файлов /etc/passwd и /etc/shadow, чтобы запретить вход пользователя в систему, либо укажите в качестве

начального командного интерпретатора программу, которая выдает специальное сообщение и завершается;

- удалите ссылки на пользователя из файла /etc/group;
- разблокируйте базу данных учетных записей, удалив файл /etc/.pwlock;
- при необходимости удалите или измените владельцев системных ресурсов, которые имел пользователь;
- при необходимости удалите или измените ссылки на пользователя в системах электронной почты, файлах контроля доступа TCP/IP, приложения и т. д.

Вместо удаления пользователя вы можете отключить его учетную запись, изменив ее пароль с помощью утилиты passwd. В этом случае остается возможность определить, какими системными ресурсами владел пользователь, т. к. преобразование идентификатора пользователя в имя пользователя продолжает работать. При выполнении этого действия утилита passwd автоматически блокирует и разблокирует базу данных учетных записей.

Если когда-нибудь возникнет необходимость воспользоваться этой учетной записью, вы сможете либо переключиться на этого пользователя с пользователя root с помощью утилиты su ("switch user"), либо войти под этой учетной записью в систему. Если вы забудете пароль этой учетной записи, то всегда можете изменить его через пользователя root.

Что следует сделать с ресурсами, которыми владел бывший пользователь? Далее перечислено несколько возможностей:

- если вы сохранили учетную запись пользователя в базе данных, но отключили ее посредством изменения пароля или начального командного интерпретатора, то можете оставить файлы как есть.
- вы можете назначить файлы другому пользователю:

```
find / -user имя_пользователя_или_ID -chown новое_имя_пользователя
```

- вы можете выполнить архивацию файлов и при необходимости перенести их на другой накопитель:

```
find / -user имя_пользователя_или_ID | pax -wf архивный_файл
```

- вы можете удалить файлы:

```
find / -user имя_пользователя_или_ID -remove!
```

Предупреждение. Если вы удалите учетную запись пользователя из базы данных, но не отмените или измените информацию о владении файлами, существует вероятность того, что созданный в будущем новый пользователь получит освобожденный идентификатор и поэтому автоматически станет владельцем всех файлов удаленного пользователя.

Определение групп

Запись пользователя в файле /etc/passwd определяет группу, в которой он будет находиться при входе в систему, а в файле /etc/group определены все другие группы, членом которых является пользователь и в которые он может перейти с помощью утилиты newgrp. Как и в случае с именами и идентификаторами пользователя, возможности доступа программы к ресурсам системы определяются числовым эффективным идентификатором группы.

Рассмотрим следующий пример. Вы хотите обеспечить доступ к файлу /home/projects для участников группы разработки, но не желаете, чтобы другие пользователи имели доступ к нему. Для этого выполните следующие действия:

- добавьте группу с именем projects в файл /etc/group и включите в нее всех необходимых пользователей (более подробные сведения см. в следующем разделе);
- если вы хотите сделать эту группу в качестве группы по умолчанию, измените записи пользователей в файле /etc/passwd, указав для них новый идентификатор группы по умолчанию;
- измените идентификатор и права доступа группы у каталога /home/projects и его содержимого:

```
chgrp -R projects /home/projects
chmod -R g+rw /home/projects
```

- запретите доступ для всех остальных пользователей:

```
chmod –R o-rwx /home/projects
```

Более подробные сведения см. в подразд. "Владение файлами и права доступа" раздела 6.

Создание новой группы

Чтобы создать новую группу, откройте файл /etc/group в текстовом редакторе и добавьте строку, определяющую имя, идентификатор и членов новой группы. Например:

```
techies:x:101:michel,jim,sue
```

Более подробные сведения об указанных полях см. в разд. "Файл /etc/group" ранее в этом разделе.

Предупреждение. Эту работу следует выполнять во время простоя системы. Если во время изменения файла /etc/group посредством текстового редактора какое-либо приложение или утилита (например, ls -l, newgrp) попытается его прочитать, это может привести к некорректной работе.

Изменение существующей группы

Чтобы включить в группу нового пользователя (например, при создании новой учетной записи пользователя с помощью утилиты passwd), необходимо отредактировать файл /etc/group и добавить этого пользователя в соответствующую запись. Например, если вы хотите включить пользователя zeke в существующую группу с именем techies, измените строку:

```
techies:x:101:michel,jim,sue
```

на

```
techies:x:101:michel,jim,sue,zeke
```

Изменять файл /etc/group следует лишь при условии, что никакая программа или пользователь не обращаются к нему в данный момент.

3.5. Устранение неполадок

Далее рассмотрены некоторые проблемы, с которыми вы можете столкнуться при работе с паролями и учетными записями пользователей.

- утилита passwd дает сбой после изменения пароля.

При обновлении базы данных паролей утилита passwd блокирует ее с помощью файла /etc/.pwlock. Существование этого файла препятствует выполнению passwd.

Если в процессе обновления базы данных система терпит аварию и файл /etc/.pwlock сохраняется, утилита passwd не будет работать до тех пор, пока системный администратор не удалит его.

В случае повреждения файлов паролей при системном сбое администратору также следует использовать резервные файлы /etc/oshadow и /etc/opasswd для восстановления /etc/shadow и /etc/passwd, соответственно, чтобы предотвратить возможные проблемы.

- почему я не могу войти в систему в графическом режиме?

Если вы вводите имя пользователя и пароль по запросу утилиты входа графической оболочки (phlogin2 или phlogin), а в ответ вновь получаете диалоговое окно с пустыми полями, это может быть вызвано следующими причинами:

  - введенные имя пользователя и пароль не соответствуют никакой учетной записи системы (как имя пользователя, так и пароль должны вводиться с учетом регистра клавиатуры);
  - начальный командный интерпретатор, указанный в вашей учетной записи, не соответствует стандарту POSIX.

В любом случае следует обратиться за помощью к системному администратору.

- почему я не могу войти в систему в текстовом режиме?

Если при вводе имени пользователя и пароля по запросу утилиты login, обеспечивающей вход в систему в текстовом режиме, возникает сообщение "Login incorrect", скорее всего, это означает, что в системе отсутствует пользователь с указанным именем или введен неверный пароль. Следует иметь в виду, что имя пользователя и пароль зависят от регистра клавиатуры, поэтому проверьте, чтобы клавиша <Caps Lock> не была нажата.

Чтобы избежать подсказок неавторизованным пользователям, утилита login не сообщает, какие именно данные (имя пользователя или пароль) введены неверно. Если вам не удается разрешить проблему самостоятельно, системный

администратор (пользователь root) может установить новый пароль для вашей учетной записи.

Подобная проблема может возникать в случаях, когда отсутствует один или несколько файлов, хранящих информацию о паролях. Если системный администратор в данный момент обновляет эти файлы, то неполадка, скорее всего, временная. Попытайтесь войти в систему заново через 1—2 минуты. Если это не поможет, обратитесь за помощью к системному администратору.

Если вы являетесь системным администратором и не можете получить доступ к системе, попробуйте сделать это с другой машины ЗОСРВ «Нейтрино» через сеть Qnet, с инструментальной машины через интерфейс qconn, либо загрузите и запустите систему с установочного компакт-диска, чтобы обнаружить и устранить неполадку посредством командного интерпретатора.

- при входе в систему в текстовом режиме выводится сообщение: "No such file or directory".

Это означает, что система не может найти утилиту, указанную в качестве начального командного интерпретатора. Это может происходить по следующим причинам:

- утилиту не удалось найти в переменной окружения **PATH** (как правило, /bin:/usr/bin) утилиты login; укажите полное путевое имя программы в записи пользователя в файле /etc/passwd (например, /usr/local/bin/myprogram);

- в записи пользователя для начального командного интерпретатора указаны аргументы или параметры, что запрещено, поскольку вся строка интерпретируется как имя файла, подлежащего запуску.

Внимание! Использование средств защиты информации от несанкционированного доступа, разработанных в соответствии с требованиями российских руководящих документов описано в документе «Описание применения. Часть 2. Комплекс средств защиты» КПДА.10964-01 31 02.

4. Командная строка

4.1. Командная строка или графический интерфейс?

ЗОСРВ «Нейтрино» строится на интерфейсе командной строки. Хотя ЗОСРВ «Нейтрино» включает в себя простой в использовании графический интерфейс (см. п. 5), иногда, особенно системному администратору, требуется ввести команды. Рекомендации по выбору подходящего режима — графической оболочки Photon или командной строки — содержатся в п. 8.

В процессе разработки программ не всегда возникает потребность в использовании командной строки, т. к. комплект разработчика содержит интегрированную среду разработки (т. н. IDE-среду), которая графически отображает средства для написания, построения и тестирования программного кода. В процессе работы IDE-среды часто используются утилиты ЗОСРВ «Нейтрино», но при этом командная строка остается "скрытой" от пользователя.

Для использования командной строки в графической оболочке Photon следует запустить терминал pterm посредством щелчка мыши по пиктограмме Terminal

в группе инструментов графической оболочки Photon (расположенной у правой границы рабочего пространства пользователя).

Одновременно можно запускать несколько терминалов, при этом в каждом из них могут выполняться многозадачные процессы. Терминалы графической оболочки Photon эмулируют устройства символьного ввода/вывода, поэтому сведения, изложенные в даннм разделе, относятся к ним в той же мере, в какой и к физическим символьным устройствам.

4.2. Обработка команды

После ввода команды она последовательно интерпретируется несколькими процессами:

- драйвер устройства символьного ввода/вывода распознает нажатия таких клавиш, как <Backspace> и <Ctrl>+<C>;
- командный интерпретатор (command interpreter или shell) разбивает командную строку на лексемы (token), интерпретирует их и затем запускает те или иные утилиты;
- утилиты анализируют командную строку, полученную от командного интерпретатора, и выполняют соответствующие действия.

4.3. Драйверы символьных устройств

При вводе команды поток символов в первую очередь интерпретируется драйвером символьного устройства. Какой именно драйвер работает, в каждом случае зависит от используемого оборудования (более подробные сведения см. в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01).

Примечание. Обработка нажатий некоторых клавиш может отличаться от описанной в этом разделе в зависимости от конфигурации системы.

Режимы ввода

Драйверы устройств символьного ввода/вывода могут функционировать либо в режиме необрабатываемых ("сырых") входных данных (raw input mode), либо в каноническом режиме (canonical mode), или режиме редактируемых входных данных (edited input mode). В режиме "сырых" входных данных все символы передаются приложению по мере их ввода. В режиме редактируемых входных данных приложение получает символы только после завершения ввода всей строки (о чем, как правило, свидетельствует символ возврата каретки).

Поддержка терминалов

Некоторым программам (например, vi) требуется информация о том, какие действия может выполнять ваш терминал (возможно ли перемещение курсора, очистка экрана и т. д.). Переменная окружения TERM указывает на тип используемого терминала, а каталог /usr/lib/terminfo представляет собой базу данных терминалов. В этом каталоге содержится набор подкаталогов (от a до z), в

которых хранится информация для соответствующего терминала. Некоторые приложения вместо каталога /usr/lib/terminfo используют каталог /etc/termcap, представляющий собой устаревшую однофайловую модель базы данных.

Терминалом по умолчанию является qansi-m (аналог терминала ANSI). Более подробные сведения об установке типа терминала см. в подразд. "Типы терминалов" раздела 9.

Служба telnet

При использовании сетевой службы telnet для связи между двумя ЭВМ, для активизации восьмибитового тракта данных следует задавать параметр -8. При подключении к ЭВМ с ЗОСРВ «Нейтрино» из ЭВМ, управляемой какой-либо другой ОС, при некорректной работе терминала, необходимо завершить telnet и запустите эту утилиту снова с параметром -8.

Примечание. При запуске telnet из ОС Windows пользуйтесь режимом ansi или vt100, в зависимости от типа терминала.

Общие сведения о клавиатуре

В табл. 4.1 указано, как драйверы устройств символьного ввода/вывода интерпретируют нажатия различных клавиш и их сочетаний (т. е. групп одновременно нажатых клавиш). Драйверы обрабатывают нажатия клавиш сразу после их выполнения.

Примечание. Отклик системы на работу пользователя с клавиатурой может отличаться от описанного далее, если:

- драйвер функционирует в режиме "сырых", а не редактируемых входных данных;
- приложение устанавливает особые условия взаимодействия с пользователем (например, специальные правила обработки нажатий клавиш);
- терминал имеет ограниченные возможности клавиатуры.

При нажатии клавиши < [illegible] драйвер символьного ввода/вывода передает командному интерпретатору команду "назад" или "вперед", а он, в свою очередь, выполняет предыдущую или последующую команду.

Таблица 4.1

Действие	Клавиша или комбинация клавиш
Переместить курсор влево	< □>
Переместить курсор вправо	< □>
Переместить курсор в начало строки	<Home>
Переместить курсор в конец строки	<End>
Удалить символ слева от курсора	<Backspace>
Удалить символ в месте расположения курсора	<Del>
Удалить все символы в строке	<Ctrl>+<U>
Переключение между режимами вставки и замещения символов (если приложение их поддерживает)	<Ins>
Завершить ввод строки или начать новую строку	<Enter>
Вызвать команду повторно	< □> или < □>
Приостановить отображение вывода	<Ctrl>+<S>
Возобновить отображение вывода	<Ctrl>+<Q>
Попытаться прекратить выполнение процесса	<Ctrl>+<C> или <Ctrl>+<Break>
Указать на конец ввода (end of input, EOF)	<Ctrl>+<D>
Очистить терминал	<Ctrl>+<L>

Физические и виртуальные консоли

Адаптер дисплея, дисплей и системная клавиатура в совокупности носят название физической консоли, управляемой драйвером консоли.

Примечание. Некоторые системы не содержат драйвера консоли. Например, встраиваемые системы могут иметь только драйвер последовательного порта (devc-ser*). Драйверы devc-con и devc-tcon в настоящее время поддерживаются только на платформах x86.

Для возможности взаимодействия пользователя сразу с несколькими приложениями ЗОСРВ «Нейтрино» позволяет запускать несколько сеансов одновременно посредством виртуальных консолей. Эти виртуальные консоли обычно именуются /dev/con1, /dev/con2 и т. д. Графическая оболочка Photon обеспечивает виртуальные консоли даже при отсутствии драйвера консоли в системе (см. раздел 5).

При запуске драйвера devc-con параметр -n задает количество включаемых виртуальных консолей. В настольной системе файл построения образа (buildfile) задает создание четырех консолей при запуске diskboot. Более подробные сведения см. в описании diskboot в разделе 8. Максимальное количество виртуальных консолей равно девяти. Встраиваемый драйвер консоли devc-tcon поддерживает только одну консоль.

Привилегированный пользователь (root) может указать программу, которая должна первоначально запускаться на каждой консоли. Утилита инициализации терминала (tinit) читает содержимое каталога /etc/config/ttys, чтобы определить, какие программы нужно запустить на консоли. По умолчанию утилита tinit выполняет команду login только на первой консоли, но может запустить login и на любой другой выбранной пользователем консоли. Это означает, что консоль 1 доступна всегда, в то время как другие консоли остаются неактивными, пока пользователь не переключится на одну из них и не нажмет какую-нибудь клавишу.

Примечание. При увеличении числа виртуальных консолей не забудьте отредактировать файл /etc/config/ttys, чтобы утилита tinit получила информацию о том, какие программы должны запускаться на добавленных консолях.

На каждой виртуальной консоли может выполняться свое приоритетное приложение, которому требуется весь экран. Клавиатура связана с той виртуальной консолью, которая является видимой (т. е. активной) в данный момент. Пользователь может переключаться с одной консоли на другую и, следовательно, с одного приложения на другое посредством нажатия комбинаций клавиш (табл. 4.2).

Таблица 4.2

Действие	Комбинация клавиш
Перейти на следующую активную консоль	<Ctrl>+<Alt>+<Enter> или <Ctrl>+<Alt>+<+>
Перейти на предыдущую активную консоль	<Ctrl>+<Alt>+<–>

Примечание. Для этих комбинаций клавиш используйте + (плюс) и − (минус) на цифровой клавиатуре.

Для быстрого перехода на любую консоль можно использовать комбинации клавиш <Ctrl>+<Alt>+<n>, где n — цифра, которая соответствует номеру виртуальной консоли. Например, чтобы перейти к консоли /dev/con2 (если она существует), следует нажать <Ctrl>+<Alt>+<2>.

Когда пользователь завершает сеанс работы с консолью посредством ввода logout или exit или нажатия комбинации клавиш <Ctrl>+<D>, эта консоль опять становится нерабочей; она не появляется при циклическом переключении между консолями с помощью сочетаний клавиш. Исключением является консоль 1, на которой по завершении сеанса система обычно производит перезапуск login.

Более подробные сведения см. в описании devc-con в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01 и в разд. "Консольные устройства" главы 11 в «Описание применения. Часть 1. Системная архитектура» КПДА.10964-01 31 01

4.4. Командный интерпретатор

После того как драйвер устройства символьного ввода/вывода обработал введенную пользователем информацию, командная строка передается командному интерпретатору (shell).

По умолчанию командным интерпретатором является утилита sh, которая на самом деле в среде ЗОСРВ «Нейтрино» представляет собой ссылку на командный интерпретатор Korn (ksh). Кроме того, могут использоваться и другие командные интерпретаторы, в том числе компактные, подходящие для встраиваемых систем (см. "Shells (командные интерпретаторы)" в «Описание программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

Если описывать в общих чертах, командный интерпретатор разбивает командную строку на лексемы, интерпретирует их и затем вызывает запрошенные

пользователем программы. Детали зависят от используемого командного интерпретатора. В данном разделе описывается работа ksh.

Когда пользователь вводит данные через клавиатуру, командный интерпретатор Korn сразу же обрабатывает нажатия клавиш и редактирует командную строку, в том числе выполняет автоматическое завершение набора команд и имен файлов. После нажатия клавиши <Enter> командный интерпретатор выполняет следующие действия для обработки введенной командной строки:

- разбивает командную строку на лексемы, разделенные пробелами или специальными символами;
- из введенных слов командный интерпретатор формирует два вида команд:
- простые команды (simple commands) — как правило, это программы, запускаемые пользователем (например, less my_file);
- составные команды (compound commands) — зарезервированные слова, конструкции группировки (grouping constructs) и описания функций;
- в одной командной строке можно задать несколько команд;
- обрабатывает псевдонимы (aliases) рекурсивно;
- выполняет все необходимые подстановки (substitutions), в том числе для параметров, команд и имен файлов;
- производит необходимые перенаправления потоков;
- остальные команды выполняются по следующим приоритетам: специальные встроенные команды (special builtins), функции, регулярные встроенные команды (regular builtins), исполняемые модули.

Для изменения порядка обработки командной строки применяются кавычки (quoting), которые изменяют значение специальных символов.

В следующих разделах вы найдете краткие описания вышеперечисленных этапов обработки командной строки. Тем не менее нужно сказать, что оболочка ksh является очень мощным командным интерпретатором. Более подробные сведения см. в описании ksh в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

Редактирование командной строки

Для редактирования командной строки командный интерпретатор Korn поддерживает команды в стиле редактора emacs (табл. 4.3).

Таблица 4.3

Действие	Комбинация клавиш
Перейти в начало строки	<Ctrl>+<A>
Перейти в конец строки	<Ctrl>+<E>
Перейти в конец текущего слова	<Esc> <F>
Перейти в начало текущего слова	<Esc> <B>
Удалить символ в месте расположения курсора	<Ctrl>+<D>
Удалить символ слева от курсора	<Ctrl>+<H>
Удалить символы от места расположения курсора до конца текущего слова	<Esc> <D>
Удалить символы от места расположения курсора до конца строки	<Ctrl>+<K>
Вставить текст	<Ctrl>+<Y>

Как и в редакторе emacs, если для ввода команды применяется клавиша <Ctrl>, то, кроме нее, должна быть нажата и какая-то другая клавиша (что в целом образует сочетание одновременно нажатых клавиш). Если же для ввода команды применяется клавиша <Esc>, то в этом случае остальные клавиши нажимаются последовательно, одна после другой. Более подробно см. в разделе "emacs interactive input-line editing" в документации по ksh.

Для обработки этих команд командный интерпретатор ksh использует устройство символьного ввода/вывода в режиме необрабатываемого ввода, но при этом полностью эмулирует работу драйвера по обработке нажатий клавиш. Другие командные интерпретаторы (например, esh) взаимодействуют с устройством символьного ввода/вывода в каноническом режиме (редактируемый ввод данных).

Завершение набора команд и имен файлов

Пользователь может сократить количество нажатий клавиш благодаря механизму завершения набора команд (command completion) и имен файлов (filename completion). Для этого нужно ввести несколько символов, которые

однозначно идентифицируют команду или имя файла, и затем дважды нажать клавишу <Esc>. Система автоматически завершит набор, если это возможно. Например, если вы ввели последовательность символов appb и затем дважды нажали клавишу <Esc>, то система завершит набор имени команды: appbuilder.

Если вы хотите применять клавишу <Tab> для автоматического завершения набора, следует изменить настройку командного интерпретатора посредством ввода следующей команды:

```
bind '^I'=complete
```

либо в командной строке, либо в файле настроек ksh.

Более подробные сведения о команде bind и назначении клавиш см. в разделе "emacs interactive input-line editing" в документации по ksh в «Описание программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01. Более подробно о файлах настройки ksh также см. в разд. "Настройка командного интерпретатора" раздела 9.

Зарезервированные слова

Командный интерпретатор Korn распознает следующие зарезервированные слова и символы (табл. 4.4).

Он использует их для построения составных команд. Например, можно организовать выполнение команд в цикле:

```
for i in *.c; do cp $i $i.bak; done
```

Таблица 4.4

case	else	function	then	!
do	esac	if	time	[[
done	fi	in	until	{
elif	for	select	while	}

Ввод нескольких команд

Пользователь может ввести сразу несколько команд, отделяя их друг от друга точкой с запятой (;). Например, для определения текущего рабочего каталога используется команда pwd, а для просмотра содержимого каталога — команда ls. Обе команды можно объединить следующим образом:

```
pwd; ls
```

Кроме того, для объединения нескольких команд в одной командной строке можно использовать неименованные программные каналы (|) — см. в подразд. "Неименованные программные каналы" далее в этом разделе.

Псевдонимы

Командный интерпретатор позволяет создавать новые команды или назначать часто применяемые ключи с помощью псевдонимов (alias). Например, ключ –F команды ls позволяет отобразить в конце имен определенные символы, указывающие на то, что файл является исполняемым или представляет собой ссылку, каталог и т. д. Если вы хотите, чтобы команда ls всегда использовалась с этим ключом, создайте псевдоним:

```
alias ls='ls –F'
```

Если же вам понадобится выполнить команду ls без этого ключа, то укажите путь к исполняемому файлу или поставьте перед ней символ \ (обратный слэш): \ls.

Псевдонимы расширяются на месте, поэтому вы не можете установить аргумент в середину расширяемой формы; если вы хотите сделать это, используйте функцию командного интерпретатора. Например, если вы хотите, чтобы версия команды cd дополнительно сообщала вам где, какой каталог является текущим, наберите приблизительно следующее в ksh:

```
function my_cd
{
 cd $1
 pwd
}
```

Псевдонимы можно добавлять в файл настроек (более подробную информацию см. в подразд. "Файл запуска ksh" раздела 9).

Подстановки

Командный интерпретатор позволяет использовать сокращенные обозначения вместо различных элементов командной строки. Обработка подстановок производится в следующем порядке:

- каталоги (символ "тильда");
- параметры;
- команды;
- арифметические выражения;
- фигурные скобки;
- групповые символы для имен файлов.

Каталоги (символ "тильда")

Символ тильды (~) интерпретируется командным интерпретатором как ссылка на личный каталог пользователя. Ряд символов (если они есть) между тильдой и следующим слэшем интерпретируется как имя пользователя. Например, строка ~mary/some_file отсылает к файлу some_file в личном каталоге пользователя с именем mary.

Если имя пользователя не указано, то подразумевается имя текущего пользователя. Например, строка ~/some_file отсылает к файлу some_file в личном каталоге текущего пользователя.

Примечание. Ваш личный каталог определен в соответствующей записи в базе паролей (см. описание /etc/passwd в разделе 3).

Параметры

Чтобы включить значение параметра в командную строку, поставьте символ доллара ($) перед именем параметра. Например, для вывода значения переменной окружения PATH следует ввести:

```
echo $PATH
```

Команды

Иногда может понадобиться выполнить команду и использовать результат ее выполнения для другой команды. Это можно сделать таким образом:

$(команда)

или в устаревшей форме, с использованием обратной кавычки (backquote):

`команда`

Например, для поиска заданной цепочки символов string во всех C-файлах введите следующее:

```
grep string $(find . -name "*.c")
```

Команда find ищет в заданном каталоге (в данном случае .) и во всех его подкаталогах файлы, имена которых заканчиваются символами .c. В результате подстановки команда grep выполняет поиск заданной цепочки символов в файлах, найденных командой find.

Арифметические выражения

Арифметические выражения задаются в командной строке следующим образом:

$((выражение))

Например:

```
echo $((5 * 7))
```

Примечание. В арифметических выражениях допустимы только целочисленные значения.

Фигурные скобки

Фигурные скобки служат для добавления префикса, суффикса или того и другого к нескольким цепочкам символов, например:

```
[prefix]{str1,...,strN}[suffix]
```

где запятые (,) служат для разделения цепочек символов. Например, my_file.{c,o} преобразуется в my_file.c my_file.o.

Групповые символы для имен файлов

Групповые символы позволяют применять команды не к одному, а к множеству файлов или каталогов (табл. 4.5).

Таблица 4.5

Значение	Групповой символ
Искать совпадение с любым количеством символов	*
Искать совпадение с любым одним символом	?
Искать совпадение с любым из символов, перечисленных в квадратных скобках (или из диапазона символов, заданного с помощью дефиса)	[]
Исключить символы, заданные в скобках	!

Примечание. Скрытые файлы, т. е. файлы, имена которых начинаются с точки (например, .profile), не включаются в поиск по групповым символам, если символ точки не задан. Например, поиск по * не выдает .profile, в отличие от .*.

Примеры в табл. 4.6 иллюстрируют использование групповых символов для копирования групп файлов в каталог /tmp с помощью утилиты ср.

Таблица 4.6

Если ввести	Утилита ср скопирует
ср f* /tmp	Все файлы, имена которых начинаются с символа f (например, frd.c, flnt)
ср fred? /tmp	Все файлы, имена которых начинаются с fred и заканчиваются одним любым символом (например, freda, fred3)
ср fred[123] /tmp	Все файлы, имена которых начинаются с fred и заканчиваются символами 1, 2 или 3 (т. е. fred1, fred2 и fred3)
ср *.[ch] /tmp	Все файлы, имена которых заканчиваются символами .c или .h (например, frd.c, barn.h)
ср *.[!o] /tmp	Все файлы, имена которых не заканчиваются символами .o (Кроме тех файлов, имена которых не совпадают с шаблоном по количеству символов (в данном случае не будут скопированы файлы с именами, в которых после точки более одного символа, например .o1, .os или .oo)
ср *.{html,tex}	Все файлы, имена которых заканчиваются символами .html или .tex

Перенаправление ввода и вывода

Большинство команд:

- производит чтение из стандартного потока ввода (stdin, или файлового дескриптора 0), обычно назначенного клавиатуре;
- производит запись в стандартный поток вывода (stdout, или файлового дескриптора 1), обычно назначенный дисплею;
- производит запись сообщений об ошибках в стандартный поток ошибок (stderr, или файлового дескриптора 2), обычно также назначенный дисплею.

При необходимости вы можете отступить от этих правил и применить другие (табл. 4.7).

Таблица 4.7

Если необходимо	Используйте символ
Прочитать из файла или с другого устройства (перенаправление ввода)	<
Записать stdout в файл (перенаправление вывода)	>
Дописать stdout в конец файла	>>

Например, команда ls выводит список файлов каталога. Если требуется перенаправить вывод ls в файл с именем filelist, введите следующее:

```
ls > filelist
```

Также для перенаправления данных можно указать файловый дескриптор. Например, если вы не хотите выводить на экран сообщения об ошибках, перенаправьте stderr в dev/null (специальный файл, так называемое "нуль-устройство" (bit bucket), которое поглощает любые записываемые в него данные и при чтении возвращает символ конца файла (end-of-file)):

```
my_command 2> /dev/null
```

Более подробные сведения см. в разделе "Input/output redirection" в «Описание программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

Неименованные программные каналы

Также можно использовать неименованные программные каналы (pipe) (|) для формирования составных команд, например:

```
grep 'some term' *.html | sort –u | wc -l
```

Такие программы, как grep, sort и wc (утилита для подсчета символов, слов и строк), которые производят чтение из стандартного потока ввода и запись в стандартный поток вывода, носят название фильтров (filter).

Применение кавычек со специальными символами

Некоторые символы могут принимать особый смысл для командного интерпретатора в зависимости от контекста. Для того чтобы командный интерпретатор обработал включенные в командную строку специальные символы как обычные, скорее всего, потребуется отменить их особое значение с помощью кавычек (quote), см. также табл. 4.8.

Следующие символы должны быть заключены в кавычки, если необходимо исключить их специальный смысл:

| $ (") & ` ; \ ' табуляция новая_строка пробел

Следующие символы могут потребовать применения кавычек в зависимости от контекста:

* ? [# ~ = %

Таблица 4.8

Для отмены особого значения	Действие
Одиночного символа	Поставить перед ним один символ обратного слэша (\)
Всех специальных символов внутри цепочки символов	Заключить всю цепочку символов в одинарные кавычки
Всех специальных символов внутри цепочки символов, кроме $, ` и \	Заключить всю цепочку символов в двойные кавычки

Например, следующие команды производят поиск всех экземпляров цепочки символов "ZOSRV Neutrino" в файле chapter1.html:

```
grep ZOSRV\ Neutrino chapter1.html
grep 'ZOSRV Neutrino' chapter1.html
grep "ZOSRV Neutrino" chapter1.html
```

Однако команда:

```
grep ZOSRV Neutrino chapter1.html
```

будет выполнять поиск цепочки символов "ZOSRV" в файлах Neutrino и chapter1.html.

Сложные команды могут потребовать применения вложенных кавычек. Например:

```
find -name "*.html" | xargs grep -l '"ZOSRV.*Neutrino"' | less
```

Эта команда выдает список всех HTML-файлов, содержащих символы ZOSRV, за которыми следует цепочка любых символов, завершающаяся символами Neutrino. Команда find формирует список всех файлов с расширением html и передает его команде xargs, которая, в свою очередь, выполняет команду grep с каждым файлом из списка. Весь вывод команды xargs передается команде less, которая постранично отображает информацию на дисплее.

На примере данной команды рассмотрим, как кавычки применяются различным образом для управления обработкой специальных символов:

- если цепочку символов *.html не заключить в кавычки, то командный интерпретатор истолкует символ * и передаст команде find список файлов с расширением html в текущем каталоге. Если цепочку символов *.html заключить в кавычки, то командный интерпретатор передаст ее команде find "как есть" для поиска файла с заданным расширением в данном каталоге и всех его подкаталогах;

- аналогично если не заключить в кавычки цепочку символов ZOSRV.*Neutrino, то командный интерпретатор сгенерирует список файлов, имена которых соответствуют заданному шаблону. При использовании только двойных кавычек ("ZOSRV.*Neutrino") происходит однократный вызов команды grep. Однако приведенный пример, кроме того, усложнен командой xargs;

- команда xargs рассматривает командную строку как аргумент, а командный интерпретатор интерпретирует каждый элемент этой командной строки, передаваемый команде xargs. Если необходимо, чтобы командный интерпретатор не интерпретировал цепочку символов ZOSRV.*Neutrino, следует заключить эту цепочку символов (используемую командой grep в качестве шаблона) во вложенные кавычки:

```
xargs grep -l '"ZOSRV.*Neutrino"'
```

- кроме того, кавычки показывают, когда должна выполняться команда less. В данном случае командный интерпретатор передает команде less вывод всех вызовов команды xargs. Наоборот, команда:

```
find -name "*.html" | xargs 'grep -l "ZOSRV.*Neutrino" | less'
```

передает команду:

```
grep -l "ZOSRV.*Neutrino" | less
```

команде xargs, которая будет давать совершенно иные результаты (если она вообще будет работать).

Более подробные сведения см. в разделе "Заключение в кавычки" в документации по ksh «Описание программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

Повторный вызов ранее введенных команд

Командный интерпретатор позволяет повторно вызывать ранее введенные команды. Для перемещения по истории сохраненных команд (history buffer) используйте клавиши <↑> и <↓>. Вы можете отредактировать команду, если необходимо, и затем нажать клавишу <Enter>, чтобы выполнить ее снова.

Командный интерпретатор также имеет встроенную команду fc, которая предназначена для отображения и редактирования ранее введенных команд, а также псевдоним r команды fc, который служит для повторного выполнения ранее использованной команды. Например:

```
r последовательность_символов
```

повторно выполняет последнюю команду, имя которой начинается с последовательности_символов.

Сценарии командного интерпретатора

Команды командного интерпретатора можно сохранить в текстовом файле (называемом сценарием (shell script)), предназначенном для исполнения в пакетном режиме (batch mode). Более подробные сведения см. в разделе 10.

4.5. Утилиты

После обработки специальных символов командным интерпретатором остается информация, которая, как правило, состоит из команд и их аргументов. Большинство команд реализуется с помощью соответствующих системных исполняемых файлов, но некоторые команды (например, cd) встроены в командный интерпретатор.

Пользователь может хранить в системе несколько исполняемых файлов с одним именем. С помощью переменной окружения **PATH** командный интерпретатор определяет, какую версию следует использовать.

Переменная **PATH** содержит список каталогов, разделенных двоеточиями (:) и расположенных в том порядке, в котором вы хотите, чтобы командный интерпретатор выполнял поиск исполняемых файлов. Для просмотра значения переменной PATH следует ввести:

```
echo $PATH
```

Предупреждение. В переменную **PATH** можно включить текущий каталог (.), но это может привести к уязвимости перед вирусами типа "троянский конь". Например, если . стоит в начале списка, хранимого в PATH, то командный интерпретатор будет выполнять поиск в первую очередь в текущем каталоге. Злонамеренный пользователь может записать в каталог программу-ловушку с именем ls.

Если вы хотите включить текущий каталог в переменную **PATH**, поместите его после каталогов, содержащих часто используемые утилиты.

Для получения сведений о задании переменной **PATH** см. подразд. "Типы терминалов" раздела 9.

Чтобы определить, какую именно версию команды выберет командный интерпретатор, используйте команду which. Например:

```
$ which ls
/bin/ls
```

Чтобы получить более подробные сведения, можно использовать ключи командной строки:

```
$ which –laf ls
-rwxrwxr–x 1 root    root        19272 May 03 2002 /bin/ls
```

Если же применить это к команде, которая встроена в интерпретатор, то which не сможет найти ее:

```
$ which cd
which: no cd in /bin:/usr/bin:/usr/photon/bin:/opt/bin
```

Команда whence отображает информацию о том, как заданная команда или псевдоним интерпретируется командным интерпретатором. Например, если существует псевдоним для ls, то результат может быть, например, следующим:

```
$ whence ls
'ls –F'
```

Синтаксис команд

В «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01 каждая команда сопровождается синтаксическим описанием, поясняющем, как пользоваться командой. Для большинства команд это описание состоит из следующих частей:

- имя_команды (command_name) – имя команды, которая должна быть выполнена. Это может быть имя выполняемой программы (например, утилиты) или имя команды, встроенной в командный интерпретатор;
- ключи (options) – с помощью ключей можно изменять условия выполнения команды. Ключи обычно состоят из буквенно-цифрового символа,

перед которым стоит дефис (например, -c). Некоторые ключи должны иметь аргумент (например, -n число). Если вы применяете ключ, который предполагает аргумент, то должны его указать;

- операнды (operands) – это данные, которые требуются для выполнения команды (например, имя файла). Если команда допускает множество операндов, они обычно обрабатываются в порядке их перечисления. В отличие от ключей, операнды не должны начинаться с дефиса (например, more my_file).

В «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01 для описания синтаксиса команд используются некоторые специальные символы:

- … – этот символ означает, что можно задать один или несколько экземпляров предшествующего элемента. Например, в соответствии с синтаксисом утилиты more, многоточие после операнда file указывает на то, что для этой команды допустимо задать более одного файла, например:

```
more myfile1 myfile2
```

- [] – элемент, заключенный в квадратные скобки, является необязательным;
- | – этот символ используется для разделения альтернативных элементов (например, -a|-f).

Конечно, при вызове команды вам не нужно вводить эти символы. Например, команда more имеет следующее описание синтаксиса:

```
more [-ceisu] [-n number] [-p pattern]
    [-/ pattern] [-t tag] [-x tabstop] [file...]
```

Ключи, которые не требуют аргумента, можно объединять. Например, вышеприведенная цепочка -ceisu означает сокращение от -c -e -i –s -u.

Если аргумент команды начинается с дефиса, то для обозначения конца списка ключей следует указать двойной дефис, например:

```
ls -l -- -my_file
```

Получение справочной информации в интерактивном режиме

Подробные описания утилит можно найти в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01. Однако если вам нужно получить краткую подсказку по синтаксису и допустимым ключам команды, вы можете воспользоваться командой use (аналогична команде man в UNIX и Linux), чтобы получить справочную информацию в интерактивном режиме (online usage message) по использованию утилиты. Например, для получения сведений о команде more введите:

```
use more
```

Если вы запрашиваете сведения об использовании программы, которая не находится ни в одном из каталогов, перечисленных в переменной **PATH** или не содержит в себе справочное сообщение, команда use выдаст сообщение об ошибке. Чтобы получить более подробные сведения о команде use, воспользуйтесь руководством «Описание программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01 или просто введите use use.

Выполнение команд на другом узле или tty-устройстве

Если машины работают в сети Qnet (см. раздел 12), вы можете выполнять команды на другой машине, т. е. применять удаленное выполнение. Например:

```
on -n /net/dasher date
```

где /net/dasher означает имя узла, на котором команда должна быть выполнена.

Когда вы вызываете команду на другом узле, ее стандартные потоки ввода, вывода и ошибок отображаются на экране консоли (или терминале), за исключением случаев, когда эти потоки явно перенаправлены на другое устройство.

Для выполнения команды на tty-устройстве следует задать ключ -t и имя терминала. Например:

```
on -t con3 login root
```

Более подробные сведения см. в описании команды on в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

Приоритеты

По умолчанию запускаемая пользователем утилита или другая программа получает тот же приоритет, что и родительский процесс. (В действительности приоритеты назначаются не процессам, а потокам.) Определить приоритет какого-либо потока можно с помощью команды pidin (сокр. от Process ID INformation).

Чтобы задать приоритет вручную, используйте команду on с ключом -p. Чтобы задать относительный приоритет, используйте команду nice.

4.6. Основные команды

В табл. 4.9 приведен список наиболее часто используемых команд ЗОСРВ «Нейтрино».

Таблица 4.9

Действие	**Команда**
Определить текущий каталог	pwd (встроенная команда интерпретатора ksh)
Изменить каталог	cd (встроенная команда интерпретатора ksh)
Вывести список содержимого каталога	ls
Переименовать (переместить) файлы и каталоги	mv
Удалить файлы	rm
Копировать файлы и файловые иерархии	cp или pax
Создать каталоги	mkdir
Удалить каталоги	rmdir
Объединить и отобразить файлы	cat
Отобразить вывод постранично	less или more
Найти файлы в соответствии с заданным критерием поиска	find
Изменить права доступа/атрибуты файла	chmod
Создать жесткую или символьную ссылку	ln
Создать "архив на магнитной ленте" ("tape archive")	tar или pax
Извлечь файлы из файла .tar	tar

Таблица 4.9

Действие	**Команда**
Извлечь файлы из файла .tar.gz или .tgz	gunzip имя_файла | pax –r или tar –xzf имя_файла

Языковые раскладки клавиатуры

Для изменения языковой раскладки клавиатуры в графической оболочке Photon вы можете использовать команду phlocale (которая также позволяет изменять некоторые другие настройки языковой поддержки). В каталоге /usr/photon/keyboard находятся раскладки для разных языков (например, немецкого, французского), а также разных версий клавиатуры Дворжака (некоторые считают эту схему клавиатуры более эффективной, чем стандартная схема QWERTY).

Если необходимая раскладка не поддерживается, вы можете воспользоваться командой mkkbd для создания новой раскладки.

Некоторые схемы клавиатур (например, для французского и немецкого языков) включают клавиши для добавления знаков акцента, которые не создают отдельных символов. В ЗОСРВ «Нейтрино» такие клавиши считаются "слепыми". При нажатии "слепой" клавиши, после которого следует нажатие другой клавиши, происходит модификация, в результате которой второй символ получает знак акцента. Например, для ввода символа Ü во французской схеме клавиатуры следует нажать <Shift>+<[> и затем <Shift>+<U>.

Кроме того, сложные символы можно создавать путем нажатия и отпускания клавиши <Alt> с последующим нажатием двух клавиш или сочетания клавиш. Например, для ввода символа Ü в английской (США) схеме клавиатуры вы можете нажать и отпустить клавишу <Alt> и затем применить комбинацию клавиш <Shift>+<"> и после него – <Shift>+<U>.

4.7. ЗОСРВ «Нейтрино» для пользователей MS-DOS

Пользователю, знакомому с Microsoft Windows, может быть, полезно знать эквиваленты основных команд и переменных DOS, существующих в ЗОСРВ «Нейтрино».

Команды DOS и их эквиваленты в ЗОСРВ «Нейтрино»

В табл. 4.10 приведены некоторые основные команды MS-DOS и их эквиваленты в ЗОСРВ «Нейтрино». Более подробные сведения о командах ЗОСРВ «Нейтрино» см. в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

Таблица 4.10

Команда DOS	**Команда ЗОСРВ «Нейтрино»**
attrib	ls -l, chmod и ls -a
Командные файлы	Сценарии командного интерпретатора (см. раздел 10 или документацию по командному интерпретатору ksh)
cacls	ls -l
call сценарий	ksh сценарий Если сценарий начинается с #!/bin/sh, его можно вызывать как обычную программу; например, сценарий (без добавления префикса sh или ksh)
chdir	cd (встроенная команда интерпретатора ksh)
chkdsk	Для файловой системы QNX 4 применяется команда chkfsys; Для файловой системы FAT применяется команда chkdosfs
cls	clear
cmd	ksh
command	ksh
comp	cmp или diff
copy	cp или pax
date	date и rtc Для изменения даты и времени аппаратных часов следует использовать команду rtc
del	rm
dir	ls
erase	rm
diskcomp	См. далее
diskpart	fdisk [команда]
driverquery	См. подразд. "Устранение неполадок" раздела 6
fc	cmp или diff
find	grep -i
findstr	grep
format	fdformat и dinit
ftype	Ассоциирование типов файлов выполняется программой Photon File Manager (pfm). См. подразд. "Просмотр файлов с

Таблица 4.10

Команда DOS	**Команда ЗОСРВ «Нейтрино»**
	помощью администратора файлов" раздела 5
getmac	См. ifconfig, netstat; а также ls /dev/io-net
help	use
logman	tracelogger
lpq	lprq
lpr	lpr
md	mkdir
mode	stty
move	mv
msiexec	qnxinstall
path	echo $PATH, export PATH=новый путь (см. подразд. "Утилиты" ранее в этом разделе или документацию по командному интерпретатору ksh)
print	lpr
query	sin, pidin и ps
rem	#
rename	mv
replace	cp -x
runas	su
schtasks	crontab
shutdown	shutdown
sort	sort
taskkill	kill или slay
tasklist	sin, pidin и ps
time	date и rtc
tracerpt	traceprinter
tracert	traceroute
type	cat
ver	uname -a
xcopy	cp или pax

diskcomp

Следующие команды ЗОСРВ «Нейтрино» являются эквивалентом команды DOS diskcomp:

- скопировать исходный диск в файл:

```
cp /dev/fd0 referencecopy
```

- сравнить другие диски с созданной копией файла:

```
cmp referencecopy /dev/fd0
```

- скопировать эталонный файл на новую дискету:

```
cp referencecopy /dev/fd0
```

Эквиваленты локальных переменных командного интерпретатора MS-DOS

В табл. 4.11 приведены некоторые встроенные локальные переменные командного интерпретатора MS-DOS и их эквиваленты (переменные окружения и команды) в ЗОСРВ «Нейтрино».

Таблица 4.11

Переменные командного интерпретатора DOS	**Эквивалент в ЗОСРВ «Нейтрино»**
%CD%	Переменная окружения **PWD**, pwd
%COMPUTERNAME%	Переменная окружения **HOSTNAME**
%COMSPEC%	Переменная окружения **SHELL**
%DATE%	Запуск утилиты date: $(date)
%ERRORLEVEL%	$? (см. раздел "Parameters" в документации по командному интерпретатору ksh)
%HOMEDRIVE%	В ЗОСРВ «Нейтрино» обозначения дисков не используются. См. %HOMEPATH%
%HOMEPATH%	Переменная окружения **HOME**
%OS%	Запуск утилиты uname: $(uname)
%PATH%	Переменная окружения **PATH**
%PATHEXT%	В ЗОСРВ «Нейтрино» расширения файлов рассматриваются как часть имени файла. Разрешение на выполнение определяется правами доступа к файлу. См. chmod
%PROCESSOR_ARCHITECTURE%	Запуск утилиты uname: $(uname -p)
%PROCESSOR_IDENTIFIER%	Запуск утилиты sin: $(sin info)
%PROMPT%	Переменные окружения **PS1**, **PS2** (см. раздел "Parameters" в документации по командному интерпретатору ksh и разд. "Файл .kshrc" приложения)

Таблица 4.11

Переменные командного интерпретатора DOS	**Эквивалент в ЗОСРВ «Нейтрино»**
%RANDOM%	Переменная окружения **RANDOM**
%SYSTEMDRIVE%	В ЗОСРВ «Нейтрино» обозначения дисков не используются. Корнем системы всегда является /
%SYSTEMROOT%	Корнем системы всегда является /
%TEMP%	Переменная окружения **TMPDIR**
%TMP%	Переменная окружения **TMPDIR**
%TIME%	Запуск утилиты date: $(date)
%USERNAME%	Переменная окружения **LOGNAME**

4.8. Устранение неполадок

Далее приводятся ответы на некоторые вопросы из тех, которые могут у вас возникнуть при работе с командной строкой.

- почему я не могу запустить мою программу с именем test?

Командный интерпретатор имеет встроенную команду с именем test. При анализе командной строки командный интерпретатор в первую очередь производит поиск по встроенным командам, а затем среди исполняемых файлов.

Переименуйте вашу программу либо укажите путь к ней (например, ./test).

- почему при попытке запуска моей программы я получаю сообщение "not found" ("не найдено")?

Возможно, каталог, где находится ваша программа, не указан в переменной окружения **PATH**. Например, по соображениям безопасности в ней не указан текущий каталог.

Для решения проблемы включите каталог с вашей программой в переменную окружения **PATH** либо укажите путь к команде (например, ./my_program). Более подробные сведения см. в подразд. "Утилиты" ранее в этом разделе.

- почему пользователь root имеет иной набор доступных команд?

Привилегированный пользователь (root) имеет другое значение переменной окружения PATH, которое включает в себя такие каталоги, как /sbin и /usr/sbin. Эти

каталоги содержат исполняемые модули и администраторы, которые, как правило, может использовать только root.

Если вы вошли в систему не как привилегированный пользователь, вы, тем не менее, можете запускать некоторые утилиты из каталога /sbin, если имеете соответствующие разрешения, но при этом вы должны либо указать полный путь (например, /sbin/logger), либо добавить каталог в переменную окружения PATH.

- когда я просматриваю список файлов в каталоге, то не вижу файлы, имена которых начинаются с точки.

Файлы, имена которых начинаются с точки (.), называются скрытыми файлами (hidden files). Для того чтобы они появились в списке, используйте команду ls с ключом -a.

- почему я получаю сообщение "No such file or directory" ("Не найден файл или каталог с таким именем")?

Командный интерпретатор не может найти заданный файл или каталог. В этом случае попробуйте проверить следующее:

- верно ли вы указали имя? В ЗОСРВ «Нейтрино» имена файлов и каталогов зависят от регистра;

- содержит ли имя пробелы или другие специальные символы? Если файл называется my file и смысл пробела не изменен, то командный интерпретатор учитывает этот пробел при разбиении командной строки на лексемы, поэтому производится поиск не одного, а двух файлов с именами my и file.

Для изменения роли специальных символов применяйте маскирование с помощью кавычек (например, less "my file" или less my\ file). Более подробные сведения см. в разд. "Применение кавычек со специальными символами" ранее в этом разделе.

- как работать с файлом, имя которого начинается с дефиса?

В ЗОСРВ «Нейтрино» дефис (-) служит для обозначения ключей (например, head -n 10 имя_файла). Если вы создаете файл, имя которого начинается с дефиса, и передаете это имя утилите в качестве аргумента, то она рассматривает это имя как один или несколько ключей.

Большинство утилит распознает двойной дефис (--) как признак "конца цепочки ключей" ("end of options"), поэтому вы можете указать его перед именем вашего файла:

```
head -- -my_file
```

- почему я получаю сообщение "Unrecognized TERM type" ("Неизвестный тип терминала") при запуске программ (например, vi)?

Либо переменная окружения **TERM** задана некорректно, либо нет записи о данном типе терминала в базе данных терминалов /usr/lib/terminfo (или, возможно, /etc/termcap). См. подразд. "Поддержка терминалов" ранее в этом разделе.

5. Графическая оболочка Photon microGUI

5.1. Обзор графической оболочки Photon

Графическая оболочка Photon microGUI является графическим пользовательским интерфейсом операционной системы ЗОСРВ «Нейтрино», который можно использовать как любую другую графическую среду настольных компьютеров, т. е. пользователь может работать в ней с приложениями посредством графических окон, выполнять операции выбора, нажатия и перетаскивания с помощью мыши, отображать каталоги и файлы графически в виде древовидной иерархии, просматривать мультимедийные файлы и т. д. Графическая оболочка Photon также служит в качестве базовой среды для графических приложений во встраиваемых системах.

Многие приложения и утилиты, входящие в состав графической оболочки Photon, документированы в «Описание программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01. Более подробные сведения о программировании Photon-приложений см. в "Photon Programmer’s Guide".

Почему "Photon"?

Каждый раз, когда пользователь щелкает кнопкой мыши или нажимает клавишу, производится ввод данных для Photon-приложения. И каждый раз, когда приложение отображает данные в окне, оно осуществляет вывод данных. Все эти операции взаимодействия обрабатываются в виде очень небольших пакетов данных, называемых событиями (event). Эти события ввода и вывода, перемещающиеся между пользователем и графической оболочкой Photon, можно представить себе в виде фотонов (photon) – световых частиц.

Почему "microGUI"?

Графическая оболочка Photon называется microGUI (графический пользовательский микроинтерфейс) из-за своего размера и архитектуры. Photon — очень компактный графический пользовательский интерфейс. Он спроектирован для встраиваемых систем, и, кроме того, его архитектура расширяема. В итоге

графическая оболочка Photon идеально подходит для сложных, высокопроизводительных, распределенных систем.

Как и сама операционная система ЗОСРВ «Нейтрино», графическая оболочка Photon основана на микроядре. Такая модульная архитектура делает Photon быстрым и гибким, а также изначально предназначенным для сетевых распределенных систем.

Рабочее пространство

При первом запуске графической оболочки Photon пользователю предлагается задать настройки оболочки и графической платы.

После запуска Photon на экране отображается рабочее пространство – область, в которой пользователь работает с приложениями (рис. 5.1). Рабочее пространство включает в себя панель задач (taskbar), системную панель (shelf) и рабочий стол (desktop).

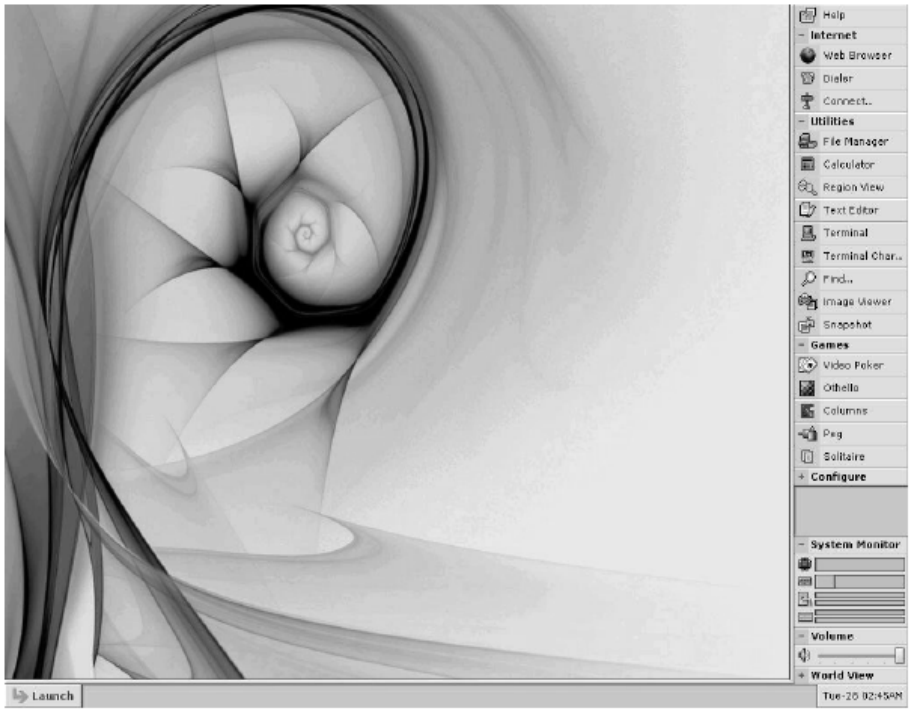


Рис. 5.1. Рабочее пространство графической оболочки Photon с панелью задач, системной панелью и рабочим столом

Рабочий стол занимает основную часть экрана. На нем отображаются окна с приложениями. В графической оболочке Photon рабочий стол по сути представляет

собой виртуальную консоль (virtual console), которая отображает часть более общего пространства рабочего стола, втрое большего по ширине и высоте. Пользователь может работать с приложениями в разных консолях и переключаться между консолями с помощью комбинаций клавиш или инструмента **World View** на системной панели.

С помощью щелчка правой кнопкой мыши в любом месте рабочего стола вызывается меню **Desktop Menu** (рис. 5.2), которое помогает запускать часто используемые приложения, конфигурировать графическую оболочку Photon и завершать работу системы. Пользователь может различным образом настраивать **Desktop Menu** (см. далее).

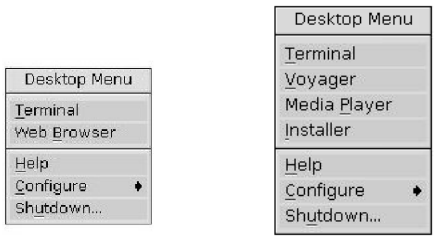


Рис. 5.2. Меню рабочего стола **Desktop Menu**

Панель задач представляет собой область, расположенную внизу экрана. По умолчанию она отображает кнопку **Launch**, дату и время, а также пиктограммы выполняемых в текущий момент приложений.

Панель задач позволяет:

- щелкать мышью по пиктограммам, чтобы скрывать или отображать окна приложений;
- щелкать мышью по кнопке **Launch**, чтобы запускать приложения, вызывать справку или завершать сеанс работы с графической оболочкой Photon.

Системная панель расположена вдоль правой стороны экрана, и с ее помощью пользователь может легко запускать часто используемые приложения и утилиты, конфигурировать систему, просматривать данные об использовании ресурсов и переключаться между консолями.

Системная панель позволяет:

- разворачивать и сворачивать компоненты системной панели с помощью мыши по категориям компонентов; рядом с развернутыми категориями отображен знак +, а рядом со свернутыми – знак –;
- запускать приложения, утилиты и конфигурационные утилиты с помощью щелчков мышью по их пиктограммам;
- менять текущую консоль с помощью щелчка мышью по нужной консоли в **World View**;

Примечание. Переключаться между консолями также можно с помощью комбинаций клавиш <Ctrl>+<Alt>+<1>...<9>, где цифра соответствует номеру консоли.

- управлять компонентом CD Player.

С помощью щелчка правой кнопкой мыши по панели задач или системной панели можно настроить или завершить приложение shelf. Чтобы запустить или перезапустить приложение shelf, следует ввести команду shelf & в командной строке.

Чтобы изменить размер панели задач или системной панели, нужно с помощью мыши переместить ее границу. Если переместить границу к нижнему или правому краю экрана, включится режим автоматического скрытия панели, в котором панель отображается только при наведении указателя мыши на границу экрана.

5.2. Настройка системной панели

Сконфигурировать системную панель можно с помощью щелчка правой кнопкой мыши по системной панели или панели задач и выбора пункта **Setup** в контекстном меню, либо с помощью команды shelf -c, выполненной в командной строке.

Диалоговое окно для конфигурирования системной панели показано на рис. 5.3.

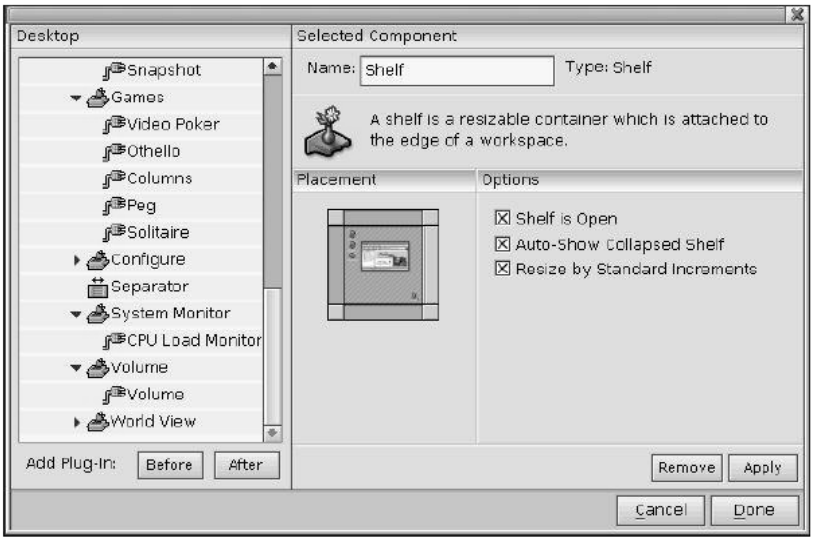


Рис. 5.3. Диалоговое окно для конфигурирования системной панели

Примечание. При конфигурировании системной панели новые настройки сохраняются только для текущего пользователя. Конфигурационный файл системной панели $HOME/.ph/shelf/shelf.cfg хранится в домашнем каталоге пользователя. При первом входе в систему версия по умолчанию /etc/photon/shelf/shelf.cfg копируется в файл $HOME/.ph/shelf/shelf.cfg.

Следующие элементы можно добавлять или изменять на системной панели:

- **Group** (Группа) – группа приложений или утилит. Группа может содержать подгруппы;
- **Drawer** (Ящик) – ящик похож на группу, но развертывается в системной панели горизонтально из родительского контейнера, а не вертикально;
- **Separator** (Разделитель) – пространство, которое визуально разделяет два контейнера. Все неиспользуемое пространство системной панели заполняется разделителем, поэтому на ней всегда есть как минимум один разделитель. Если попытаться удалить разделитель, он восстанавливается;
- **World view** (Обозреватель рабочих столов) — подключаемый модуль (плагин), который позволяет видеть, какие консоли содержат открытые окна, и задавать текущую консоль;
- **CD player** (Проигрыватель компакт-дисков) — плагин для воспроизведения компакт-дисков;

- **Volume** (Регулятор громкости) — плагин для управления громкостью.

С помощью кнопки **Browse** можно выбирать дополнительные плагины, в том числе:

- cdplayer.so — плагин, который выполняет функции воспроизведения компакт-диска, остановки и перемотки дорожек вперед и назад. В текстовом поле плагина отображается информация о дорожке;
- clock.so — плагин для настройки шрифта часов и его размера, включения и отключения отображения даты и секунд, установки формата даты и времени (12- или 24-часовой). Если выбрать этот плагин на системной панели, запустится утилита User Configuration, которая позволяет задавать и настраивать дату и время;
- launcher.so — плагин, с помощью которого можно создать на системной панели элемент для запуска выбранной команды;
- launchmenu.so — плагин, который поддерживает меню Launch. В каждый момент времени на системной панели может находиться только одно такое меню. Попытка добавить в системную панель второе меню Launch игнорируется. Чтобы изменить расположение меню Launch, необходимо сначала удалить его, а затем добавить новое меню Launch в другом месте.

Следует иметь в виду, что этот плагин не работает в ящике (drawer) на системной панели; он должен находиться на верхнем уровне системной панели. Более подробные сведения о настройке содержимого меню **Launch** см. в подразд. "Настройка меню Launch" далее в этом разделе;

- led.so — набор из трех "светодиодов", которые отображают состояние клавиш <Num Lock>, <Caps Lock> и <Scroll Lock>. Индикатор горит, если соответствующая клавиша включена;
- pload.so — монитор загрузки центрального процессора (CPU Load Monitor), который выводит гистограммы, отображающие уровни загрузки центрального процессора, использования памяти и активности диска и сети;
- ptrcam.so — этот плагин реализует "видеокамеру указателя мыши", которая увеличивает изображение, находящееся непосредственно под указателем.

Пользователь может задать горизонтальный и вертикальный радиус "видеокамеры" в пикселах;

- taskbar.so — панель задач, которая позволяет легко переключаться между приложениями, выбирая соответствующие пиктограммы. Пользователь может изменять шрифт, размер шрифта, цвет активных и неактивных приложений, а также включать и отключать отображение всплывающих подсказок на элементах панели задач;
- volume.so — бегунок, с помощью которого можно управлять громкостью звука, поступающего от звуковой платы. Для выключения или включения звука в этом плагине служит небольшая пиктограмма громкоговорителя;
- worldview.so — миниатюрная панель из девяти виртуальных консолей. Пользователь может выбрать режим отображения границ окон: всегда, никогда или в зависимости от размера World View.

5.3. Настройка меню Launch

Плагин launchmenu.so заполняет меню Launch на основе содержимого каталогов $HOME/.ph/launchmenu и /etc/photon/launchmenu.

Примечание. В случае конфликта между элементами меню приоритет имеет элемент, который найден первым, т. е. элементы каталога launchmenu, находящегося в домашнем каталоге, имеют приоритет над элементами глобального каталога launchmenu.

Создание элементов меню и подменю

В каталогах $HOME/.ph/launchmenu и /etc/photon/launchmenu каждый подкаталог соответствует подменю, а каждый файл или символьная ссылка — элементу меню. Однако применяются следующие исключения:

- файлы с именами .menu содержат специальные команды форматирования меню (см. разд. "Дополнительное управление меню" далее в этом разделе);
- файлы с расширением .tgt содержат описания исполняемых целевых файлов (см. разд. "Файлы описания запуска целей" далее в этом разделе);
- другие элементы, начинающиеся с точки (.), игнорируются.

Для всех прочих файлов плагин создает элементы меню. Если пользователь выбирает элемент меню, происходит следующее:

- если файл является символьной ссылкой, то она заменяется на имя исходного файла;
- если файл является исполняемым, то плагин выполняет его;
- если файл не является исполняемым, плагин пытается обнаружить подходящее приложение для просмотра этого файла, используя механизм связывания файлов в графической оболочке Photon (см. описание утилиты pfm в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01);
- во всех других случаях никакие действия не выполняются.

Для всех элементов (кроме *.tgt) плагин launchmenu.so использует имя файла в качестве текста, отображаемого в соответствующем элементе меню. Имя файла может состоять из любых символов (в рамках ограничений используемой файловой системы), при этом принимается, что имена файлов приведены в кодировке UTF-8.

Символ амперсанда (&) имеет особый смысл: плагин launchmenu.so интерпретирует следующий за ним символ как клавишу быстрого вызова данного элемента меню. Чтобы отобразить сам амперсанд, следует задать его в имени файла как &&.

Файлы описания запуска целей

Плагин launchmenu.so использует файлы описания запуска целей (*.tgt) вместо простых файлов, что делает более гибким управление запускаемыми целями (target) и их представлением в меню. В файле описания можно указать один или более исполняемых объектов, каждый из которых соответствует одному элементу меню. Описание целей имеет следующую форму:

```
[элемент1_текст]
target = действие
...
[элемент2_текст]
target = действие
```

. . .

Файлы описания структурированы в виде одного или более разделов, каждый из которых определяет цель. Квадратные скобки являются частью синтаксиса файла. В них заключен текст, который по умолчанию применяется для элемента меню и соответствует рассмотренным ранее соглашениям для имен файлов.

Каждая целевая система описывается парами ключ=значение внутри раздела. Пользователь должен задать пару target=действие, которая определяет действие, выполняемое при вызове элемента меню. В качестве действия может быть одно из следующих:

- полный путь к другому файлу или каталогу. Плагин launchmenu.so проверяет этот файл и обрабатывает его соответствующим образом. Путь может указывать на исполняемый файл, обычный файл, каталог или другой файл описания;
- имя исполняемого файла, который можно найти в каталогах, указанных в переменной окружения **PATH**;
- команда в формате командного интерпретатора, которая имеет вид:

```
перем1=знач1... параметры_команды
```

Если ключ target не указан, плагин launchmenu.so пропускает раздел при генерации элементов меню.

Следующие ключи являются необязательными:

- sicon — полный путь к небольшой (размером не более 24x18 пикселов) пиктограмме, которая отображается для элемента меню. Если пиктограмма не указана, плагин launchmenu.so пытается найти ее с помощью механизма связывания или извлечь пиктограмму из исполняемого файла (если элемент меню является PhAB-приложением);
- licon — полный путь к большой (размером до 48x48 пикселов) пиктограмме, которая отображается на элементе меню;

- group — эта запись позволяет логически группировать элементы из разных файлов описания и служит для упорядочения элементов. Более подробные сведения см. в следующем разделе;
- order — эта запись позволяет определять порядок элементов, как правило, в сочетании с записью group.

Как отмечено ранее, название раздела задает текст, который по умолчанию отображается на элементе меню. Если требуется создать элементы меню на разных языках, можно указать запись, ключом которой является код языка, используемый переменной окружения **ABLANG** (см. главу "International Language Support Photon" в руководстве программиста Photon "Programmer's Guide"), а значением — текст на этом языке. Например:

```
[Calculator]
target = phcalc
fr_FR = Calculatrice
```

Упорядочение элементов

По умолчанию плагин launchmenu.so сортирует элементы по отображаемому тексту в алфавитно-цифровом порядке. Кроме того, он предоставляет некоторые возможности управления сортировкой элементов с помощью описания цели. Если в систему добавляется пакет, который включает в себя множество элементов меню, и эти элементы необходимо упорядочить определенным образом независимо от их имен (например, пользователь считает некоторые элементы особо важными и желает расположить их первыми), можно обеспечить это упорядочение одним из следующих способов.

Множественные файлы описания.

Если цели по какой-либо причине распределены среди множества файлов описания, то необходимо применить логическую группировку для сортирования элементов. Для этого в цели указывается запись group. Ее значением может быть любая символьная строка, однако во избежание потенциальных конфликтов рекомендуется соблюдать следующее правило:

```
Название компании:Семейство продуктов:Название
```

В большинстве случаев достаточно указать только название компании, однако в зависимости от числа используемых линеек продуктов может потребоваться более точная классификация.

После логической группировки элементов с помощью записи group плагин сортирует элементы в алфавитно-цифровом порядке по записи order. Порядок может быть представлен любой символьной строкой. Можно просто использовать числа или выбрать более сложную схему, которая позволит добавлять другие элементы в будущем.

Единственный файл описания

К элементам, которые определены в единственном файле описания, можно применять неявное упорядочение. При отсутствии записи group это означает, что элементы автоматически наследуют значение, которое также доступно всем остальным целям файла описания. В этом случае нужно указать только запись order, как описано ранее.

Дополнительное управление меню

Каталоги, файлы и цели обеспечивают все механизмы, необходимые для заполнения меню содержанием, и даже дают некоторые возможности для упорядочения элементов меню. Для точной настройки общего вида меню и визуальной группировки его элементов можно также воспользоваться файлами управления форматом с именами .menu.

Формат меню задается как в стиле файла PxConfig, в котором каждый раздел определяет некоторую форму управления. Существуют следующие типы управления.

- размещение элементов обеспечивает дополнительное управление общей сортировкой элементов меню. Чтобы воспользоваться им, следует указать имя элемента в качестве названия раздела. Можно задать название раздела на других языках, включив в раздел запись, ключом которой является код языка, как описано ранее.

В простейшем случае раздел не содержит записей, однако можно воспользоваться записью type, чтобы уточнить, к какому типу элементов следует

применять шаблон. По умолчанию с шаблоном сравниваются все элементы, но сравнение можно ограничить определенными типами элементов, указав в качестве значения записи type сочетание одного или нескольких приведенных ниже символов:

- a — все типы (обработка по умолчанию);
- c — командные элементы (команды в формате командного интерпретатора, как описано выше);
- d — каталоговые элементы (подменю);
- e — исполняемые элементы (элементы, которые могут быть исполнены непосредственно без синтаксического разбора и использования стороннего приложения просмотра);
- f — файловые элементы (элементы, которые не могут быть исполнены непосредственно и требуют приложение для просмотра).

Например, приведенный далее код группирует сначала все элементы подменю, а затем все остальные элементы:

```
[*]
type = d
[*]
type = cef
```

- разделители позволяют создавать визуальные разграничения в меню. Название раздела должно начинаться с дефиса (-). Плагин игнорирует весь другой текст и все записи, которые имеются в разделе.

Например, приведенный далее код отделяет подменю от остальных элементов:

```
[*]
type = d
[-]
[*]
type = cef
```

Примечание. Плагин launchmenu.so не создает разделитель в начале и конце меню или рядом с другим разделителем.

- встроенные цели можно указывать в файле .menu так же, как и в файлах описания (см. разд. "Файлы описания запуска целей" ранее в этом разделе).

Вопросы и ответы

- как связать пиктограмму с подменю? Как указать альтернативный текст для элемента подменю, например, чтобы перевести его на другой язык?

Поскольку плагин launchmenu.so игнорирует файлы, имена которых начинаются с точки (.), первым шагом должно быть скрытие каталога путем добавления точки в начало его имени. Далее следует создать файл с расширением .tgt (имя файла не имеет значения, однако оно не должно начинаться с точки). В поле target необходимо задать полный путь к новому скрытому каталогу. Затем можно указать любую дополнительную информацию, например пиктограммы и переводы текста;

- можно ли использовать для создания меню файлы, которые находятся в других местах файловой системы?

Да, можно. Например, можно поместить в каталог $HOME/.ph/launchmenu символьную ссылку, которая указывает на любое место в файловой системе. Следует иметь в виду, что плагину launchmenu.so необходимо просканировать файлы и создать иерархию на основе их содержимого. Для этого может потребоваться некоторое время в зависимости от числа файлов и подкаталогов, которые обнаружит плагин;

- я отредактировал файл описания. Что нужно сделать для того, чтобы отобразить изменения в меню **Launch**?

Плагин launchmenu.so наблюдает только за изменениями каталогов, поскольку наблюдение за всеми файлами может занимать слишком много времени. Кроме того, каталоги обычно обновляются при установке и удалении элементов меню, поэтому плагин оперативно получает информацию о добавленных и удаленных элементах. Чтобы немедленно зарегистрировать изменение, которое внесено в файл, можно выполнить одно из следующих действий:

- перезапустить системную панель (ввести в командной строке команду shelf &);

  или:

- выполнить команду touch над каталогом, который содержит элемент меню, — плагин launchmenu.so обновит соответствующее подменю.

- будут ли отображены в меню пакеты, которые установлены с помощью старого установщика?

Утилита launchmenu_notify создает .tgt-файл, в котором представлены старые пакеты и пакеты третьих сторон.

- я установил пакет при помощи старого установщика, но элемент пакета не появился в меню. Что делать?

Можно попробовать выполнить следующие действия.

- запустить утилиту launchmenu_notify -vvv в командной строке. Она предоставит информацию о том, какие сторонние и существующие элементы имеются в меню, какие элементы нужно добавить и какие можно удалить;
- проверить наличие искомого элемента в списке существующих элементов меню. Если элемент отсутствует в меню **Launch**, то, вероятно, цель определяет недопустимый файл (например, несуществующий файл).

Если указанные действия не помогают разрешить проблему, пожалуйста, свяжитесь со службой технической поддержки.

- я создал собственный элемент меню, но он не отображается в меню **Launch.**

Возможно, цель определяет недопустимый или несуществующий файл. Плагин launchmenu.so не отображает элементы меню, которые не имеют цели, и элементы, цель которых не может быть распознана. Следует убедиться в том, что в качестве цели задан полный путь, исполняемый файл, и эту цель может обнаружить командный интерпретатор (чтобы определить это, следует воспользоваться утилитой which).

5.4. Настройка меню Desktop

Меню **Desktop** появляется на экране при щелчке правой кнопкой мыши в любом месте рабочего стола графической оболочки Photon.

Утилиту phmenu можно запустить с помощью ввода команды phmenu & в командной строке. Эта утилита позволяет помещать элементы меню в корзину (trash) и изменять их местоположение при помощи операции перетаскивания (drag-and-drop). Выбрав элемент меню, можно изменить его название, клавишу быстрого запуска и вызываемую команду. Чтобы добавить в меню новый элемент, следует выбрать его и перетащить в желаемое место в иерархической структуре меню.

Более подробные сведения о phmenu и pwm см. в документе "Описание программы" КПДА.10964-01 13.

5.5. Автоматический запуск приложений

Приложения могут запускаться автоматически вместе с графической оболочкой Photon. Для этого следует добавить имя исполняемого файла приложения в конфигурационный файл $HOME/.ph/phapps. Например:

```
ped &
pterm &
helpviewer &
```

Примечание. Если конфигурационный файл не существует, его необходимо создать и сделать исполняемым посредством изменения его свойств с помощью администратора файлов File Manager или посредством команды chmod +x ~/.ph/phapps.

5.6. Конфигурационные инструменты

Графическая оболочка Photon позволяет изменять ее настройки с помощью различных конфигурационных инструментов. Все конфигурационные инструменты можно запускать в командной строке, а некоторые — в системной панели и меню **Launch**.

- интерфейс: утилита pwmopts,элемент Appearance на системной панели или элемент меню **Launch→Configure→Appearance**.

Эти средства позволяют задавать цвет фона, узор и настройки изображения рабочего стола, а также выравнивание оконных заголовков и работу экранных окон, в том числе:

- отображение окна целиком или только его контура при перетаскивании;
- перемещение клавиатурного фокуса посредством щелчка мышью по окну или автоматически, следуя за указателем мыши;
- вывод окна на передний план при щелчке мышью по нему.

Чтобы задать цвет и узор рабочего стола, а также выбрать для него фоновое изображение, следует перейти на вкладку **Background**.

При изменении настроек pwmopts редактирует конфигурационный файл /usr/photon/config/wm/wm.cfg. Следующий пример иллюстрирует типовой файл wm.cfg:

```
[wm config]
fore_color = 0xD8D8D8
active_color = 0x5C8BDF
title_color = 0x65
inactive_color = 0xB1C1D9
base_color = 0xBDBDAA
border_active = 0
placement = 4
text_align = 2
auto_raise = 0
keyboard = 0
focus_cursor = 0
click_front = 1
drag = 1
[background]
vert_align = CENTER
horz_align = CENTER
image_op = PROPORTIONAL STRETCH
image = /usr/share/backdrops/1024x768/default.jpg
```

Если файл wm.cfg отсутствует, то различные цвета будут подменены. Например, вместо светло голубого цвета панели заголовка окна RGB 0x5C8BDF будет использоваться светло зеленый RGB 0x008070 (см. выше). Если отсутствует файл wframe_updated.so, то панель заголовка окна будет выглядеть в стиле Photon версии 1.x.

Для получения обычного внешнего вида необходимо убедиться в том, что файлы wframe_updated.so и wm.cfg корректно установлены на целевой системе.

- шрифты: утилита phfont.

Эта утилита позволяет изменять подстановку шрифтов, устанавливать такие параметры, как сглаживание, и конфигурирует определение азиатских шрифтов. Более подробные сведения см. в подразд. "Настройка шрифтов" раздела 9.

- графика: утилита phgrafx, элемент **Display** на системной панели или элемент меню **Launch→Configure→Display.**

Эти средства позволяют выбрать графические настройки оболочки Photon. Утилита phgrafx предоставляет список доступных графических режимов, из которого можно выбрать режим для каждого видеодрайвера, поддерживаемого графической платой компьютера. Этот список генерируется при сканировании оборудования, которое графическая оболочка Photon выполняет в процессе установки.

- локализация: утилита phlocale, элементы Localization или Time & Date на системной панели или элемент меню Launch→Configure→Localization.

Эти средства позволяют установить часовой пояс, язык, раскладку клавиатуры (см. подразд. "Языковые раскладки клавиатуры" раздела 4), системное время и дату.

Примечание. Изменение языка компьютера влияет только на приложения, которые поддерживают выбранный язык. Другие приложения продолжают использовать язык, установленный по умолчанию.

- мышь: утилита input-cfg, элемент Mouse на системной панели или элемент меню **Launch→Configure→Mouse.**

Эти средства позволяют задать скорость и ускорение указателя мыши. Можно также поменять местами кнопки мыши, чтобы уменьшить неудобство при работе с мышью левой рукой, а также включить колесо мыши при его наличии.

- сеть: утилита phlip, элемент Network на системной панели или элемент меню **Launch→Configure→Network.**

Это средства для управления настройками сети и модема.

- администратор печати: утилита prjobs.

Эта утилита выполняет просмотр, запуск и отмену заданий, которые находятся в очереди печати.

- удаленный доступ: утилита phrelaycfg.

Создает и удаляет файл /etc/system/config/noditto, который блокирует доступ к рабочему пространству графической оболочки Photon посредством утилиты pditto с любой удаленной машины.

- экранная заставка: утилита savercfg, элемент системной панели Screen Saver или элемент меню **Launch→Configure→Screen Saver.**

Эти утилиты конфигурируют экранную заставку графической оболочки Photon. Можно выбрать экранную заставку из списка, задать время ее активизации, пароль и любые допустимые командно-строковые параметры.

5.7. Просмотр файлов с помощью администратора файлов

В состав графической оболочки Photon входит администратор файлов (File Manager) — утилита pfm, которая позволяет просматривать каталоги и файлы при помощи графического интерфейса (рис. 5.4). Чтобы открыть администратор файлов, щелкните мышью по элементу File Manager группы Applications на системной панели или введите команду pfm & в командной строке.

Рис. 5.4. Администратор файлов графической оболочки Photon

Администратор файлов в графическом виде представляет файлы и папки. При двойном щелчке мышью по папке происходит открытие папки и отображение ее содержимого. При двойном щелчке по файлу выполняется его открытие в связанном с ним приложении (если связь существует). Администратор файлов также поддерживает операцию перетаскивания с помощью мыши. Например, можно переместить файл в папку. С помощью щелчка правой кнопкой мыши по файлу или папке отображается меню быстрого запуска команд, применимых к этому файлу или папке.

В верхней части окна администратора файлов расположены два текстовых поля, с помощью которых можно перемещаться по спискам каталогов и выполнять их фильтрацию. Чтобы непосредственно перейти в каталог, можно ввести его путевое имя в поле **Path**. Для просмотра файлов только определенного типа или файлов, начинающихся с определенного символа, следует воспользоваться полем **Filter**. Например, чтобы просмотреть файлы, которые начинаются с символа p, в поле **Filter** нужно ввести p*, а для просмотра файлов с расширением .ps следует ввести *.ps.

Меню администратора файлов графической оболочки Photon позволяют выполнять многие задачи управления файлами. Панель инструментов в верхней части окна администратора файлов включает в себя кнопки быстрого запуска

некоторых часто используемых команд. Более подробные сведения о pfm см. в руководстве по утилитам "Описание программы" КПДА.10964-01 13.

Чтобы просмотреть список наиболее часто используемых комбинаций клавиш для быстрого вызова, следует выбрать **Help→Quick Reference** в меню администратора файлов. Чтобы просмотреть все текущие закладки в панели, нужно щелкнуть по кнопке **Bookmarks** на панели инструментов.

Некоторые команды панели инструментов можно также запустить с помощью меню, вызываемого щелчком правой кнопкой мыши в администраторе файлов.

5.8. Обозреватель справки

Для отображения справочной документации в графической оболочке Photon используется обозреватель Helpviewer (рис.5.5). Документация хранится в каталоге $QNX_TARGET/usr/help/product.

Чтобы открыть обозреватель справки, следует щелкнуть по кнопке **Help** на системной панели или выбрать пункт **Help** в меню быстрого запуска на рабочем столе. Обозреватель справки также можно запустить с помощью ввода команды helpviewer & в командной строке.

Чтобы увидеть, какие подразделы содержатся в том или ином разделе, в списке **Topics** следует щелкнуть мышью по стрелке, изображенной рядом с разделом, а чтобы переместить раздел в начало списка, нужно дважды щелкнуть по нему мышью. При щелчке мышью по названию раздела его содержимое отображается в панели просмотра.

К другим разделам можно также переходить с помощью гипертекстовых ссылок внутри текущего раздела. Ссылки выделяются цветом и подчеркиванием.

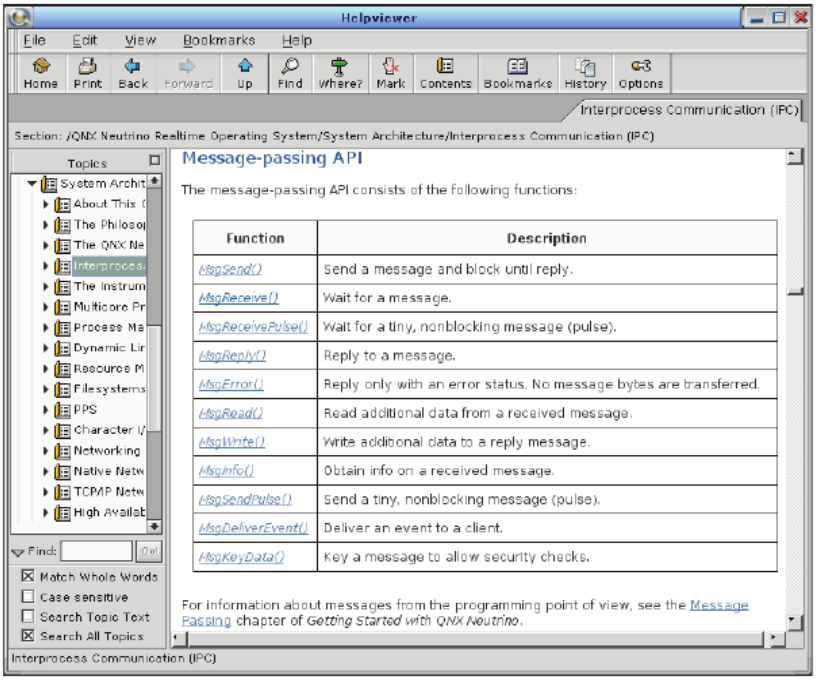


Рис. 5.5. Обозреватель справки графической оболочки Photon

Поиск разделов и ключевых слов

Панель **Find** позволяет выполнять поиск слов в справочных файлах. Она расположена под списком **Topics**. Если список **Topics** не отображен, следует выбрать пункт меню View→Topics или нажать комбинацию клавиш <Ctrl>+<T>. Затем необходимо ввести одно или несколько слов в поле **Find** и нажать кнопку **Go!**. Если ввести несколько слов, поиск будет выполняться по тем разделам, которые содержат все заданные слова.

Примечание. При необходимости вы можете сгенерировать индекс полнотекстового поиска по файлам справки, если он отсутствует. Для этого нужно выбрать File→Generate Index. Если справка имеет большой объем, операция генерации индекса может занять несколько минут.

Можно уточнить поиск с помощью одного или нескольких параметров:

- **Match Whole Words** — флажок для поиска по целым словам. Если он не установлен, то обнаруживаются слова, которые частично совпадают с искомым (например, для слова "grep" будет найдено слово "egrep");
- **Case Sensitive** — флажок для поиска с учетом регистра;

- **Search Topic Text** — флажок для поиска во всем тексте раздела. Если он не установлен, поиск выполняется только в заголовках разделов;
- **Search All Topics** — флажок для поиска во всех разделах справки.

Терминальное окно pterm позволяет выбирать (выделять) часть текста и затем вызывать обозреватель справки одним из следующих способов:

- щелчком правой кнопки мыши вызвать меню утилиты pterm и выбрать элемент Search help;

или:

- нажатием комбинации клавиш <Ctrl>+<Alt>+<H>.

Обозреватель справки запускается и выполняет поиск разделов, которые содержат выделенный текст, в оглавлении справки. Первый найденный раздел автоматически отображается на экране.

Можно также просто ввести с клавиатуры слова для поиска в окне утилиты pterm и затем нажать комбинацию клавиш <Ctrl>+<Alt>+<H>.

Примечание. Большинство документов ЗОСРВ «Нейтрино» включает в себя индекс ключевых слов, который также помогает при поиске. При использовании онлайновой документации для этого следует щелкнуть мышью по кнопке индекса ключевых слов, которая находится в начале и в конце каждого файла .

Создание закладок для разделов

Если какой-либо раздел понадобится еще в будущем, на нем можно создать закладку. При создании закладки ссылка для быстрого перехода к соответствующему разделу сохраняется в списке закладок. Чтобы сделать закладку на текущем разделе, следует воспользоваться командой **Bookmarks→Add Bookmarks.**

Для просмотра списка закладок нужно нажать кнопку **Bookmarks** на панели инструментов. Чтобы просмотреть раздел, на который сделана закладка, следует щелкнуть мышью по элементу списка. Закладки также отображаются в меню **Bookmarks.**

Чтобы удалить закладку, нужно сначала открыть раздел, на который она сделана, и затем выбрать **Bookmarks→Remove Bookmark**.

Навигация в файлах справки

Для навигации по обозревателю справки можно использовать способы, представленные в табл. 5.1.

Таблица 5.1

Действие	Команда меню	Комбинация клавиш для быстрого запуска
Перейти к самому верхнему разделу справки	**File --> Home**	<Ctrl>+<H>
Перейти к предыдущему просмотренному разделу	**File --> Back**	<Alt>+< ☐>
Вернуться к следующему разделу (после применения команды **File | Back**)	**File --> Forward**	<Alt>+< ☐>
Перейти на один уровень вверх, если открыта папка	**File --> Up**	<Ctrl>+<U>
Открыть панель разделов, если она закрыта	**View --> View Topics**	<Ctrl>+<T>
Открыть панель результатов поиска, если она закрыта	**View --> View Search Results**	<Ctrl>+<S>
Просмотреть расположение текущего раздела в списке разделов	**View --> Where?**	
Просмотреть список просмотренных ранее разделов	**View --> History List**	<Ctrl>+<Y>

Онлайновая документация также имеет несколько кнопок навигации, которые расположены в начале и в конце каждого файла (рис. 5.6).

Рис. 5.6. Кнопки навигации в онлайновой документации

При нажатии кнопки **Contents** происходит перемещение "вверх", а именно:

- в руководстве, как правило, выполняется переход к разделу "About This Guide";

- в справочнике выполняется переход к списку элементов, которые начинаются с заданной буквы. Например, при просмотре описания функции abs() нажатие кнопки **Contents** приводит к переходу к списку функций, начинающихся с буквы A.

Одновременный просмотр нескольких разделов

Пользователь может просматривать несколько разделов одновременно, открывая их в новых окнах. Каждому открытому окну соответствует вкладка над панелью разделов.

Чтобы открыть новую панель разделов, следует воспользоваться командой **File→New Section** или нажать комбинацию клавиш <Ctrl>+<N>. Для просмотра какого-либо из открытых разделов нужно щелкнуть по его вкладке. Чтобы закрыть текущий раздел, следует выбрать **File→Close Section** или нажать комбинацию клавиш <Ctrl>+<D>.

5.9. Просмотр Web-страниц

Графическая оболочка Photon включает в себя Web-браузер, который служит для просмотра локальных HTML-файлов и страниц в Интернете. Для его запуска необходимо нажать соответствующую кнопку в группе **Internet** системной панели Photon.

Примечание. В состав операционной системы ЗОСРВ «Нейтрино» также входит Web-сервер Slinger, который обеспечивает встраиваемые системы доступом к Интернету. Более подробные сведения см. в разделе 16.

5.10. Соединение с другими системами

Компьютеры, которые используют графическую оболочку Photon, могут быть соединены между собой следующими методами:

- с помощью резидентной утилиты phditto, которая позволяет просматривать другое рабочее пространство графической оболочки Photon, находящееся в сети, и взаимодействовать с ним;

- с помощью инструмента связи Phindows, который обеспечивает соединение платформы Microsoft Windows с графической оболочкой Photon и ее приложениями, находящимися на удаленном компьютере с операционной системой ЗОСРВ «Нейтрино».

Утилита phditto

Утилита phditto позволяет установить соединение с сеансом графической оболочки Photon, который выполняется на другом компьютере. Можно как подключиться к существующему сеансу, так и запустить новый. Утилита phditto дает возможность взаимодействовать с удаленным сеансом графической оболочки Photon так, как будто он проходит локально. Для того чтобы получить доступ к удаленному узлу с помощью утилиты phditto, необходимо запустить на нем утилиту phrelay.

Для завершения сеанса утилиты phditto следует выбрать команду Close в окне меню. Для вызова этого меню следует щелкнуть правой кнопкой мыши по метке phditto в панели задач.

Инструмент Phindows

Утилита Phindows ("Photon in Windows") является инструментом связи, который позволяет использовать платформы Windows для подключения к приложениям графической оболочки Photon, выполняющимся на удаленном компьютере с операционной системой ЗОСРВ «Нейтрино», и для взаимодействия с ними.

Конфигурирование стека протоколов TCP/IP в ЗОСРВ «Нейтрино»

При использовании стека протоколов TCP/IP необходимо убедиться в корректности его конфигурации до запуска утилиты Phindows:

- стек протоколов TCP/IP должен быть установлен и запущен на удаленном хосте с операционной системой ЗОСРВ «Нейтрино»;
- на удаленном хосте также должна выполняться утилита inetd, а в конфигурационные файлы TCP/IP необходимо добавить следующие элементы:
- в файл /etc/inetd.conf требуется добавить строку:

```
phrelay stream tcp nowait root /usr/bin/phrelay phrelay
```

- в файл /etc/services требуется добавить строку:

```
phrelay 4868/tcp
```

Примечание. Приведенные выше строки уже включены в конфигурационные файлы, но они закомментированы. Чтобы добавить их, достаточно удалить знак номера (#).

Эти две записи указывают утилите inetd наблюдать за входящими запросами на создание нового сеанса графической оболочки Photon. При обнаружении запроса (в данном случае от удаленного Phindows-клиента) утилита inetd автоматически устанавливает полное TCP/IP-соединение и запускает для его обслуживания утилиту phrelay. После этого утилита Phindows полностью подключена к локальному компьютеру.

Более подробные сведения об утилите inetd см. в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

Запуск утилиты Phindows

Чтобы запустить утилиту Phindows на компьютере с операционной системой Windows, следует выполнить одно из следующих действий:

- щелкнуть мышью по пиктограмме Phindows, если эта пиктограмма была создана;
- выбрать в меню команду Start→Programs→QNX Momentics→Phindows;
- запустить программу C:\Program Files\phindows\phindows.exe.

Утилита Phindows отображает диалоговое окно **Connect**, в котором можно задать тип соединения (TCP/IP или последовательное). Для соединения можно применять различные параметры, однако в большинстве случаев настройки по умолчанию вполне достаточны.

При запросе TCP/IP-соединения требуется также задать интернет-адрес компьютера с операционной системой ЗОСРВ «Нейтрино», к которому осуществляется подключение (например, 198.53.31.1). Если удаленный компьютер конфигурирован корректно, то соединение с ним устанавливается успешно, а

графическая оболочка Photon запускается и на экране появляется ее приглашение для входа в систему.

При запросе последовательного соединения необходимо указать COM-порт (например, COM1 или COM2). Если пользователь не задает скорость передачи, утилита Phindows использует текущие значения по умолчанию операционной системы Windows. При последовательном соединении утилита Phindows изначально выполняет функции простого текстового терминала, который позволяет вводить команды непосредственно в модем (например, ATDT1-613-591-0934). После создания соединения следует войти в операционную систему ЗОСРВ «Нейтрино» и ввести следующую команду:

```
/usr/bin/phrelay
```

Эта команда переключает утилиту Phindows из режима текстового терминала в режим графического терминала Photon. После этого отображается экран для входа в систему графической оболочки Photon.

Дополнительные настройки

Командно-строковые параметры утилиты Phindows позволяют:

- задавать параметры сжатия и кэширования данных;
- устанавливать соединение с удаленным сеансом графической оболочки Photon;
- использовать нестандартную цветовую палитру;
- отображать один сеанс графической оболочки Photon в нескольких экранах;
- работать с сеансом графической оболочки Photon совместно с другими пользователями;
- создавать ярлыки приложений Photon на рабочем столе операционной системы Windows.

5.11. Комбинации клавиш быстрого запуска

Многочисленные комбинации клавиш быстрого запуска позволяют выполнять задачи быстро и легко. В табл. 5.2—5.6 приведены комбинации клавиш быстрого запуска для вызова функций утилиты pterm, редактирования текстовых полей в Photon-приложениях, управления окнами, использования рабочего пространства графической оболочки Photon и других действий.

Утилита pterm

Эмулятор терминала графической оболочки Photon называется pterm. Он работает подобно драйверу символьного устройства (см. разд. "Общие сведения о клавиатуре" раздела 4).

Примечание. При вводе текста в режиме замены нажатие клавиши <Enter> переключает утилиту pterm в режим вставки.

Утилита pterm также поддерживает комбинации клавиш быстрого запуска (табл. 5.2).

Таблица 5.2

Действие	**Клавиши быстрого запуска**
Скопировать выделенный текст в буфер обмена	<Ctrl>+<Alt>+<X> или <Ctrl>+<Alt>+<C>
Вставить выделенный текст из буфера обмена	<Ctrl>+<Alt>+<V> или <Ctrl>+правая кнопка мыши
Обратить выделение текста	<Ctrl>+<Alt>+<R>
Выполнить поиск выделенного текста в справке	<Ctrl>+<Alt>+<H>
Задать настройки утилиты pterm	<Ctrl>+<Alt>+<O>
Выполнить прокрутку строк, находящихся в буфере	<Ctrl>+<Alt>+<□>, <Ctrl>+<Alt>+<□>, <Ctrl>+<Alt>+<Page Up>, <Ctrl>+<Alt>+<Page Down>, <Ctrl>+<Alt>+<Home> и <Ctrl>+<Alt>+<End>
Увеличить или уменьшить размер шрифта и окна	<Ctrl>+<Alt>+<[> и <Ctrl>+<Alt>+<]>
Увеличить или уменьшить только размер шрифта	<Ctrl>+<Alt>+<,> и <Ctrl>+<Alt>+<.>

Текстовое поле

Комбинации клавиш быстрого запуска действий для работы с текстом представлены в табл. 5.3.

Таблица 5.3

Действие	**Клавиши быстрого запуска**
Вырезать выделенный текст	<Ctrl>+<X> или <Ctrl>+<Alt>+<X>
Скопировать выделенный текст в буфер обмена	<Ctrl>+<C> или <Ctrl>+<Alt>+<C>
Вставить выделенный текст из буфера обмена	<Ctrl>+<V>, <Ctrl>+<Alt>+<V> или <Ctrl>+правая кнопка мыши

Окно

Оконный администратор pwm позволяет использовать следующие комбинации клавиш быстрого запуска (табл. 5.4).

Таблица 5.4

Действие	**Клавиши быстрого запуска**
Переместить окно на передний план	<Alt>+<F2>
Переместить окно на задний план	<Alt>+<F3>
Закрыть окно	<Alt>+<F4> или двойной щелчок мышью по кнопке оконного меню
Восстановить размеры окна, которые оно имело до развертывания	<Alt>+<F5> или двойной щелчок мышью по строке заголовка
Переместить окно	<Alt>+<F7>
Изменить размер окна (для задания нового размера следует использовать мышь или клавиши управления курсором)	<Alt>+<F8>
Свернуть окно	<Alt>+<F9>
Развернуть окно	<Alt>+<F10> или двойной щелчок мышью по строке заголовка

Рабочее пространство

Оконный администратор pwm позволяет использовать следующие комбинации клавиш быстрого запуска (табл. 5.5).

Таблица 5.5

Действие	**Клавиши быстрого запуска**
Переместить самое нижнее окно на передний план	<Alt>+<Esc>
Переместиться на следующее окно	<Alt>+<Shift>+<Esc>
Переместиться на следующую или предыдущую консоль	<Ctrl>+<Alt>+<Enter> или <Ctrl>+<Alt>+<Backspace>
Перейти на консоль n, где n — число от 1 до 9	<Ctrl>+<Alt>+<n>
Отобразить меню **Desktop**	<Alt>+<Enter>

При перемещении между виртуальными консолями графическая оболочка Photon пропускает пустые консоли.

Процесс wmswitch, который автоматически запускается графической оболочкой Photon, позволяет использовать следующие комбинации клавиш быстрого запуска (табл. 5.6).

Таблица 5.6

Действие	**Клавиши быстрого запуска**
Переместиться в следующее приложение	<Alt>+<Tab>
Переместиться в предыдущее приложение	<Alt>+<Shift>+<Tab>

Выход из графической оболочки Photon

Чтобы выйти из графической оболочки Photon, можно воспользоваться комбинацией клавиш <Ctrl>+<Alt>+<Shift>+<Backspace>.

Внимание! Перед тем как ввести эту команду, необходимо убедиться в том, что на компьютере не выполняются какие-либо приложения или утилиты. В противном случае некоторые файлы могут остаться открытыми. Кроме того, перезагрузка в процессе критически важного обновления может вызвать необходимость исправления файловой системы.

Чтобы отключить этот метод выхода из графической оболочки Photon, следует передать параметр -b драйверу устройств ввода системы. Более подробные сведения см. в описании devi-* в справочнике по утилитам.

5.12. Переменные окружения графической оболочки Photon

Переменные окружения определяют параметры и поведение системы. Для задания переменных окружения, которые конфигурируют графическую оболочку Photon, можно воспользоваться командной строкой, однако команда, которую необходимо ввести, зависит от используемого командного интерпретатора. В командных интерпретаторах ksh и esh применяется команда export.

Далее приведен список переменных окружения, специфичных для графической оболочки Photon.

- **ABLANG** — языковой код (например, en_CA, который соответствует канадскому варианту английского языка), задающий язык, на котором многоязычное Photon-приложение отображает информацию.;

Более подробные сведения см. в разд. "International Language Support" руководства программиста Photon "Programmer’s Guide". Языковые коды, которые поддерживаются в настоящее время, перечислены в файле /usr/photon/appbuilder/languages.def.

- **ABLPATH** — список каталогов, в которых многоязычное Photon-приложение выполняет поиск переводных файлов;

Более подробные сведения см. в подразделе ph руководства "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01;

- **AB_RESOVR** — путевая переменная, в которой перечислены каталоги для поиска записей ресурсов приложений, созданных с помощью построителя приложений PhAB;

- **AUTOCONNECT** — для того чтобы запустить сценарий /etc/autoconnect, необходимо установить значение этой переменной окружения равным 1. Более подробные сведения см. в описании /etc/autoconnect в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01;

- **DISPLAY** — имя физического дисплея, на котором выполняется рисование;

- **IVE_HOME** — может использоваться виртуальной машиной Java VM;

- **J9PLUGIN_ARGS** — аргументы, которые передаются подключаемым Java-модулям браузера;

- **PHEXIT_DISABLE** — установка этой переменной окружения отключает кнопку **Exit** в диалоговом окне входа графической оболочки Photon, что не позволяет пользователям выходить в текстовый режим. Более подробные сведения см. в описании phlogin2 и phlogin в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01;

- **PHFONT** — зарегистрированное имя сервера шрифтов (например, /dev/phfont). Более подробные сведения см. в описании ph в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01;

- **PHFONT_USE_EXTERNAL** — если эта переменная окружения существует, утилита io-graphics запускает администратора шрифтов как отдельный процесс (см. раздел phfont), а не использует библиотеку phfont.so». Переменную окружения PHFONT_USE_EXTERNAL следует создавать в системах, в которых доступ к службам шрифтов инструментального компьютера осуществляется удаленными клиентами (например, phindows и phditto);

- **PHFONTMEM** — размер области общедоступной памяти, которая выделяется задаче и графической оболочке Photon для возврата текстовых растров (обычно требуется только графическим драйверам);

- **PHFONTOPTS** — параметры, которые передаются серверу шрифтов графической оболочки Photon. Более подробные сведения см. в описании phfont;

- **PHGFX** — полная команда, которую сценарий ph выполняет вместо команд по умолчанию для запуска графического драйвера;

- **PHINPUT** — полная команда, которую сценарий ph выполняет вместо команд по умолчанию для запуска драйвера устройств ввода;

- **PHINSTANCE** — число экземпляров графической оболочки Photon. Более подробные сведения см. в описании phlogin2 и phlogin в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01;

- **PHOTON** — имя устройства графической оболочки Photon (как правило, /dev/photon). Более подробные сведения см. в описании ph в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01;

- **PHOTONOPTS** — дополнительные параметры, которые передаются серверу графической оболочки Photon при его запуске (только в версии Windows-hosted);
- **PHOTON_PATH** — имя корневого каталога, который содержит файлы графической оболочки Photon (обычно /usr/photon). Более подробные сведения см. в описании ph в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01;
- **PHWM** — название оконного диспетчера, который запускается вместе с графической оболочкой Photon. Более подробные сведения см. в описании ph в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01;
- **PHWMEXIT** — установка этой переменной окружения отключает диалоговое окно подтверждения при выходе из графической оболочки Photon. Более подробные сведения см. в описании pwm в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01;
- **PHWMOPTS** — параметры, которые передаются оконному диспетчеру при его запуске. Более подробные сведения см. в описании pwm в руководстве "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01;
- **PTERMPAL** — путевое имя файла палитры утилиты pterm;
- **PTERMRC** — имя локального конфигурационного файла утилиты pterm.
- **PWMOPTS** — параметры, которые передаются оконному администратору при его запуске (только в версии Windows-hosted). Более подробные сведения см. в описании pwm в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01;
- **PWM_PRINTSCRN_APP** — приложение, которое запускается при нажатии клавиши <Print Screen>. По умолчанию запускается приложение snapshot.

5.13. Устранение неполадок

Далее рассмотрены некоторые проблемы и вопросы, соотносящиеся к графической оболочке Photon.

- как изменить цветовую схему в графической оболочке Photon?

В графической оболочке Photon можно изменить цвет строки заголовка для любого состояния окна. Чтобы сделать это, нужно воспользоваться командой **Launch→Configure→Appearance**, а затем выбрать вкладку **Window**. На этой вкладке можно выбрать стандартную цветовую схему из списка или задать цвет для каждого состояния окна (активного и неактивного) и цвет оконного заголовка в отдельности.

- я задал псевдоним в файле .profile, но он не установился в терминалах графической оболочки Photon.

Командный интерпретатор не экспортирует псевдонимы. В графической оболочке Photon терминал pterm по умолчанию не запускается как начальный командный интерпретатор, а следовательно, не считывает конфигурационные файлы /etc/profile и ~/.profile.

Если псевдоним необходимо определить во всех командных интерпретаторах (внутри и вне терминала графической оболочки Photon), следует задать его в файле запуска командного интерпретатора. Более подробные сведения см. в подразд. "Файл запуска ksh" раздела 9.

В качестве альтернативы можно передать параметр –l утилите pterm, чтобы она запустилась как начальный командный интерпретатор и считала файл .profile. При желании можно изменить элемент Terminal на системной панели так, чтобы он запускал команду pterm -l. Для этого следует щелкнуть правой кнопкой мыши по системной панели и выбрать пункт Setup. Далее необходимо выбрать элемент Terminal и заменить команду pterm на pterm -l.

Следует также соответствующим образом изменить всплывающее меню **Desktop**. Для этого необходимо отредактировать файл $HOME/.ph/wm/wm.menu или запустить утилиту phmenu и добавить параметр -l в команду pterm.

- можно ли пропустить приглашение для входа в систему при загрузке компьютера в графический режим?

Да. Более подробные сведения см. в подразд. "Файл rc.local" раздела 8.

- как изменить языковую раскладку клавиатуры?

Следует воспользоваться элементом **Localization** на системной панели. Можно сделать выбор из нескольких различных конфигураций клавиатуры; см. подразд. "Языковые раскладки клавиатуры" раздела 4.

- как добавлять в обозреватель справки (Helpviewer) файлы, например справочные файлы устанавливаемых программ и новые документы, которые найдены в сети? Меню **File** не позволяет выполнить запрос файлов и осуществить поиск справочного файла.

Обозреватель справки ищет файлы с расширением .toc в каталоге /usr/help/product. Чтобы открыть какой-либо файл, не создавая .toc-файлы, следует воспользоваться браузером Voyager вместо обозревателя справки.

- я попытался создать новые файловые связи с помощью утилиты pfm, но они не работают корректно. Например, я попробовал связать .txt-файлы с программой ped, используя существующие связи, но приложение ped не запускается.

Следует убедиться в том, что путь /usr/photon/bin присутствует в переменной окружения PATH, а затем выполнить следующие действия:

- запустить утилиту pfm;
- нажать клавишу <F11> или выбрать элемент **Associations** в меню **Edit**;
- добавить новую файловую связь, щелкнув мышью по кнопке **Add**;
- ввести следующие настройки:

```
Pattern: *.txt
Open: ped
View: ped
Edit: ped
```

- закрыть диалоговое окно **New Association Type**, нажав кнопку **Done**;
- закрыть диалоговое окно **Associate**, нажав кнопку **Done.**

- как отключить комбинации клавиш <Ctrl>+<Alt>+<1>, <2>, ..., которые переключают консоли в графической оболочке Photon?

Следует поместить в файл /etc/rc.d/rc.local следующую строку:

export PHWMOPTS="-S"

Более подробные сведения см. в описании pwm в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

- как завершить работу графической оболочки Photon, если мышь не подключена к компьютеру?

Можно нажать комбинацию клавиш <Ctrl>+<Alt>+<Shift>+<Backspace>. Если это не приводит к завершению работы графической оболочки Photon, компьютер, возможно, заблокирован. В этом случае можно нажать кнопку перезагрузки компьютера. Чтобы не прибегать к использованию кнопки перезагрузки, следует запустить утилиту inetd, затем запустить в окне утилиту telnet и применить команду slay к процессам, используемым графической оболочкой Photon.

- как изменить цвет текста и фона для терминала?

Утилита pterm имеет параметр -K, который позволяет выбирать начальные цвета.

Например, команда pterm -K 17 задает голубой цвет текста (1) и светло-серый цвет фона (7). Можно также задать точные RGB-значения для всех 16 цветов, которые использует утилита pterm, создав файл палитры. Более подробные сведения см. в описании pterm в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

- при установке нового языкового параметра в элементе **Localization** системной панели ничего не изменяется. Почему?

Этот параметр устанавливает значение переменной окружения **ABLANG**, с помощью которой некоторые приложения определяют, какой язык следует использовать. Некоторые приложения могут не поддерживать язык, который установил пользователь. Обычно изменение языка не влияет на работающие приложения и действует только на новые приложения.

- как отключить системную панель?

Для того, что бы закрыть системную панель в текущей сессии Photon необходимо выполнить команду shelf -e.

Более постоянным решением является присвоение переменной окружения PHSHELF_DISABLE значения 1. Это может быть задано в файле .profile пользователя с помощью команды: export PHSHELF_DISABLE=1.

Для получения более подробных сведений об утилите shelf см. документ «Описание программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

6. Работа с файлами

Примечание. Этот раздел посвящен работе с файловой системой типа «QNX4», используемой в ЗОСРВ «Нейтрино» и по умолчанию совместимой с файловой системой операционной системы QNX 4.

6.1. Файлом является все

В операционной системе файлами является практически все. Устройства, данные и даже службы обычно представлены в виде файлов. Это позволяет легко работать с локальными и удаленными ресурсами непосредственно из командной строки или с помощью любой программы, которая оперирует файлами.

Типы файлов

ЗОСРВ «Нейтрино» поддерживает следующие типы файлов (приведенный в скобках значок используется командой ls -l для идентификации соответствующего типа):

- обычный (-) — файл, содержащий данные пользователя, такие как код C, HTML и данные. Например, /home/fred/myprog.c;
- каталог (d) — по сути, это нечто, содержащее файлы и другие каталоги. Например, /home/fred. Каталог выполнен в виде файла на диске, в котором хранится список имен файлов и других каталогов. Имя файла может быть ассоциировано с записью inode. Более подробные сведения см. в подразд. "Файловая система QNX 4" раздела 11;
- символьная ссылка (l) — дополнительное имя файла или каталога. Например, /usr/bin/more является символьной ссылкой на /usr/bin/less. Более подробные сведения см. в разд. "Символьные ссылки" раздела 11;
- специальный именованный (n) – область разделяемой памяти, например /dev/shmem/Pg101e0001;
- символьно-ориентированные специальные файлы (c) — записи, представляющие символьное устройство. Например, /dev/ser1 представляет последовательный порт;

- специальные файлы FIFO (p) — постоянные именованные программные каналы, через которые две программы взаимодействуют, например, PipeA;
- блочно-ориентированные специальные файлы (b) — записи, представляющие блочное устройство, например, диск. К примеру, /dev/hd0 представляет "сырые" блочные данные на основном диске;
- сокеты (s) — записи, представляющие коммуникационные сокеты, в частности сокет домена UNIX.

Некоторые файлы являются постоянными и не удаляются при перезагрузке системы — таково большинство файлов в файловой системе диска. Другие файлы могут существовать только до тех пор, пока выполняется отвечающая за них программа. В качестве примеров таких файлов можно привести объекты разделяемой памяти, объекты в файловой системе /proc и временные файлы на диске, которые продолжают использоваться, даже если ссылки на эти файлы (их файловые имена) уже удалены.

6.2. Имена файлов и путевые имена

Для обращения к какому-либо файлу или каталогу вам необходимо указать путевое имя (pathname) — символьное имя, которое указывает программе, где искать файл в иерархической структуре каталогов, основанной на корневом каталоге (/).

Типичное путевое имя в ЗОСРВ «Нейтрино» выглядит следующим образом: /home/fred/.profile (рис. 6.1). В этом примере файл .profile находится в каталоге fred, который, в свою очередь, располагается в каталоге home в / (корневом каталоге).

Рис. 6.1. Типичное путевое имя

Как и в системе Linux и других UNIX-подобных операционных системах, в ЗОСРВ «Нейтрино» компоненты путевого имени разделяются косой чертой (/). В отличие от этих ОС, в операционных системах Microsoft используется обратная косая черта (\).

Примечание. Для просмотра файлов и каталогов в вашей системе используйте утилиту ls. Она является эквивалентом команды dir в системе MS-DOS. Более подробные сведения см. в подразд. "Основные команды" раздела 4 или в описании утилиты ls в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

Абсолютные и относительные путевые имена

В ЗОСРВ «Нейтрино» различаются два типа путевых имен.

- абсолютный маршрут доступа к файлу. Путевые имена, начинающиеся с косой черты, указывают местонахождение конкретного файла по отношению к корневому каталогу (/). Например, /usr/lib/libmalloc.so.2;
- относительный маршрут доступа к файлу. Путевые имена, начинающиеся не с косой черты (/), обозначают местонахождение файла по отношению к текущему рабочему каталогу. Например, если ваш текущий каталог /home/fred, то относительный маршрут доступа к файлу .ph/helpviewer будет соответствовать абсолютному маршруту /home/fred/.ph/helpviewer.

Путевое имя /home/fred/.ph/helpviewer на самом деле задает каталог, а не обычный файл. По виду путевого имени нельзя определить, указывает ли оно на обычный файл, каталог, символьную ссылку или файл какого-либо иного типа. Для определения типа файла используйте команды file или ls -ld.

Исключением является путевое имя, которое заканчивается косой чертой (/), оно всегда указывает на каталог. При использовании опции -F в утилите ls эта утилита покажет косую черту в конце имени каталога.

Каталоги "точка" и "две точки"

Каждый каталог в файловой системе QNX 4 содержит следующие специальные ссылки:

- . (точка) – текущий каталог;
- .. (две точки) – каталог, в котором появляется текущий каталог.

Например, чтобы вывести список каталога, находящегося над текущим каталогом, наберите команду:

```
ls ..
```

Если текущий каталог /home/fred/.ph/helpviewer, вы можете вывести список корневого каталога с помощью команды:

```
ls ../../../..
```

Однако абсолютный путь (/) значительно короче, и вам не нужно гадать, сколько добавлять "двойных точек".

Примечание. Файловые системы флэш-памяти не поддерживают символы . и .., однако командный интерпретатор может распознать их перед тем, как передать путь в файловую систему. Вы также можете установить жесткие ссылки на эти имена в файловой системе флэш-памяти.

Примечание о команде cd

В некоторых традиционных системах UNIX команда cd (перейти в другой каталог) модифицирует заданное имя пути, если оно содержит символьные ссылки. В результате путевое имя нового текущего рабочего каталога (которое вы можете

отобразить с помощью команды pwd) может отличаться от того, которое было передано команде cd.

Однако в ЗОСРВ «Нейтрино» команда cd не модифицирует имя пути (за исключением случаев использования двух точек (..)). Например, при выполнении следующей команды:

```
cd /home/dan/test/../doc
```

текущим рабочим каталогом станет каталог /home/dan/doc, даже если какие-либо элементы в имени пути были символьными ссылками.

Отсутствие буквенных обозначений для дисков

В отличие от ОС Microsoft Windows, в которой диски обозначаются буквами, стоящими впереди путевого имени (например, C:\), в ЗОСРВ «Нейтрино» диски обозначаются как обычные каталоги в пространстве путевых имен. Каталоги, которые связаны с другой файловой системой (например, в другом разделе жесткого диска), называются точками монтирования (mountpoints).

Обычно файловая система, расположенная на основном диске, монтируется в каталоге / (корневом каталоге пространства путевых имен). Полная установка ЗОСРВ «Нейтрино» (например, при резидентной установке комплекта разработчика QNX Momentics) автоматически монтирует все дополнительные дисковые файловые системы в каталоге /fs. Пример представлен на рис. 6.2.

Таким образом, если в системе на основе DOS для обращения ко второму разделу диска нужно набрать D:\, в системе ЗОСРВ «Нейтрино» вы можете обратиться ко второму разделу файловой системы QNX 4 на первом жестком диске следующим образом: /fs/hd0-qnx4-2.

Рис. 6.2. Точка монтирования — каталог /fs

Более подробные сведения о структуре типичного пространства путевых имен в ЗОСРВ «Нейтрино» см. в разд. "Где все хранится?" далее в этом разделе. Более подробные сведения о монтировании файловых систем см. в разделах 11 и 8.

Путевые имена, начинающиеся с точки

Когда вы выводите список каталога, утилита ls обычно не показывает файлы и каталоги, имена которых начинаются с точки. Программы ставят перед именами конфигурационных файлов и каталогов точку, чтобы скрыть их. Поэтому такие файлы (как легко догадаться) называются скрытыми (hidden).

Кроме особого отношения к скрытым файлам со стороны утилиты ls и некоторых других программ (таких как администратор файлов pfm оболочки Photon), скрытые файлы больше ничем не отличаются от любых других. Для того чтобы отобразить список всех файлов, включая скрытые, используйте команду ls -a.

Расширения

Расширение имени файла (т.е. что-то в конце имени файла) указывает программам и пользователю, какой тип данных содержится в файле. В файловой системе QNX 4 (собственной дисковой файловой системе ЗОСРВ «Нейтрино») расширения просто являются обычной частью имени файла и могут иметь любую длину, учитывая однако, что общий размер имени файла не должен превышать 505 байт.

В большинстве случаев расширения файловых имен являются просто условными наименованиями, однако некоторые утилиты работают по-разному в зависимости от расширения. Список наиболее часто используемых в системе ЗОСРВ «Нейтрино» расширений приведен в подразд. "Расширения файловых имен" далее в этом разделе.

Отображение пространства путевых имен

Вы могли заметить, что мы говорим о файлах и каталогах так, будто они, скорее, "появляются" в своих родительских каталогах, чем находятся в них. Это связано с тем, что в ЗОСРВ «Нейтрино» пространство имен файлов является виртуальным и определяется не только файловой системой, которая находится на носителе, смонтированном в корневой каталог, но и путями и псевдонимами путевых имен, зарегистрированными администратором процессов.

Рассмотрим, например, небольшую часть пространства путевых имен (рис. 6.3).

Рис. 6.3. Фрагмент пространства путевых имен

В типичной дисковой системе на основе ЗОСРВ «Нейтрино» каталог / отображается в корневом каталоге файловой системы на разделе физического жесткого диска. Эта файловая система на диске в действительности не содержит каталог /dev — он существует виртуально и регистрируется администратором процессов. В свою очередь имя файла ser1 также не существует в дисковой файловой системе — оно было назначено драйвером последовательного порта.

Эта особенность позволяет создавать виртуальные объединения каталогов (directory union). Это происходит, когда несколько администраторов ресурсов получают файлы, находящиеся в общем каталоге в пространстве путевых имен.

Примечание. Чтобы система была более удобной в обслуживании, рекомендуется создавать как можно меньше объединений каталогов.

Более подробная информация об организации пространства путевых имен приведена в разд. "Управление именами путей" главы "Администратор процессов" руководства "Описание применения. Часть 1. Системная архитектура".

Правила образования имен файлов

ЗОСРВ «Нейтрино» поддерживает ряд файловых систем, обладающих различными возможностями и имеющих различные правила для формирования допустимых файловых имен. Более подробные сведения о файловых системах можно найти в разделе 11, а в разделе 21 — сведения об ограничениях системы.

Файловая система QNX 4 — это обычная дисковая файловая система, которую использует ЗОСРВ «Нейтрино». В этой файловой системе имена файлов могут иметь длину не более 48 байт, однако вы можете расширить их до 505 байт (см. подразд. "Имена файлов" раздела 11). Отдельные байты в имени файла могут иметь любое значение, кроме следующих (все значения — в шестнадцатеричной форме):

- от 0x00 до 0x1F (все управляющие символы);
- 0x2F (/);
- 0x7F (символ исключения);
- 0xFF.

Если для представления международных символов вы используете кодировку UTF-8, предельная длина имени файла будет меньше, в зависимости от использования символов в расширенном диапазоне.

С помощью кодировки UTF-8 вы можете использовать международные символы в именах файлов. Если вы используете графическую оболочку Photon microGUI, это делается прозрачным образом (вы можете ввести необходимые

символы непосредственно с клавиатуры, и они корректно отобразятся в администраторе файлов Photon). Имена файлов, содержащие символы UTF-8, как правило, невидимы в командной строке.

Для отображения международных символов вы также можете использовать такие наборы символов, как ISO-Latin1 и набор символов для ПК. Однако отображение этих 8-битовых символов будет зависеть от установок вашего дисплея; кроме того, они могут выглядеть не так, как вы ожидаете, в оболочке Photon и в других операционных системах, которые получают доступ к этим файлам через сеть.

Большинство других операционных систем, включая Microsoft Windows, поддерживает символы UTF-8/Unicode, и их файловые имена выводятся корректно в среде Photon microGUI. Имена файлов из более старых версий Microsoft Windows могут быть закодированы с использованием 8-битовых символов из различных кодовых страниц. Файловая система DOS в ЗОСРВ «Нейтрино» может перевести эти имена файлов в кодировку UTF-8, но для этого необходимо сообщить системе, какую кодовую страницу использовать, посредством соответствующей опции в командной строке. Более подробные сведения см. в описании fs-dos.so в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

> **Примечание**. Все дисковые файловые системы ЗОСРВ «Нейтрино» за исключением fs-qnx4.so — т.е. fs-cd.so, fs-dos.so, fs-ext2.so, Power-safe (fs-qnx6.so) и fs-udf.so — используют для представления файловых имен кодировку UTF-8; попытка задать в этих файловых системах имя файла не в кодировке UTF-8 будет неуспешной (с ошибкой EILSEQ).

6.3. Где все хранится?

Файловая система ЗОСРВ «Нейтрино», принятая по умолчанию, в основном отвечает стандарту на структуру каталогов файловой системы (Filesystem Hierarchy Standard), однако мы не заявляем о полной совместимости с ним. Данный стандарт описывает, где должны размещаться файлы и каталоги в UNIX-подобных

операционных системах. Более подробную информацию можно найти на сайте: **http://www.pathname.com**.

Примечание. Пространство путевых имен в ЗОСРВ «Нейтрино» чрезвычайно гибкое, поэтому вы можете по-разному конфигурировать вашу систему.

В данном разделе описывается содержание каталогов, изображенных на рис. 6.4.

Рис. 6.4. Иерархия каталогов

/

Каталог / является корневым каталогом пространства путевых имен, обычно здесь располагается файловая система вашего основного жесткого диска или флэш-памяти. В файловой системе QNX 4 этот каталог содержит следующие файлы:

- /.altboot — альтернативный образ ОС, который загружается, если во время начальной загрузки вы нажали клавишу <Esc> (см. раздел 8);
- /.bitmap — системный файл, содержащий битовую карту, отображающую области диска, используемые файловой системой. Каждый блок представляется одним битом. Если бит установлен, значит, файловая система использует соответствующий блок.

Для предотвращения порчи диска вам необходимо следить за целостностью этого файла. После неожиданного выключения электропитания запустите утилиту chkfsys, которая просмотрит всю файловую систему, проверит содержание файла и при необходимости скорректирует его. Более подробные сведения см. в

разд. "Файловая система QNX 4" раздела 11, а также в описании утилиты chkfsys в руководстве "Описание программы. Часть 1. Системные утилиты";

- /.boot — в загружаемой файловой системе Power-Safe (fs-qnx6.so) это каталог, содержащий образы ОС, которые могут загружаться вторичным загрузчиком при начальной загрузке системы. В загружаемой файловой системе QNX4 (fs-qnx4.so) это файл, содержащий основной образ ОС. Более подробные сведения см. в разделе 8.
- /.diskroot — файл, указывающий, какую из файловых систем QNX 4 монтировать в качестве корневого каталога /. Более подробные сведения см. в разделе 8;
- /.inodes — содержит дополнительные данные, указывающие на дополнительные блоки индексных дескрипторов, требуемые для файлов, занимающих более одного экстента (т. е. более одной непрерывной последовательности блоков на дисковом устройстве). Более подробные сведения см. в подразд. "Файловая система QNX 4" раздела 11.

Корневой каталог / также содержит каталоги, характерные для используемой платформы (например, mipsle, ppcbe, x86), а также каталоги, перечисленные в последующих разделах.

/bin

Каталог /bin содержит двоичные файлы базовых утилит, таких как chmod, ls и ksh.

Чтобы узнать основной синтаксис утилиты, наберите в командной строке use имя_утилиты. Более подробные сведения см. в описании утилиты use в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

/boot

Каталог /boot содержит файлы и каталоги, используемые для создания загружаемых образов ОС (образные файловые системы). Образные файловые системы содержат компоненты ОС, исполняемые файлы и файлы данных, которые

должны присутствовать и запускаться сразу же после загрузки системы. Общие сведения по этой теме см. в описании утилиты mkifs в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

Этот каталог содержит:

- /boot/build/ — каталог с файлами, используемыми утилитой mkifs для построения образов ОС. Для стандартной системы ЗОСРВ «Нейтрино» на основе архитектуры x86 файлами построения образа являются base.build и basedma.build;
- /boot/fs/ — как это обычно принято, мы используем этот каталог для сохранения образных файловых систем, построенных при помощи утилиты mkifs. Для того чтобы выполнить загрузку с одного из образов, необходимо сначала скопировать его в каталог /.boot на загрузочном устройстве в файловой системе QNX 4;
- /boot/sys/ — каталог с начальным загрузчиком и кодом начальной загрузки. Это один из путей, который просматривает утилита mkifs при попытке обнаружить компоненты, указанные в файле компоновки.

/dev

Как было указано ранее, каталог /dev принадлежит администратору процессов. В этом каталоге содержатся файлы устройств, в том числе там могут находиться следующие файлы:

- /dev/cdn — блочное устройство CD-ROM (сведения о драйверах см. в описании devb-* в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01);
- /dev/conn — консольные tty-устройства с выводом в текстовом режиме (см. описание devc-con в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01);
- /dev/console — устройство, используемое для диагностических сообщений. В полной системе на основе архитектуры x86 это устройство предназначено только для записи и управляется утилитой slogger. Файлы построения образа для встраиваемых систем могут создавать ссылку от этого пути

к другому устройству, такому как последовательный порт (см. описание утилиты slogger в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01);

- /dev/fdn — блочное устройство флоппи-диска (сведения о драйверах см. в описании утилиты devb-fdc в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01);
- /dev/hdn — блочное устройство жесткого диска; данные, представляющие весь диск целиком, охватывающие все его разделы (см. описание утилит devb-* в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01);
- /dev/hdntn — блочное устройство раздела жесткого диска; данные в этих устройствах являются подмножеством данных, представленных соответствующим файлом hdn (см. описание утилит devb-* в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01);
- /dev/io-net/ — каталог, принадлежащий утилите io-net и управляемый ею. В этом каталоге находятся файлы, относящиеся к сетевым устройствам локальных сетей. Программы на С могут применять к этим файлам функции devctl() для обеспечения взаимодействия с драйвером, например для получения статистики драйвера;

Примечание. В каталоге /dev/io-net/ свои элементы создают только унаследованные драйвера io-net; оригинальные драйвера io-pkt* этого не делают.

- /dev/mem — устройство, представляющее всю физическую память;
- /dev/mq, /dev/mqueue — пространство путевых имен, в котором появляются записи об очередях сообщений;
- /dev/null — "мусорная корзина", в которую вы можете направлять данные. Данные удаляются;
- /dev/parn — параллельные порты, например для принтеров (более подробные сведения о конфигурировании см. в описании stty; о драйверах — в описании утилиты devc-par в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01);

- /dev/pci — это устройство, создаваемое сервером PCI, дает возможность программам взаимодействовать с этим сервером (см. описание утилит pci-* в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01);
- /dev/phfont — создается сервером шрифтов Photon, либо утилитой io-graphics с использованием библиотеки phfont.so, либо phfont в качестве отдельного процесса. Этот файл дает возможность программам взаимодействовать с сервером шрифтов (см. описание утилит io-graphics и phfont в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01);
- /dev/photon — специальный файл, который используется программами для присоединения к серверу Photon, запущенному на данной машине. Более подробные сведения см. в описании Photon в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01;
- /dev/pipe — принимается администратором неименованных программных каналов (pipe). Присутствие этого файла говорит другим программам (например, сценарию начального запуска, встроенному в образ ОС), что администратор неименованных программных каналов (Pipe manager) выполняется успешно;
- /dev/pty[p-zP-T][0-9a-f] — управляющая сторона псевдотерминальной пары устройств. Псевдотерминальные устройства имеют имена, включающие один символ (p-z или P-T), после которого следует шестнадцатеричная цифра, что позволяет иметь до 256 устройств. См. описание утилиты devc-pty в руководстве по системным утилитам;
- /dev/random — устройство для получения случайных данных (см. описание утилиты random в справочнике по утилитам);
- /dev/sem — пространство путевых имен, в котором появляются записи об именованных семафорах;
- /dev/sern — последовательные порты. Информация по конфигурированию приведена в описании утилиты stty, информация о драйверах — в описании утилит devc-ser* в справочнике по утилитам;
- /dev/shmem/ — содержит файлы, представляющие области разделяемой памяти системы (также иногда используемые для файлов, отображаемых в памяти).

Более подробные сведения см. в разд. "“Файловая система” в оперативной памяти: каталог /dev/shmem" раздела 11;

- /dev/slog — устройство, управляемое утилитой slogger и используемое для чтения или записи системного журнала. Для чтения информации из журнала используйте команду sloginfo /dev/slog. Более подробные сведения см. в описании утилит slogger и sloginfo в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01;
- /dev/socket/ — каталог, управляемый стеком TCP/IP, входящим в модуль io-pkt*. Этот каталог содержит путевые имена, по которым приложения взаимодействуют со стеком. Более подробные сведения см. в разделе 13;
- /dev/text — этот файл управляется утилитой procnto. Текст, записанный на это устройство, выводится через процедуры отладочного вывода начального загрузчика, указанные в коде начального запуска.

Результат варьируется в зависимости от процессорной платы. На стандартном ПК (с BIOS) по умолчанию выполняется вывод на консоль ПК. Более подробные сведения см. в описании утилит startup-* в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01;

- /dev/tty — виртуальное устройство, принадлежащее администратору процессов (procnto), которое позволяет определять управляющее терминальное устройство, связанное с сеансом любого процесса, который открывает данный файл. Это полезно для программ, которые уже закрыли свой стандартный поток ввода, вывода или ошибок, но позже возникла необходимость вывода на терминальное устройство;
- /dev/tty[p-zP-T][0-9a-f]— исполнительная сторона соответствующего файла /dev/pty[p-zP-T][0-9a-f]. Управляемая программа обычно использует один из этих файлов для своих стандартных потоков ввода, вывода или ошибок;
- /dev/zero — обеспечивает бесконечный поток байтов, имеющих нулевое значение.

/etc

Каталог /etc содержит специфичные системные файлы и программы, используемые для управления и конфигурирования, включая следующие:

- /etc/acl.conf — определяет разрешенные операции на заданном контексте SNMP (см. описание утилиты /etc/acl.conf в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01);
- /etc/autoconnect — сценарий автоматического соединения и конфигурации TCP/IP (см. описание утилиты /etc/autoconnect в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01);
- /etc/bootptab — конфигурационный файл сервера протокола сетевой загрузки (см. описание утилиты /etc/bootptab в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01);
- /etc/config/ — каталог, содержащий системные конфигурационные файлы, такие как файл ttys, который используется утилитой tinit для конфигурирования терминальных устройств;
- /etc/context.conf — определения контекста для SNMP v2 (см. описание утилиты /etc/context.conf в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01);
- /etc/country — создается утилитой phlocale. Используется приложениями для настройки применительно к той стране, в которой используется система;
- /etc/default/ — каталог, содержащий принятые по умолчанию конфигурационные файлы, прежде всего для средств TCP/IP;
- /etc/dhcpd.conf — протокол динамического конфигурирования узла (Dynamic Host Configuration Protocol, DHCP) (см. описание утилиты /etc/dhcpd.conf в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01);
- /etc/ftpd.conf — определяет опции конфигурации для ftpd, которые используются после аутентификации соединения (см. описание утилиты /etc/ftpd.conf в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01);

- /etc/ftpusers — определяет пользователей, которые могут иметь доступ к машине через протокол FTP (см. описание утилиты /etc/ftpusers в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01);
- /etc/group — база данных групп пользователей (см. раздел 3);
- /etc/hosts — справочная база имен хостов; см. также пункт /etc/resolv.conf далее в этом списке и описание утилиты /etc/hosts в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01;
- /etc/inetd.conf — конфигурационный файл суперсервера Интернета, определяющий сервисы Интернета, которые утилита inetd динамически запускает и прекращает (см. описание утилиты /etc/inetd.conf в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13);
- /etc/mib.txt — определяет формат для задания имен переменных для утилит SNMP (см. описание утилиты /etc/mib.txt в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01);
- /etc/motd — содержит ASCII-сообщение, которое может быть выведено при входе пользователей в систему, если /etc/profile сконфигурирован для этого.

Принятый по умолчанию /etc/profile выводит этот файл, только если файл /etc/motd имеет более позднюю дату, чем дата последнего входа в систему, определяемая по времени последней модификации файла $HOME/.lastlogin. Более детальная информация приведена в описании /etc/profile в разделе 9;

- /etc/networks — файл базы данных сетевых имен. Более детальная информация приведена в описании утилиты /etc/networks в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01;
- /etc/nsswitch.conf – конфигурационный файл для переключения между службами имен. Для получения дополнительных сведений см. описание /etc/nsswitch.conf в справочнике по утилитам.
- /etc/opasswd — резервная копия файла /etc/passwd, создаваемая перед его последним изменением утилитой passwd (см. раздел 3);
- /etc/oshadow — резервная копия файла /etc/shadow, создаваемая перед его последним изменением утилитой passwd (см. раздел 3);

- /etc/party.conf — конфигурационный файл для SNMP v2. Более детальная информация приведена в описании утилиты /etc/party.conf в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01;
- /etc/passwd — этот файл определяет учетные записи для входа в систему. Более детальная информация приведена в разделах 2 и 3, а также в описании утилит passwd, login, phlogin2, и phlogin в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01;
- /etc/photon/ — каталог, содержащий некоторые конфигурационные файлы, относящиеся к графической оболочке Photon, включая следующие:
- pterm — конфигурационные файлы для утилиты pterm;
- shelf/ — каталог, содержащий принятый по умолчанию конфигурационный файл для системной панели (shelf) и принятый по умолчанию формат меню Launch;
- shells/ — дополнительный каталог, в который вы можете помещать конфигурационные файлы для phlogin2 или phlogin;
- wm — конфигурационные файлы для администратора окон pwm.

Более подробные сведения см. в разделе 5;

- /etc/printers/ — каталог, содержащий файлы printertype.cfg и файл fontmap, которые используются утилитой phs-to-ps. Более детальная информация приведена в подразд. "Печать с помощью утилиты spooler" раздела 14;
- /etc/profile — сценарий профиля начальной загрузки, выполняемый командным интерпретатором при входе в систему; выполняется до $HOME/.profile (см. раздел 9);
- /etc/profile.d/ — каталог, в котором принятый по умолчанию сценарий /etc/profile ищет сценарии для выполнения при входе пользователя в систему. Сценарий /etc/profile запускает каждый сценарий в данном каталоге, который соответствует *.$(SHELL##*/}. Например, если переменная окружения SHELL имеет значение /bin/sh, то этот сценарий запускает сценарии, соответствующие *.sh;

- /etc/rc.d/ — каталог, в котором обычно хранятся локальные файлы инициализации системы. Более детальная информация приведена в описании /etc/system/sysinit в разделе 8;
- /etc/resolv.conf — конфигурационный файл распознавателя (см. приведенное ранее описание каталога /etc/hosts, а также описание утилиты /etc/resolv.conf в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01);
- /etc/skel/ — каталог, в котором располагается принятая по умолчанию версия .profile. При добавлении нового пользователя этот файл копируется в домашний каталог пользователя. Более детальная информация приведена в описании /etc/default/passwd в документации к passwd, а также в описании .profile в разделе 9;
- /etc/system/ — каталог, включающий файлы и каталоги, используемые при загрузке системы, в том числе:
- /etc/system/sysinit — основной сценарий для инициализации системы;
- /etc/system/config/nophoton — файл, указывающий на запрет запуска Photon;
- /etc/system/config/useqnet — файл, указывающий на запуск Qnet (более детальная информация приведена в разделе 12);
- /etc/system/enum — расположение конфигурационных файлов для программы распознавания устройств (см. также раздел 8).
- /etc/timezone/ — каталог, в котором утилита phlocale ищет список возможных временных зон (см. разд. "Задание часового пояса" раздела 9).

/fs

Дополнительные файловые системы монтируются в каталоге /fs. Более подробные сведения см. в разделе 11, а также в описании devb-* и mount руководства «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01. Данный каталог может включать:

- /fs/cdn/ — файловые системы CD-ROM;

- /fs/fdn/ — файловые системы флоппи-дисков;
- /fs/hdn-тип[-число]/ — файловые системы на разделах жесткого диска.

/home

Здесь находятся домашние каталоги обычных пользователей. Имя домашнего каталога часто совпадает с именем пользователя.

/lib

Каталог, содержащий разделяемые библиотеки, необходимые для работы программ (имя_файла.so), а также статические библиотеки, используемые в процессе разработки (см. также /usr/lib и /usr/local/lib).

Каталог /lib включает в себя:

/lib/dll/, содержащий дополнительные разделяемые библиотеки, которые служат в качестве драйверов и служб ОС (например, администраторов файловых систем и т. п.). Некоторые примеры использования разделяемых библиотек для различных типов драйверов и служб см. в разделах "Файловые системы", "Сеть Qnet" и "Поддержка стека протоколов TCP/IP" в руководстве по системной архитектуре. Подробная информация по конкретным разделяемым объектам в каталоге /lib/dll приведена в соответствующих пунктах в справочнике по утилитам.

/proc

Этот виртуальный каталог, принадлежащий администратору процессов (procnto), содержит информацию о процессах и конфигурации пространства путевых имен.

Каталог /proc содержит подкаталог для каждого процесса. Идентификатор процесса используется в качестве имени каталога. Каждый из этих каталогов включает в себя запись (as), которая определяет адресное пространство процесса. Различные утилиты используют эту запись для получения информации о процессе.

Каталог /proc также содержит:

- /proc/boot/ — образ файловой системы, включающий в себя образ начальной загрузки;

- /proc/dumper — специальный файл, в который отправляется сообщение при аварийном прекращении процесса;
- /proc/self/ — адресное пространство процесса, производящего запрос;
- /proc/mount/ — точки монтирования пространства путевых имен.
- /proc/qnetstats — файл, регистрируемый в /proc модулем lsm-qnet.so при использовании протокола Qnet. Если открыть этот файл и выполнить чтение из него, то ресурс-менеджер Qnet вернет текущую статистику по использованию данного протокола.

Примечание. Когда вы выводите список каталога /proc, список /proc/mount не отображается, однако вы можете отдельно вывести список каталога /proc/mount.

/root

Каталог /root является домашним каталогом пользователя root.

/sbin

Этот каталог содержит важные двоичные файлы системы, включая:

- драйверы (например, devb*, devc*, devf*, devp*, devu*);
- программы распознавания устройств (например, enum-devices);
- программы инициализации (например, diskboot, seedres);
- утилиты конфигурирования (например, dinit) и утилиты восстановления (например, chkfsys, chkdosfs);
- администраторы (например, io-pkt*, mqueue, pipe).

Многие из этих файлов используются при загрузке системы (более детальная информация приведена в разделе 8).

/tmp

Этот каталог содержит временные файлы. Предполагается, что программы должны удалять созданные ими временные файлы после их использования, однако иногда они этого не делают либо вследствие ошибок в коде, либо из-за аварийного завершения. Периодически можно проводить очистку этого каталога от ненужных временных файлов.

/usr

Каталог /usr содержит предназначенные только для чтения разделяемые данные. Этот каталог включает в себя:

- /usr/bin/ — каталог, содержащий большинство пользовательских команд (например, diff, errno и wc);
- /usr/help/ — каталог, содержащий документацию (в каталоге product) и стандартные изображения (в каталоге lib/images). Более детальная информация приведена в подразд. "Обозреватель справки" раздела 5 и в описании утилиты helpviewer руководства «Описание программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01;
- /usr/include/ — верхний уровень каталогов, содержащих заголовочные файлы C и C++. Этот каталог включает в себя каталог sys, каталоги, специфичные для платформы, и другие каталоги;
- /usr/info/ — документация для различных утилит;
- /usr/lib/ — объектные файлы, библиотеки и внутренние двоичные файлы, которые не должны выполняться непосредственно или в сценариях. Эти библиотеки путем компоновки используются при разработке программ;
- /usr/libexec/ — каталог, который может содержать системные сервисы и системные утилиты. Как привило, они выполняются только по запросу других программ;
- /usr/local/ — каталог, в котором администратор системы может устанавливать локальное программное обеспечение. Изначально этот каталог пустой;
- /usr/man/ — файлы справочной системы man-страниц для различных утилит;
- /usr/photon/ — верхний уровень каталогов, содержащих исполняемые файлы, файлы данных и др., связанные с графической оболочкой Photon;
- /usr/qde/ — верхний уровень каталогов, содержащих выполняемые файлы, файлы данных, подключаемые модули и др., связанные с интегрированной средой разработки, которая поставляется в составе комплекта разработчика;

- /usr/sbin/ — необязательные системные двоичные файлы (например, cron, dumper, nicinfo);
- /usr/share/ — данные, не зависящие от архитектуры, например пиктограммы, фоны и различные программы gawk;
- /usr/src/ — каталог для исходного текста.

/var

Каталог /var содержит различные файлы данных, в том числе кэш-файлы, файлы блокировок, журнальные файлы, а также следующие каталоги:

- /var/dumps — каталог, в котором dumper сохраняет все дампы, созданные в результате аварийного завершения программ.

6.4. Владение файлами и права доступа

Каждый файл и каталог относится к определенному идентификатору пользователя или идентификатору группы и имеет набор прав доступа (также называемых режимами — modes). Для управления владением и правами доступа к ним используются следующие утилиты (табл. 6.1).

Таблица 6.1

Действие	**Утилита**
Задать права доступа для файла или каталога	chmod
Изменить владельца (и при необходимости группу) для файла или каталога	chown
Изменить группу для файла или каталога	chgrp

Более подробные сведения см. в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

Примечание. Вы можете изменить права доступа (permission) или владения (ownership) файла или каталога, только если вы являетесь его владельцем или вошли в систему как пользователь root. Если вы хотите изменить как права доступа, так и права владения, следует сначала изменить права доступа.

После того как вы назначите права владения другому пользователю, вы не сможете изменить права доступа.

Права доступа подразделяются на следующие категории:

- u — права доступа для пользователя (т. е. владельца);
- g — права доступа для группы;
- o — права доступа для других пользователей (т. е. для всех, кто не является членом группы).

Каждый набор прав доступа включает в себя:

- r — права доступа на чтение;
- w — права доступа на запись;
- x — права доступа на выполнение. Для каталога это права доступа на просмотр содержимого каталога и поиск по нему;
- s или S — Setuid или setgid (см. далее);
- t или T — sticky-бит (см. далее).

Например, если вы запросите список содержимого домашнего каталога (с помощью ls -al), то можете получить такой результат:

```
total 94286
drwxr-xr-x 18  barney   techies     6144  Sep 26 06:37 ./
drwxrwxr-x  3  root     root        2048  Jul 15 07:09 ../
drwx------  2  barney   techies     4096  Jul 04 11:17 .AbiSuite/
-rw-rw-r--  1  barney   techies      185  Oct 27  2000 .Sig
-rw-------  1  barney   techies       34  Jul 05  2002 .cvspass
drwxr-xr-x  2  barney   techies     2048  Feb 26  2003 .ica/
-rw-rw-r--  1  barney   techies      320  Nov 11  2002 .kshrc
-rw-rw-r--  1  barney   techies        0  Oct 02 11:17 .lastlogin
drwxrwxr-x  3  barney   techies     2048  Oct 17  2002 .mozilla/
drwxrwxr-x 11  barney   techies    2 048  Sep 08 09:08 .ph/
-rw-r--r--  1  barney   techies      254  Nov 11  2002 .profile
drwxrwxr-x  2  barney   techies     4096  Jul 04 09:06 .ws/
-rw-rw-r--  1  barney   techies     3585  Dec 05  2002 123.html
```

Первая колонка представляет собой набор прав доступа. Буква d впереди указывает на то, что данный элемент является каталогом (см. разд. "Типы файлов" ранее в этом разделе).

Вы также можете использовать восьмеричные числа для указания режимов (см. описание утилиты chmod в справочнике по утилитам).

Setuid и setgid

Для того чтобы некоторые программы (например, passwd) работали корректно, они должны выполняться от имени определенного пользователя:

```
$ which -l passwd
-rwsrwxr-x  1  root     root       21544  Mar 30 23:34 /usr/bin/passwd
```

Заметим, что третий символ в правах доступа владельца — s. Этот символ обозначает команду setuid (от англ. set user ID — установить идентификатор пользователя). Когда вы запускаете утилиту passwd, эта программа выполняется как владелец файла (т. е. root). Символ S означает, что бит setuid для данного файла установлен, а бит выполнения – не установлен.

Вы также можете найти некоторые команды setgid (от англ. set group ID — установить идентификатор группы), которые выполняются с тем же идентификатором группы, что и у владельца файла, но не с идентификатором владельца-пользователя. Если атрибут setgid установлен для каталога, то содержащиеся в этом каталоге файлы будут иметь идентификатор группы каталога, а не идентификатор создателя файла.

Эта схема часто применяется для областей спулинга, таких как /usr/spool/mail, которой владеет группа mail и для которой установлен соответствующий атрибут setgid, поэтому программы, выполняющиеся в качестве группы mail, могут вносить изменения в этом каталоге, но его файлы по-прежнему принадлежат своим обычным владельцам.

Примечание. Если вы изменяете владельца исполняемого файла, имеющего бит setuid, то, если вы не зарегистрированы в системе как root, бит setuid

сбрасывается. Аналогичным образом, если вы изменяете группу программы, имеющей бит setgid, то этот бит сбрасывается, если вы не root.

При выполнении утилит mkefs, mketfs и mkifs в среде Windows невозможно получить из параметров файла атрибуты прав доступа на выполнение (x), setuid ("set user ID") или setgid ("set group ID"). Для того чтобы определить эти права доступа явным образом, используйте атрибут perms. Для корректного определения прав владения, возможно, понадобится использование атрибутов uid и gid. Сведения о том, необходима ли установка битов setuid или setgid для той или иной утилиты, можно найти в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

Внимание! Программы с битами setuid и setgid могут привести к возникновению проблем безопасности. Если вы установили эти биты, убедитесь в том, что только владелец имеет права на запись к этим программам и что несанкционированный пользователь не может ими воспользоваться, особенно если их владельцем является root.

Sticky-бит

Sticky-бит — это право доступа, связанное с исполняемыми файлами и каталогами.

- если sticky-бит установлен для выполняемого файла, ядро сохраняет этот исполняемый файл в памяти в течение некоторого времени после завершения программы. Это может повысить производительность системы, если вы часто запускаете программу (например, компилятор или компоновщик).
- для каталога атрибут sticky-бит определяет, кто может удалить файл в каталоге. Для выполнения этого действия всегда необходимо иметь для этого каталога права доступа на запись, однако если для него установлен sticky-бит, вам также необходимо быть владельцем файла или каталога или иметь права доступа на запись для этого файла.

Если третий символ в наборе прав доступа — t (например, r-t), это означает, что установлены как sticky-бит, так и право на исполнение, тогда как символ T указывает на то, что установлен только sticky-бит.

Принятые по умолчанию права доступа к файлу

Используйте команду umask для задания маски прав доступа для новых файлов. По умолчанию принята маска 002, поэтому все новые файлы имеют права доступа на чтение и запись для пользователя (т. е. владельца файла) и всех остальных членов группы этого пользователя, а также права доступа на чтение для всех остальных пользователей. Если вы хотите удалить права доступа на чтение и запись для других пользователей, добавьте следующую команду к вашему профилю .profile:

```
umask 006
```

Если вы являетесь администратором системы и хотите, чтобы это изменение было применено ко всем пользователям, измените установку umask в /etc/profile. Более подробные сведения о профилях можно найти в разделе 9.

6.5. Расширения файловых имен

В табл. 6.2 приведены некоторые наиболее часто встречающиеся расширения имен файлов, используемые в ЗОСРВ «Нейтрино».

Таблица 6.2

Расширение	**Описание**	**Соответствующие программы/утилиты**
.1	Текст формата Troff, например, из справочных man-страниц UNIX	man и troff в репозитории продуктов третьих сторон
.a	Библиотечный архив	ar
.awk	Сценарий Awk	awk
.b	Библиотека или программа арифметической утилиты bc (bench calculator)	bc
.bat	Командный файл MS-DOS	Для использования в системах DOS. В ЗОСРВ «Нейтрино» выполняться не будет (см. раздел 10 и описание

Таблица 6.2

Расширение	**Описание**	**Соответствующие программы/утилиты**
		утилиты ksh)
.bmp	Графическое растровое изображение	pv (программа просмотра в графической среде Photon)
.build	Файл построения образа ОС	mkifs
.c	Исходный текст программы на С	qcc, make (необходим комплект разработчика)
.C, .cc, .cpp	Исходный текст программы на С++	QCC, make (необходим комплект разработчика)
.cfg	Конфигурационные файлы различных форматов	Различные программы разных форматов
.conf	Конфигурационные файлы различных форматов	Различные программы разных форматов
.css	Каскадные стилевые таблицы (CSS)	Используется в комплекте разработчика для документации Eclipse
.def	Файл определений С++	QCC, make (необходим комплект разработчика)
.dll	Динамически компонуемая библиотека MS Windows	В ЗОСРВ «Нейтрино» не используется явным образом. Необходима для поддержки некоторых программ, выполняемых в ОС MS Windows (например, некоторых инструментов разработки). См. описание .so (разделяемые объекты) ЗОСРВ «Нейтрино»
.gif	Графическое изображение в формате GIF	pv (программа просмотра в среде Photon)
.gz	Сжатый файл	gzip (для создания резервных копий и восстановления данных)
.h	Заголовочный файл С	qcc, make (необходим комплект разработчика)
.htm	Файл языка разметки гипертекста (HTML) для просмотра Web-страниц	Web-браузер Voyager
.html	Файл языка разметки гипертекста (HTML) для просмотра Web-страниц	Helpviewer, Web-браузер Voyager
.ifs, .img	Файловая система образов	mkifs;

Таблица 6.2

Расширение	**Описание**	**Соответствующие программы/утилиты**
	ЗОСРВ «Нейтрино»; обычно загружаемый образ	
.jar	Архив Java, состоящий из множества файлов Java (файлы классов и т. д.), сжатых в один файл	Приложения Java (например, IDE)
.jpg	Графическое изображение в формате JPEG	pv (программа просмотра в среде Photon)
.kbd	Компилированные файлы определений клавиатуры среды Photon	Photon, mkkbd
.kdef	Исходные файлы определений клавиатуры среды Photon	mkkbd
.kev	События ядра, сгенерированные инструментированным ядром и используемые для профилирования системы ЗОСРВ «Нейтрино» целиком	procnto*-instr, tracelogger, traceprinter
.mk	Исходный текст make-файла, обычно используемый для рекурсивной компиляции	make (комплект разработчика)
.o	Двоичный файл, полученный в результате компиляции исходного файла на языке C, C++ или ассемблере	qcc, make (комплект разработчика)
.pal	Файл палитры Photon	Photon
.pfr	Файл Bitstream TrueDoc Portable Font Resource	phfont
.phf	Файл растровых шрифтов	phfont
.S, .s	Файл исходного кода на языке ассемблера	Компилятор языка ассемблер GNU as (комплект разработчика)
.so, .so.n	Разделяемый объект	qcc, make (комплект разработчика)
.tar	Архив на магнитной ленте	tar (для создания резервных копий и

Таблица 6.2

Расширение	**Описание**	**Соответствующие программы/утилиты**
		восстановления данных)
.tar.gz, .tgz	Сжатый архив на магнитной ленте	gzip, tar (для создания резервных копий и восстановления данных)
.toc	Файл содержания для программы Helpviewer	Helpviewer
.TTF	Шрифты TrueType	phfont
.txt	Текстовый файл формата ASCII	Множество текстовых редакторов, приложений и индивидуальных пользователей
.ttf	Файл шрифтов TrueType	phfont
.use	Файл, содержащий краткую справку по использованию, включаемую в исполняемый файл программы, если эта информация не включена в исходный код	make (комплект разработчика)
.wav	Звуковой WAVE-файл	
.xml	XML-файл; различные области применения, в том числе документация IDE в комплекте разработчика	
.zip	Сжатый архивный файл	gzip

Если вы не уверены в формате файла, используйте утилиту file:

```
file имя_файла
```

6.6. Устранение неполадок

Далее описаны основные проблемы с файлами, с которыми вы можете столкнуться.

- я пытаюсь записать файл, но получаю сообщение "permission denied" ("доступ запрещен").

У вас нет прав доступа на запись для этого файла. Если вы являетесь владельцем (или root), вы можете изменить права доступа (см. разд. "Владение файлами и права доступа" ранее в этом разделе).

- я пытаюсь вывести список содержимого каталога, для которого имею право доступа, но получаю сообщение "permission denied" ("доступ запрещен").

Чтобы вывести список каталога, необходимо иметь права доступа на чтение и исполнение для данного каталога (см. подразд. "Владение файлами и права доступа" ранее в этом разделе).

- возникают проблемы с файлом, в имени которого есть пробел.

Командный интерпретатор разбирает командную строку и использует знак пробела для разбивки команды на лексемы (token). Если имя файла содержит пробел, необходимо заключить этот пробел в кавычки, чтобы командный процессор интерпретировал его буквально. Более подробные сведения об этом и других специальных символах см. в подразд. "Применение кавычек со специальными символами" раздела 4.

7. Редакторы

7.1. Выбор редактора

Редактор — это утилита, которая предназначена для просмотра и изменения файлов. Редакторы не применяют постоянное форматирование к просматриваемому тексту, хотя многие из них используют цвета и стили для отображения дополнительной контекстуальной информации, например информации о типах данных в файлах с исходным кодом программ. При редактировании кода на языке C некоторые редакторы отображают ключевые слова, строки, числа и другие элементы программы разными цветами.

Выбор редактора значительно зависит от личных предпочтений пользователя:

- требуется ли использовать мышь либо другое указательное устройство, или достаточно только клавиатуры?
- требуется ли вводить международные символы, знаки ударения и диакритические знаки, или только ASCII-символы?
- каков предпочтительный способ запуска команд? В одних редакторах команды активизируются вводом одного символа, в других – нажатием комбинации клавиш, в третьих – щелчком по кнопке или выбором элемента в меню.

Редакторы классифицируются на два важных типа: текстовые и графические. Текстовые редакторы обладают большей гибкостью, поскольку с ними можно работать как в текстовом режиме, так и в окне консоли графической оболочки Photon, удаленно с помощью утилит telnet и qtalk, а также другими способами. Графические редакторы более удобны и просты в использовании, но работают только в графическом окне.

Примечание. Чтобы иметь возможность использовать текущее окно во время работы редактора, рекомендуется запускать графический редактор из командной строки как фоновый процесс, добавляя символ амперсанда (&) в

конце команды. Текстовый редактор рекомендуется запускать как интерактивный процесс, опуская символ амперсанда.

Операционная система ЗОСРВ «Нейтрино» включает в себя следующие редакторы:

- vi — функциональный, но несколько сложный для освоения текстовый редактор, который имеется в большинстве (а возможно, и во всех) UNIX-подобных операционных системах, подобных UNIX;
- ped – простой в использовании графический редактор графической оболочки Photon;
- qed – устаревший полноэкранный текстовый редактор.

Комплект разработчика включает в себя интегрированную среду разработки (IDE), в состав которой входят различные специальные редакторы для создания программ на языках C и C++, файлов построения образа и др.

7.2. Поддерживаемые редакторы

vi

Редактор vi входит в состав всех операционных систем, подобных UNIX. Утилита vi фактически представляет собой визуальный интерфейс редактора с именем ex. Чтобы запустить редактор vi, следует ввести команду:

vi имя_файла

Редактор vi имеет следующие два режима.

- режим команд;

Клавиатура связывается с набором комбинаций клавиш быстрого запуска команд перемещения по тексту и редактирования текста. Команды редактора vi состоят из одной или несколько букв, а команды редактора ex начинаются с двоеточия (:).

- режим вставки;

Обеспечивает обычный ввод текста.

Чтобы переключиться в режим команд, следует нажать клавишу <Esc>, а для переключения в режим вставки — одну из следующих клавиш:

- <I> или <i>, чтобы выполнить вставку в начало текущей строки или перед курсором;
- <A> или <a>, чтобы добавить текст в конец текущей строки или после курсора;
- <O> или <o>, чтобы начать новую строку над или под курсором.

Два режима ввода в редакторе vi могут запутать начинающего пользователя, т. к. по умолчанию редактор vi не указывает режим, в котором он находится в данный момент. Если в режиме команд ввести команду:

:set showmode

то редактор отобразит текущий режим в правом нижнем углу монитора. Чтобы режим редактора vi отображался постоянно, следует добавить указанную команду (без символа двоеточия) в профиле утилиты vi — $HOME/.exrc.

В табл. 7.1 перечислены некоторые часто используемые команды редактора vi.

Таблица 7.1

Действие	**Команда**
Выйти из редактора vi без сохранения изменений	:q!
Сохранить текущий файл	:w
Сохранить текущий файл и выйти из редактора	:wq, :x или ZZ
Переместить курсор влево	h
Переместить курсор вправо	l
Переместить курсор на одну строку вверх	k
Переместить курсор на одну строку вниз	j
Переместить курсор в начало следующего слова	w
Переместить курсор в конец текущего или следующего слова (в зависимости от положения курсора)	e
Переместить курсор в начало текущего или предыдущего слова (в зависимости от положения курсора)	b
Переместить курсор на страницу назад	<Ctrl>+<B>
Переместить курсор на страницу вперед	<Ctrl>+<F>
Скопировать текущую строку	yy

Таблица 7.1

Действие	**Команда**
Скопировать текстовый фрагмент от курсора до конца текущего слова	yw
Удалить текстовый фрагмент от курсора до конца текущего слова	dw
Удалить текущую строку	dd
Вставить текст перед курсором	P
Вставить текст после курсора	p

Примечание. В некоторых реализациях редактора vi, в том числе в версии, которая входит в операционную систему ЗОСРВ «Нейтрино», курсор можно перемещать с помощью клавиш-стрелок как в режиме команд, так и в режиме вставки.

Для большего удобства можно использовать сочетания команд. Например, чтобы одновременно удалить несколько строк, можно ввести число перед командой dd. Кроме того, редактор vi имеет 26 именованных буферов, которые позволяют с легкостью вырезать, копировать и вставлять различные текстовые блоки.

Существуют многочисленные онлайновые ресурсы, учебные пособия и справочники по командам редактора vi. В ЗОСРВ «Нейтрино» vi фактически является ссылкой на утилиту elvis (см. руководство "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01).

ped

Редактор ped графической оболочки Photon представляет собой простой графический редактор, который похож на редакторы других оконных систем (рис. 7.1). Поскольку утилита ped работает в окне графической оболочки Photon, доступ к ней через текстовые консоли и системы, в которых отсутствует графический интерфейс, невозможен.

Редактор ped удобен для пользователей, которым необходимо вводить международные символы, знаки ударения и диакритические знаки, поскольку он поддерживает кодировку UTF-8. Для ввода международных символов в редакторе ped используются комбинации клавиш (compose sequences), которые описаны в

разделе "Photon compose sequences" руководства пользователя "Unicode Multilingual Support Photon Programmer’s Guide".

Чтобы запустить редактор ped, следует выбрать команду Utilities→Text Editor в меню Launch или ввести команду в окне терминала pterm:

```
ped [имя_файла] &
```

Более подробные сведения об использовании редактора ped см. в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

Рис. 7.1. Редактор графической оболочки Photon ped

7.3. Редактор по умолчанию

Некоторым системным процессам необходимо, чтобы пользователь вводил нужную им информацию с помощью текстового редактора. Например, система контроля версий CVS запрашивает у пользователя информацию о внесенных изменениях. Такие процессы определяют редактор, который требуется запускать, с помощью переменной окружения **VISUAL** или **EDITOR** (или обеих). По умолчанию используется редактор vi.

Переменная окружения **EDITOR** традиционно использовалась для задания строкового редактора, а переменная окружения **VISUAL** — для задания полноэкранного редактора. Приложения считывают одну из этих переменных или обе переменные одновременно. В некоторых приложениях, которые используют обе переменные, переменная **VISUAL** имеет приоритет, когда требуется запустить полноэкранный редактор, а переменная **EDITOR** — когда требуется запустить строковый редактор.

Современные приложения редко вызывают строковые редакторы, а пользователи часто задают несколько редакторов в переменной окружения **EDITOR**, поэтому нельзя рассчитывать на приоритеты, с которыми приложения используют эти переменные. Как правило, рекомендуется присваивать переменным окружения **VISUAL** и **EDITOR** одинаковые значения.

Опробовав различные редакторы, пользователь может задать в переменных окружения наиболее предпочтительный редактор. Для этого в приглашении командной строки необходимо ввести следующие команды:

```
export VISUAL=путь
export EDITOR=путь
```

где путь — это путь к исполняемому файлу редактора. Например, чтобы сделать JED редактором по умолчанию, следует ввести команды:

```
$ which jed
/usr/local/bin/jed
$ export VISUAL=/usr/local/bin/jed
$ export EDITOR=/usr/local/bin/jed
```

Чтобы проверить значение переменной окружения **EDITOR**, следует ввести команду:

```
echo $EDITOR
```

Значения переменных окружения **VISUAL** и **EDITOR** можно задать в профиле пользователя $HOME/.profile, чтобы они устанавливались при каждом

входе пользователя в систему. Более подробные сведения см. в разд. "Файл $HOME/.profile" раздела 9.

8. Управление запуском ЗОСРВ «Нейтрино»

Подробности процесса запуска системы зависят от оборудования компьютера. В этом разделе приводится лишь общее описание запуска операционной системы ЗОСРВ «Нейтрино».

Примечание. Чтобы изменить какие-либо файлы, которые система исполняет в процессе запуска, необходимо войти в систему как пользователь root.

8.1. Что происходит при загрузке?

При загрузке системы процессор сбрасывается и исполняет команду, на которую указывает его вектор сброса. Вектор сброса процессоров семейства x86 обычно указывает на BIOS, однако на других платформах вектор сброса может указывать на ПЗУ-монитор или команду прямого перехода на код начального загрузчика платы. После запуска ПЗУ-монитор обычно передает управление начальному загрузчику. BIOS делает то же самое либо непосредственно передает управление началу образа операционной системы (рис. 8.1).

Начальный загрузчик (IPL) копирует загрузочный образ в память и переходит на код начальной загрузки. Код запуска инициализирует оборудование, заполняет системную страницу информацией об оборудовании, загружает функции межбиблиотечных вызовов, которые используются ядром для взаимодействия с оборудованием, а затем загружает и запускает микроядро и администратор процессов — модуль procnto (который, начиная с версии 6.3.0, так же поддерживает именованные семафоры). Начальный загрузчик и код начальной загрузки, как правило, входят в состав BSP-пакета для конкретной платы.

После завершения своей инициализации модуль procnto выполняет команды, помещенные в загрузочный сценарий, которые позволяют продолжить настройку среды исполнения с помощью сценария командного интерпретатора или программы, написанной на языке C и/или C++.

Рис. 8.1. Загрузка операционной системы ЗОСРВ «Нейтрино»

Если система имеет процессор, который не принадлежит к семейству x86, и загружается с диска, то процесс настройки прост: большую его часть выполняет загрузочный сценарий или сценарий командного интерпретатора, который вызывается загрузочным сценарием.

Загрузка системы с архитектурой x86 с использованием BIOS выполняется сложнее (рис. 8.2).

Получив управление, BIOS конфигурирует оборудование, а затем проверяет наличие сигнатур расширений BIOS (0x55AA). BIOS вызывает свои расширения (например, сетевая плата с загрузочным ПЗУ или контроллером жесткого диска) до тех пор, пока одно из них не выполнит загрузку системы. Если ни одно расширение не загружает систему, BIOS выводит сообщение об ошибке (как правило, странное).

При сетевой загрузке аппаратный загрузчик (как правило, bootp) загружает образ с сервера, копирует его в память, а затем передает управление началу образа.

Поскольку загрузочному образу обычно необходимо использовать стек сетевого протокола, он запускает какую-либо сетевую файловую систему, чтобы загрузить дополнительные программы и файлы или получить к ним доступ.

Рис. 8.2. Загрузка ЗОСРВ «Нейтрино» в системе с архитектурой x86 с использованием BIOS

Чтобы создать образ операционной системы, можно воспользоваться утилитой mkifs. Пример файла построения образа операционной системы см. в приложении.

Процесс загрузки операционной системы ЗОСРВ «Нейтрино» с диска более сложен, а инициализация системы – еще сложнее. После того как BIOS выберет загрузку с жесткого диска, вызывается первичный загрузчик (который иногда называют загрузчиком раздела). Этот загрузчик "индифферентен" по отношению к операционной системе, т. е. способен загрузить любую операционную систему. Загрузчик, который устанавливается ЗОСРВ «Нейтрино», отображает следующее сообщение:

```
Press F1-F4 to select drive or select partition 1,2,3? 1
```
(Нажмите клавишу F1-F4 для выбора диска или раздела 1, 2, 3? 1)

После короткой паузы загружается операционная система, которая расположена в выбранном разделе. Эта загрузка выполняется загрузчиком /boot/sys/ipl-diskpc1. Записать загрузчик на диск можно с помощью утилиты dloader.

8.2. Загрузка образа ЗОСРВ «Нейтрино»

При выборе раздела с ОС запускается вторичный загрузчик (который иногда называют загрузчиком операционной системы). Этот загрузчик специфичен для ЗОСРВ «Нейтрино».

Файловая система Power-Safe

При использовании файловой системы Power-Safe (fs-qnx6.so) вторичный загрузчик осуществляет валидацию файловой системы и определяет самый новый из ее стабильных срезов (snapshot). Затем он отображает список всех подходящих файлов каталога .boot в прокручивающемся списке, из которого пользователь может выбрать необходимый загружаемый образ.

Если каталог .boot содержит только один файл, то загрузка начинается немедленно; в противном случае загрузчик 3-4 секунды ожидает нажатие кнопки. Для перемещения между файлами используются стрелки вверх и вниз, выбор осуществляется нажатием клавиши «Ввод» (Enter). Для перехода в начало или конец списка так же можно использовать клавиши Home и End. На экране может отображать до 10 файлов; для просмотра остальных необходимо удерживать клавишу со стрелкой вниз или вверх.

Если пользователь не нажимает клавишу, то по истечению таймаута загрузчик продолжит работу, используя образ по умолчанию. Это файл, имеющий самое недавнее время модификации, в списке он всегда отображается первым. В общем случае это будет последний образ, скопированный в каталог .boot; для

изменения образа по умолчанию можно использовать утилиту touch. Для определения образа по умолнанию можно выполнить команду:

```
ls -t /.boot | head -1
```

Начальный загрузчик может быть обновлен без переформатирования раздела (т.е. без потери данных файловой системы) с помощью команды: mkqnx6fs -B.

Для загрузки могут использоваться только файловые системы «little-endian» (т.е. отформатированные на любом компьютере командой mkqnx6fs -el либо отформатированные на компьютере «little-endian» без указания порядка следования байт).

Начальный загрузчик поддерживает только два уровня иерархии блоков файловой системы. Следовательно при использовании 512-байтных блоков «область видимости» (cutover) составляет 128 Кбайт, поэтому файловая система, отформатированная командой mkqnx6fs -b512 вероятнее всего не будет пригодной для загрузки. При использовании блоков размером 1 Кбайт (это значение по умолчанию) «область видимости» составляет 1 Гбайт.

Начальный загрузчик может выдавать следующие сообщения об ошибках:

```
Unsupported BIOS
```

Данная BIOS не поддерживает расширения INT13 LBA.

```
Missing OS Image
```

Файловая система не является fs-qnx6 либо каталог .boot пуст.

```
Invalid OS Image
```

Выбранный файл не является загружаемым образом x86.

```
Disk Read Error
```

При чтении диска возникла физическая ошибка ввода-вывода.

```
Ram Error
```

При копировании образа возникла физическая ошибка ОЗУ.

Файловая система QNX 4

При использовании файловой системы QNX4 начальный загрузчик выведет следующее сообщение:

```
Hit Esc for .altboot
```

Если интервал ожидания истекает, то загрузчик загружает образ операционной системы из файла /.boot, а при нажатии клавиши <Esc> извлекает образ из файла /.altboot. В процессе считывания образа загрузчик печатает последовательность точек. Если происходит ошибка, загрузчик выводит один из перечисленных далее символов и процесс загрузки останавливается:

- S — сигнатура системы не обнаружена;
- D или ? — произошла ошибка при чтении диска.

Единственное различие между образами, которые устанавливаются по умолчанию, заключается в том, что образ /.boot осуществляет прямой доступ к памяти EIDE-контроллера, а образ /.altboot — нет.

В каталоге /boot/build/ находятся примеры файлов построения образа, например, basedma.build — файл построения образа .boot (см. приложение).

8.3. diskboot

Файл построения образа по умолчанию .boot, basedma.build включает в себя следующие строки:

```
[+script] startup-script = {
# To save memory make everyone use the libc in the boot image!
# For speed (less symbolic lookups) we point to libc.so.2 instead
# of libc.so
procmgr symlink ../../proc/boot/libc.so.2 /usr/lib/ldqnx.so.2

# Default user programs to priority 10, other scheduler (pri=10o)
# Tell "diskboot" this is a hard disk boot (-b1)
# Tell "diskboot" to use DMA on IDE drives (-D1)
# Start 4 text consoles buy passing "-n4" to "devc-con" (-o)
# By adding "-e" Linux ext2 filesystem will be mounted as well.
```

```
[pri=10o] PATH=/proc/boot diskboot -b1 -D1 -odevc-con,-n4
}
```

Этот сценарий запускает систему с помощью программы diskboot, которая предназначена для загрузки операционной системы ЗОСРВ «Нейтрино» на дисковых компьютерах.

Примечание. Можно передавать параметры программе diskboot (для управления загрузкой системы), а также драйверам устройств. В приведенном ранее файле построения образа программа diskboot передает параметр -n4 драйверу devc-con, чтобы задать число виртуальных консолей.

Можно настроить компьютер так, чтобы программа diskboot не использовалась.

Утилита diskboot позволяет обновлять драйвера devb-*. Подробнее см. пункт “Обновление дисковых драйверов” далее в этом разделе.

При запуске программа diskboot отображает следующее приглашение:

```
Press the space bar to input boot options...
```

(Нажмите клавишу пробела, чтобы ввести параметры загрузки...)

Большая часть этих параметров предназначена для отладочных целей. Программа diskboot выполняет поиск раздела операционной системы ЗОСРВ «Нейтрино», чтобы смонтировать его, а затем запускает последовательность сценариев для инициализации системы (рис. 8.3).

Рис. 8.3. Выполнение инициализации программой diskboot

Основным сценарием инициализации системы является /etc/system/sysinit. Локальные файлы инициализации обычно хранятся в каталоге /etc/rc.d. Чтобы запустить на узле дополнительные команды, например, для монтирования диска с файловой системой NFS, можно создать файл сценария с именем rc.local, сделать его исполняемым и поместить его в каталог /etc/rc.d. Более подробные сведения см. в описании файла rc.local далее в этом разделе.

Программа diskboot выполняет следующие действия.

- запускает утилиту журналирования slogger. Приложения используют утилиту slogger для регистрации системных сообщений в журнале. Просмотреть журнал с этими сообщениями можно с помощью утилиты sloginfo;
- затем программа diskboot запускает утилиту seedres, чтобы считать настройки BIOS для PnP-устройств и заполнить базу данных ресурсов модуля procnto;
- далее программа diskboot запускает утилиту pci-bios, чтобы обеспечить поддержку BIOS для PCI-устройств;
- после этого программа diskboot запускает драйвер devb-eide или другие дисковые драйверы.

Примечание. Параметры для драйвера devb-eide и других драйверов следует передавать программе diskboot в файле построения образа.

- затем программа diskboot осуществляет поиск файловых систем для монтирования (т. е. разделов и компакт-дисков), которое выполняется в зависимости от типа раздела. Программа diskboot распознает разделы следующих типов:
    - CD-ROM;
    - типы 1, 4, 6, 11, 12, 14 — операционная система DOS;
    - тип 131 — файловая система Ext2, если программе diskboot передан параметр -e;
    - тип 177, 178, 179: файловая система Power-Safe — 177 и 178 указывают вторичные разделы;
    - типы 77, 78, 79 — операционная система КПДА.00002-01.

Эти разделы монтируются в каталоги /fs/cdx (для CD-ROM) и /fs/hdx-тип-y, где x является номером диска (например, /fs/cd0, /fs/hd1), а y — увеличивающийся параметр, который добавляется для уникальности. Например, вторым DOS-разделом на жестком диске 1 будет /fs/hd1-dos-2.

В нарушение этого правила один раздел операционной системы QNX 4 по умолчанию монтируется в каталог /. Это можно проверить, просмотрев содержимое файла .diskroot на каждом разделе QNX 4. Если существует только один раздел QNX 4, в файле .diskroot которого определена точка монтирования /, этот раздел демонтируется из каталога /fs/hdx-тип-y, а затем монтируется в каталог /. Если имеется несколько таких разделов, программа diskboot предлагает выбрать один из них.

Файл .diskroot, как правило, пуст, но может содержать команды. Более подробные сведения см. далее.

- запускает расширенный встраиваемый интерпретатор fesh (необязательно).
- затем программа diskboot запускает драйвер консоли devc-con-hid;
- наконец, программа diskboot запускает главный сценарий инициализации системы /etc/system/sysinit.

8.4. .diskroot

Программа diskboot использует файл .diskroot для того, чтобы определить, какой раздел с операционной системой QNX 4 смонтировать в каталог /.

Файл .diskroot может являться:

- файлом с нулевой длиной; по умолчанию файл .diskroot имеет нулевую длину, что означает запрос на точку монтирования /;
- однострочным файлом, который задает требуемую точку монтирования, например:

/home

Строка не должна начинаться со знака номера (#) или содержать знак равенства (=). Программа diskboot игнорирует пробельные символы в начале и в конце строки;

- многострочным конфигурационным файлом; в этом случае файл .diskroot должен содержать спецификацию точки монтирования, а также может включать в себя дополнительные спецификации. Все спецификации имеют формат:

метка = значение

Программа diskboot игнорирует любые пробельные символы в начале и в конце строки, а также до и после знака равенства.

Распознаются следующие метки:

- mount или mountpt — каталог, в который выполняется монтирование. Пример:

mount = /home

- opt или options — параметры монтирования, общие или специфичные для конкретной точки монтирования. Параметры отделяются друг от друга запятыми. Пример:

options = ro,noexec

Более подробные сведения см. в документации по утилите mount и конкретным драйверам в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01;

- desc или description — программа diskboot распознает и разбирает эти метки, однако в настоящее время игнорирует информацию, которая содержится в них;
- type — программа diskboot распознает строки qnx4, ext2 и dos, но в настоящее время игнорирует эту метку. Программа diskboot определяет тип раздела по его номеру, как описано в разд. "diskboot" ранее в этом руководстве.

8.5. /etc/system/sysinit

Файл /etc/system/sysinit является сценарием, который запускает основные системные службы. Для того чтобы отредактировать этот файл, необходимо войти в систему как пользователь root.

Примечание. Перед тем как редактировать сценарий sysinit, рекомендуется создать резервную копию его последней работоспособной версии. Следует помнить, что сценарии, которые написаны пользователем, необходимо делать исполняемыми перед использованием (см. chmod в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01).

Сценарий sysinit выполняет следующие действия.

- запускает утилиту slogger, если она еще не запущена;
- запускает администратор каналов pipe. Этот администратор позволяет передавать выходные данные одной команды в качестве входных данных другой команде. Более подробные сведения см. в подразд. "Перенаправление ввода и вывода" раздела 4;
- затем сценарий sysinit запускает утилиту mqueue, которая управляет очередями сообщений и именованными семафорами (named semaphores), используя "традиционную" реализацию. Если вам нужна альтернативная реализация очередей сообщений, использующая асинхронный обмен сообщениями,

то вам необходимо запустить сервер mq. Дополнительную информацию см. в справочнике по утилитам.

- во время первой перезагрузки после установки операционной системы сценарий sysinit запускает сценарий /etc/rc.d/rc.setup-once, который создает различные каталоги и своп-файлы.
- далее, присваивает конфигурационной строке _CS_TIMEZONE значение, которое хранится в файле /etc/TIMEZONE. Если этот файл не существует, сценарий sysinit задает часовой пояс UTC (Universal Time Coordinated — всеобщее скоординированное время, ранее называвшееся Greenwich Mean Time — время по Гринвичу). Более подробную информацию см. в подразд. "Задание часового пояса" раздела 9.
- если файл /etc/rc.d/rc.rtc существует и является исполняемым, сценарий sysinit запускает его для настройки часов реального времени.

Рекомендуется устанавливать аппаратные часы по часовому поясу UTC, а временную зону задавать с помощью конфигурационной строки _CS_TIMEZONE или переменной окружения TZ. Система отображает и рассчитывает местное время, а также автоматически определяет начало и окончание летнего и зимнего времени.

Этот подход к управлению системным временем дает возможность пользователям, которые находятся в разных часовых поясах и подключаются к одному компьютеру, видеть правильное местное время. Кроме того, настройка аппаратных часов на время UTC удобна при передаче данных между разными часовыми поясами. К данным добавляется штамп времени по часовому поясу UTC, и все компьютеры с легкостью сравнивают штампы времени, которые установлены в разных часовых поясах.

Некоторые операционные системы, например Windows, устанавливают местное время на аппаратных часах. Если операционные системы Windows и ЗОСРВ «Нейтрино» устанавливаются на один компьютер, то следует задать на аппаратных часах местное время, выполнив следующую команду от имени пользователя root и поместив ее в файл /etc/rc.d/rc.rtc:

```
rtc -l hw
```

В графической оболочке Photon достаточно снять флажок **The hardware clock uses UTC/GMT** в программе phlocale. В этом случае утилита phlocale создаст файл rc.rtc, который содержит указанную выше команду.

- после установки часов сценарий sysinit записывает в переменную окружения HOSTNAME имя хоста системы. Сценарий sysinit определяет это имя с помощью команды hostname или считывает его из файла etc/HOSTNAME, если команда hostname не срабатывает;

  Примечание. Имя хоста может состоять только из букв, цифр и символов дефиса, а также не должно начинаться и заканчиваться символом дефиса. Более подробные сведения см. в RFC 952.

- затем сценарий sysinit запускает сценарий /etc/rc.d/rc.devices для распознавания устройств системы (см. разд. "Распознавание устройств" далее в этом разделе). В зависимости от обнаруженных устройств этот сценарий запускает драйвер io-pkt* и другие различные драйверы.

- если файл /etc/system/config/useqnet существует и драйвер io-pkt* работает, сценарий sysinit инициализирует собственную сеть операционной системы ЗОСРВ «Нейтрино» (см. раздел 12 и npm-qnet.so в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01).

- далее сценарий sysinit запускает сценарий инициализации системы /etc/rc.d/rc.sysinit (см. далее).

- если сценарий инициализации системы не срабатывает, то сценарий sysinit пытается запустить утилиту sh или fesh (если sh не срабатывает), чтобы обеспечить работу, как минимум, командного интерпретатора при отказе остальных компонентов системы.

8.6. Распознавание устройств

Для обнаружения всех известных аппаратных устройств компьютера и запуска соответствующих драйверов в операционной системе ЗОСРВ «Нейтрино» используется программа распознавания устройств (device enumerator) —

администратор enum-devices. Он вызывается сценарием /etc/rc.d/rc.devices, который запускается программой /etc/system/sysinit.

Чтобы определить действия, которые должны быть выполнены при обнаружении конкретных аппаратных устройств, администратор enum-devices использует набор конфигурационных файлов. После считывания конфигурационных файлов утилита enum-devices вызывает различные программы распознавания, чтобы определить, какие устройства имеются в системе. Затем идентификаторы устройств сравниваются с идентификаторами, которые указаны в конфигурационных файлах. Если устройство обнаруживается, то
исполняются операторы, которые связаны с этим устройством. Конфигурационные файлы программ распознавания расположены в каталоге /etc/system/enum.

Далее приведен пример кода конфигурационного файла:

```
device(pci, ven=2222, dev=1111)
   uniq(sernum, devc-ser, 1)
   driver(devc-ser8250, "-u$(sernum) $(ioport1),$(irq)")
```

Этот код указывает программе распознавания выполнить следующие действия при обнаружении устройства 1111 производителя 2222:

- установить параметр sernum равным следующему уникальному серийному номеру устройства, начиная с 1;
- запустить драйвер devc-ser8250 с указанными параметрами (программа распознавания устройств задает переменные ioport и irq).

Чтобы распознать новое оборудование или задать дополнительные параметры, можно расширить набор конфигурационных файлов следующими способами:

- каталог oem;
- файл overrides;
- набор файлов для распознавания, специфичный для данной системы.

Перед обработкой программа распознавания считывает и объединяет содержимое всех конфигурационных файлов, которые расположены в выбранном каталоге.

Более подробные сведения о других командно-строковых параметрах и описание синтаксиса конфигурационных файлов см. в enum-devices руководства «Описание программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

Каталог oem

Производитель комплектного оборудования, который написал собственные драйверы для устройств, может создать каталог oem в каталоге /etc/system/enum для хранения конфигурационных файлов устройств.

Файл overrides

Если необходимо настроить устройства или параметры, которые специфичны для конкретной системной конфигурации, следует создать файл overrides в каталоге /etc/system/enum. Программа распознавания включает этот файл в множество конфигурационных файлов последним и добавляет определения, которые содержатся в нем, в набор, с которым работает администратор enum-devices. Если файл overrides содержит определение, которое имеется в ранее включенном файле, то используется более позднее определение.

Рассмотрим примеры.

- если требуется отключить запуск или изменить режим запуска какого-либо устройства, следует создать файл /etc/system/enum/overrides и добавить в него запись device(...) для этого устройства:

```
device(pci, ven=1234, dev=2000)
device(pci, ven=1234, dev=2001)
   requires( $(IONET CMD), )
   uniq(netnum, devn-en, 0)
   mount(-Tio-net        /lib/dll/devn-pcnet.so,        "/dev/io-net/en$(netnum)")
```

```
device(pci, ven=1234, dev=2002)
device(pci, ven=1234, dev=2003)
```

Если программа распознавания обнаруживает устройства 2000 и 2001 производителя 1234, то первый фрагмент этого кода задает следующие действия:

- запустить драйвер io-pkt*. IONET_CMD представляет собой макрос, который определен в файле /etc/system/enum/include/net, и задает командную строку драйвера io-pkt* по умолчанию;
- записать в параметр netnum следующий по счету уникальный номер устройства сетевого интерфейса, начиная с 0;
- смонтировать драйвер PCNET в диспетчер устройств io-pkt*.

Второй фрагмент кода указывает программе распознавания не выполнять никаких действий при обнаружении устройств 2002 и 2003 производителя 1234.

Примечание. При добавлении записей device, которые отключают распознавание устройств, следует убедиться в том, что после этих записей отсутствуют какие-либо операторы действий. Группы операторов действий, которые следуют за одной или несколькими записями устройств, применяются к этим устройствам. Записи устройств, распознавание которых требуется отключить, следует помещать в конец конфигурационного файла overrides.

- чтобы изменить режим запуска протокола TCP/IP программой распознавания, необходимо переопределить основную команду драйвера io-pkt*, которая задана в файле /etc/system/enum/include/net. По умолчанию она выглядит следующим образом:

```
io-pkt-v4-hc -ptcpip
```

Если требуется включить протокол IPSec, в файл overrides нужно добавить следующий код:

```
all
     set(IOPKT_CMD, io-pkt-v4-hc -ptcpip ipsec)
```

Специфичные для системы программы распознавания

Для того чтобы точнее настроить программы распознавания в соответствии с конфигурацией системы, можно создать каталог /etc/host_cfg/$HOSTNAME/system/enum. Если эта структура каталогов существует, сценарий rc.devices указывает программе распознавания считывать конфигурационные файлы из нее, а не из каталога /etc/system/enum.

Примечание. Даже если каталог /etc/host_cfg/$HOSTNAME/system/enum существует, программа распознавания выполняет поиск каталога oem и файла overrides в каталоге /etc/system/enum.

Каталог /etc/host_cfg/$HOSTNAME/system/enum можно легко сконфигурировать следующим образом: скопировать каталог /etc/system/enum (в том числе все его подкаталоги) в каталог /etc/host_cfg/$HOSTNAME/system, а затем приступить к настройке.

8.7. Файл /etc/rc.d/rc.sysinit

Сценарий /etc/system/sysinit запускает сценарий /etc/rc.d/rc.sysinit для локальной инициализации системы (рис. 8.4).

Рис. 8.4. Выполнение инициализации с помощью сценария /etc/rc.d/rc.sysinit

Сценарий rc.sysinit выполняет следующие действия.

- запускает безопасный генератор случайных чисел random, который создает случайные числа для шифрования и других задач.
- если каталог /var/dumps существует, то сценарий rc.sysinit запускает утилиту dumper для записи дампов процессов, которые завершились ненормально, в каталог /var/dumps.

- если файл /etc/host_cfg/$HOSTNAME/rc.d/rc.local существует и является исполняемым, то сценарий rc.sysinit запускает его. В противном случае сценарий rc.sysinit запускает файл /etc/rc.d/rc.local, если он существует и является исполняемым. У этого файла не существует версии по умолчанию, поэтому при необходимости ее нужно создать вручную. Более подробные сведения см. в подразд. "Файл rc.local" далее в этом разделе.
- наконец, сценарий rc.sysinit запускает программу tinit. По умолчанию система запускает графическую оболочку Photon, но в случае если создан файл /etc/system/config/nophoton, сценарий rc.sysinit указывает утилите tinit использовать текстовый режим. Более подробные сведения см. в разд. "Утилита tinit" далее в этом разделе.

8.8. Файл rc.local

Как описано ранее, сценарий rc.sysinit запускает файл /etc/host_cfg/$HOSTNAME/rc.d/rc.local или /etc/rc.d/rc.local, если он существует и является исполняемым.

С помощью файла rc.local можно конфигурировать запуск системы:

- запускать дополнительные программы;
- подключать библиотеки к работающим процессам.

Файл rc.local также позволяет уничтожать работающие процессы с помощью утилиты slay и перезапускать их с другими параметрами, однако такой подход является громоздким. Вместо него для запуска процессов с требуемыми параметрами следует изменить распознавание устройств. Более подробные сведения см. в разд. "Распознавание устройств" ранее в этом разделе.

С помощью файла rc.local пользователь может, к примеру:

- запустить NFS- или CIFS-клиента для монтирования удаленной файловой системы;
- запустить утилиту inetd, чтобы обеспечить пользователям других компьютеров доступ к своему компьютеру (см. раздел 13);

- запустить утилиту lpd и/или отдельный экземпляр утилиты spooler для поддержки печати (см. раздел 14);
- пропустить приглашение для входа в систему при загрузке графической оболочки Photon, добавив строку:

/usr/photon/bin/Photon -l '/usr/photon/bin/phlogin -O -Uпользователь:пароль'

Следует иметь в виду, что пароль необходимо поместить в файл rc.local в виде простого текста, однако нужно сказать, что пользователь, который желает пропустить приглашение для входа в систему, вероятно, не заботится о ее безопасности.

Параметр -O утилиты phlogin переводит систему в текстовый режим после завершения сеанса графической оболочки Photon. При отсутствии параметра -O нажатие комбинации клавиш <Ctrl>+<Shift>+<Alt>+<Backspace> приводит к повторному входу в систему.

В качестве альтернативы можно задать запуск графической оболочки Photon командой ph в файле .profile, а затем добавить в файл rc.local следующую команду:

```
login -f имя_пользователя
```

Более подробные сведения см. в описании login руководства "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01.

Не следует задавать переменные окружения в сценарии rc.local, поскольку после его выполнения запускается другой командный интерпретатор, и к моменту, когда пользователю предлагается войти в систему, все переменные окружения, которые определены в файле rc.local, перестают существовать.

Примечание. После создания файла rc.local следует с помощью команды chmod +x rc.local установить его атрибут разрешения на исполнение (executable bit).

8.9. Утилита tinit

Программа tinit инициализирует терминал следующим образом:

- если указан параметр -p, она запускает графическую оболочку Photon.

- в противном случае программа tinit считывает файл /etc/config/ttys и запускает утилиту login или командные интерпретаторы в соответствии с его содержанием.

8.10. Обновление драйверов диска

В процессе начальной загрузки ЗОСРВ «Нейтрино» можно динамически добавлять драйверы блочного ввода/вывода (т. е. драйверы диска), что дает возможность загружаться в системе с новыми контроллерами.

Под обновлением драйвера понимается обновление собственно драйвера (только devb-*) и добавление простого конфигурационного файла. Конфигурационный файл состоит из строк обычного текста (с окончаниями строк в стиле DOS или UNIX) следующего формата:

```
drvr_name|type|timeout|add_args
```

Первые три поля являются обязательными. Поля имеют следующее назначение:

- drvr_name — имя файла драйвера;
- type — строка, отображаемая в процессе загрузки, когда очередь доходит до драйвера;
- timeout — полное время ожидания устройств;
- add_args — любые дополнительные аргументы, необходимые драйверу (например, blk cache=512k).

Конфигурационному файлу может быть дано имя drivers.cfg, и вы должны добавить обновления на физический носитель, в настоящее время это CD-ROM или USB-диск. В процессе начальной загрузки вначале просматривается корневая область файловой системы, а затем каталог с именем qnxdrvr. Это может помочь уменьшению перегруженности корневой зоны файловой системы.

Исходной может быть любая из поддерживаемых файловых систем. Известно, что работают такие файловые системы:

- стандартные файловые системы ISO9660 на CD-ROM;

- файловые системы DOS (раздел t7) и QNX 4 (раздел t79) на USB-диске.

Если обновление размещено где-то в Интернете в формате zip или tar с сохраненной структурой qnxdrvr, то конечный пользователь просто должен загрузить архив, развернуть его на USB-диск и вставить USB-диск на этапе загрузки.

Применить обновленные версии драйверов можно, нажав клавишу <Пробел> в процессе загрузки, а затем клавишу <F2>. После этого завершится этап начальной загрузки в систему стандартных блочных драйверов, и в исходную файловую систему могут быть загружены обновления. Вам будет выдана подсказка о выборе файловой системы и будет предложено вставить носитель с обновлениями.

Примечание. Если необходимо повторно просканировать разделы (например, для поиска USB-диска, который вы вставили после начала процесса загрузки), нажмите клавишу <F12>.

Использование такого механизма дает возможность обновить существующие драйверы или просто модифицировать их аргументы (например, спецификацию PCI ID).

Если идет процесс инсталляции, то программа производит копирование обновленных драйверов в каталог /sbin, а конфигурационного файла — в каталог /boot/sys. Одновременно производится также копирование стандартных файлов компоновки в каталог /boot/build (кроме файлов, имеющих отношение к многоядерности), файлам присваиваются имена qnxbase-drvrup.build и basedma-drvrup.build. Эти файлы затем используются для создания новых файлов образа с именами qnxbase-drvrup.ifs и basedma-drvrup.ifs в папке /boot/fs. DMA-версия нового файла копируется в каталог /.boot, не-DMA-версия — в каталог /.altboot.

Примечание. Программа инсталляции не выполняет перекомпоновку многоядерных (SMP) образов.

8.11. Устранение неполадок

Рассмотрим некоторые проблемы, связанные с конфигурированием запуска системы.

Приложения, которые указаны в файле rc.local, не запускаются. Следует убедиться в том, что:

- файл rc.local является исполняемым, и при необходимости сделать его исполняемым с помощью команды chmod:

```
chmod +x /etc/rc.d/rc.local
```

- исполняемый файл находится в каталоге, который включен в определение переменной окружения **PATH** на момент исполнения сценария /etc/rc.d/rc.local.

Мой файл rc.local испорчен, и теперь я не могу загрузить систему. Вам необходимо:

- загрузиться с компакт-диска и откорректировать файл rc.local

или:

- загрузить систему в режиме "отладочной оболочки" ("debug shell"). Для этого во время загрузки нажмите клавишу <Пробел>, а затем клавишу <F5>, запустится отладочная оболочка.

Когда вы войдете в отладочную оболочку (fesh), введите команду exit, а затем дождитесь второй подсказки от оболочки. Далее введите команду:

```
export PATH=/bin:/usr/bin:/sbin:/usr/sbin
```

Вы можете откорректировать ваш файл rc.local или убрать его с пути загрузки, чтобы загрузка шла без его участия:

```
cd /etc/rc.d
cp rc.local rc.local.bad
rm rc.local
```

9. Настройка среды

Предыдущий раздел описывает процесс запуска и возможности настройки операционной системы. В этом разделе обсуждаются возможности конфигурирования среды, в которую пользователь попадает после входа в систему, а также некоторые процедуры настройки, которые могут оказаться необходимыми.

9.1. Что происходит при входе в систему?

Перед тем как приступить к конфигурированию начальной среды, пользователь должен понять, что происходит при входе в систему, поскольку расположение параметров, которые требуется задать, зависит от характера конфигурирования. Пользователю следует продумать следующие вопросы:

- должно ли это изменение применяться ко всем пользователям или только к текущему пользователю?
- требуется ли выполнять некоторые действия только при первом входе в систему или при каждом запуске командного интерпретатора?

При входе в систему запускается начальный командный интерпретатор, который указан в записи пользователя в базе данных учетных записей (см. подразд. "Файл /etc/passwd" раздела 3). Как правило, начальным командным интерпретатором является sh, который обычно представляет собой ссылку на командный интерпретатор Korn ksh.

Когда ksh запускается как начальный командный интерпретатор, он исполняет следующие профили, если они существуют и являются исполняемыми:

- /etc/profile;
- $HOME/.profile.

Для чего нужны два профиля? Настройки, которые применяются ко всем пользователям, хранятся в профиле /etc/profile, а настройки, которые соответствуют конкретному пользователю – в его собственном профиле .profile. Редактировать файл /etc/profile может только пользователь root.

В действительности существует третий профиль, который предназначен для командного интерпретатора. Его особенность заключается в том, что он исполняется при каждом запуске командного интерпретатора (см. разд. "Файл запуска ksh" далее в этом разделе).

9.2. Настройка домашнего каталога

Домашний каталог (home directory) позволяет хранить все файлы и каталоги, которые относятся к конкретному пользователю. В домашнем каталоге удобно содержать пользовательские двоичные файлы и сценарии. Запись, которая соответствует пользователю в базе данных паролей, определяет домашний каталог (см. разд. "Файл /etc/passwd" раздела 3), а переменная окружения **HOME** содержит его имя.

В домашнем каталоге также хранится информация, которая используется для конфигурирования среды при входе пользователя в систему. По умолчанию приложения устанавливают в домашний каталог конфигурационные файлы. Конфигурационные файлы обычно начинаются с точки (.) и запускаются при входе пользователя в систему (например, файл .profile) или при запуске приложения (например, файл .jedrc).

Приложения графической оболочки Photon представляют собой особый случай. Приложения, которые выполняются в графической оболочке Photon, как правило, хранят свои конфигурации в каталоге $HOME/.ph. Если необходимо автоматически запускать какие-либо приложения вместе с графической оболочкой Photon, следует поместить соответствующие команды в файл $HOME/.ph/phapps.

9.3. Настройка командного интерпретатора

Существует множество конфигурационных файлов среды. В этом разделе описано несколько наиболее полезных из них:

- /etc/profile;
- $HOME/.profile;
- файл запуска командного интерпретатора ksh.

Файл /etc/profile

Начальный командный интерпретатор исполняет файл /etc/profile, если этот файл существует и является исполняемым. Файл /etc/profile задает настройки командного интерпретатора, которые применяются ко всем пользователям, поэтому его содержимое представляет интерес для системных администраторов. Чтобы отредактировать файл /etc/profile, необходимо войти в систему как пользователь root.

Файл /etc/profile выполняет следующие задачи:

- присваивает значения переменным окружения **HOSTNAME**, **PROCESSOR** и **SYSNAME**, если они еще не заданы;
- добавляет надлежащие каталоги в переменную окружения **PATH** (переменная **PATH** пользователя root включает в себя каталоги, которые содержат системные исполняемые файлы, например, /sbin);
- задает маску файловых разрешений (umask) (см. подразд. "Владение файлами и права доступа" раздела 6);
- отображает дату входа пользователя в систему, "сообщение дня", которое найдено в файле /etc/motd, и дату последнего входа в систему;
- присваивает переменной окружения **TMPDIR** значение /tmp, если оно еще не задано;
- запускает все сценарии в каталоге /etc/profile.d с помощью команды "точка" (т. е. эти сценарии запускаются не в отдельных экземплярах командного интерпретатора, а выполняются в текущем командном интерпретаторе). Более подробные сведения о команде "точка" см. в ". (dot) builtin command" в документации командного интерпретатора ksh в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

Если необходимо запускать определенный сценарий при каждом запуске начального командного интерпретатора любым пользователем системы, следует поместить этот сценарий в каталог /etc/profile.d. Чтобы добавить файл в этот каталог, требуются привилегии уровня root.

Например, если при входе любого пользователя в систему необходимо задавать глобальные переменные окружения или запускать определенные задачи, следует поместить в каталог /etc/profile.d сценарий, который выполняет эти действия. Если в качестве начального командного интерпретатора используется sh, то необходимо, чтобы сценарий имел расширение .sh.

Файл $HOME/.profile

Файл $HOME/.profile запускается при каждом входе пользователя в систему после выполнения файла /etc/profile. Если в файл .profile вносятся изменения, то они вступают в силу только при следующем входе пользователя в систему.

В файле .profile следует задавать настройки, которые выполняются однократно или должны наследоваться всеми командными интерпретаторами.

Например, с помощью файла .profile можно:

- задавать переменные окружения (см. разд. "Переменные окружения" далее в этом разделе);
- запускать любые необходимые команды;
- задавать маску файловых разрешений (см. подразд. "Владение файлами и права доступа" раздела 6).

Примечание. Псевдонимы следует создавать в профиле командного интерпретатора пользователя (см. следующий раздел), а не в файле .profile, поскольку командный интерпретатор не экспортирует псевдонимы. Если задать псевдоним в файле .profile, то он будет установлен только в командных интерпретаторах, которые запускаются как начальные с помощью параметра -l.

Не следует запускать приложения графической оболочки Photon из сценария .profile, поскольку при его выполнении графическая оболочка Photon не работает. Photon-приложения следует запускать из файла $HOME/.ph/phapps.

Файл запуска ksh

Как описано ранее, начальный командный интерпретатор выполняет определенные профили. Кроме того, можно создать профиль, который будет выполняться командным интерпретатором ksh при запуске другого командного

интерпретатора независимо от того, является ли командный интерпретатор ksh начальным.

У этого профиля нет определенного имени; при запуске командного интерпретатора ksh он проверяет переменную окружения ENV. Если эта переменная существует, то командный интерпретатор ksh считывает из нее имя профиля. Чтобы задать значение переменной ENV, следует добавить в файл $HOME/.profile строку, подобную следующей:

```
export ENV=$HOME/.kshrc
```

Пользователи часто называют упомянутый профиль именем .kshrc, но ему можно присвоить любое другое имя. Этот файл не обязан быть исполняемым.

В профиле командного интерпретатора ksh можно задавать желаемые псевдонимы и выполнять другие действия. Например, если требуется, чтобы утилита ls всегда отображала символы, которые указывают тип файла (исполняемый файл, каталог или ссылку), следует добавить в профиль командного интерпретатора следующую строку:

```
alias ls="ls -F"
```

Любые изменения, которые вносятся в профиль, применяются к новым командным интерпретаторам, но не к их существующим экземплярам.

9.4. Переменные окружения

Многие приложения управляют своими действиями с помощью переменных окружения. Например, утилита less считывает ширину терминала или окна из переменной окружения COLUMNS, а многие утилиты записывают временные файлы в каталог, который определяется переменной окружения TMPDIR.

Запускаемый процесс наследует копию окружения своего родителя. Это означает, что если задать переменную окружения в файле .profile, то все командные интерпретаторы и процессы унаследуют ее при условии, что определение этой переменной не будет отменено каким-либо процессом в цепочке.

Например, если пользователю принадлежит каталог с именем bin, этот каталог можно включить в переменную окружения PATH, добавив в файл .profile следующую строку:

```
export PATH=$PATH:/home/имя_пользователя/bin
```

Если системный администратор захочет применить это изменение ко всем пользователям, ему будет необходимо экспортировать переменные окружения из файла /etc/profile или из сценария в каталог /etc/profile.d. Более подробную информацию см. в разд. "Файл /etc/profile" ранее в этом разделе.

Задание переменных PATH и LD_LIBRARY_PATH

Утилита login не сохраняет переменные окружения за исключением нескольких особых переменных, например PATH и TERM.

Переменная окружения **PATH** определяет пути для поиска команд, а переменная **LD_LIBRARY_PATH** — пути для поиска компоновщиком совместно используемых библиотек.

Начальные значения по умолчанию переменных окружения **LD_LIBRARY_PATH** и **PATH** заданы в файле построения образа до запуска модуля procnto. Две конфигурационные строки (см. разд. "Конфигурационные строки" далее в этом разделе), _CS_PATH и _CS_LIBPATH, принимают значения по умолчанию переменных PATH и LD_LIBRARY_PATH. Утилита login задает переменную окружения PATH с помощью строки _CS_PATH и передает переменную PATH, а также обе конфигурационные строки, своим дочерним процессам.

Если ввести команду set или env в командном интерпретаторе, который был запущен утилитой login, переменная окружения PATH будет отображена, а переменная LD_LIBRARY_PATH — нет. Строка _CS_LIBPATH используется для задания переменной LD_LIBRARY_PATH аналогичным образом.

Чтобы указать утилите login, какие переменные окружения следует сохранять, нужно воспользоваться файлом /etc/default/login. Этот файл можно отредактировать, добавив в него новые переменные, например

LD_LIBRARY_PATH, но изменять существующие переменные, такие как PATH и TERM, нельзя.

Если в качестве начального командного интерпретатора используется ksh, можно переопределить существующие переменные и задать новые переменные, отредактировав профили /etc/profile и $HOME/.profile. Все переменные окружения, которые заданы в файле /etc/profile, заменяют определения файла /etc/default/login, а содержимое файла $HOME/.profile заменяет определения обоих файлов /etc/default/login и /etc/profile.

Более подробные сведения о конфигурационных строках см. в следующем разделе.

9.5. Конфигурационные строки

В дополнение к переменным окружения операционная система ЗОСРВ «Нейтрино» использует конфигурационные строки (configuration strings). Конфигурационные строки представляют собой системные переменные, которые похожи на переменные окружения, но более динамичны.

При задании переменной окружения ее новое значение действует только в текущем экземпляре командного интерпретатора и процессах, которые созданы им после задания переменной. Задание нового значения конфигурационной строки начинает действовать во всей системе немедленно.

> **Примечание**. Операционная система ЗОСРВ «Нейтрино» также поддерживает настройку конфигурационных лимитов (configurable limits) — переменных, которые хранят информацию о системе. Более подробные сведения см. в разделе 21.

Для того чтобы получить значение конфигурируемого лимита или конфигурационной строки, можно воспользоваться POSIX-утилитой getconf. Операционная система ЗОСРВ «Нейтрино» также определяет утилиту setconf, которая отсутствует в стандарте POSIX и позволяет задавать определенные

конфигурационные строки пользователю root. Чтобы считать значение конфигурационной строки в программе, следует вызвать функцию confstr().

Имена конфигурационных строк начинаются с _CS_ и состоят из символов верхнего регистра, хотя утилиты getconf и setconf позволяют использовать любой регистр, опускать первый символ подчеркивания или весь префикс при условии, что остальная часть имени является однозначной.

С помощью утилиты setconf можно задавать значения следующих конфигурационных строк:

- _CS_ARCHITECTURE — название архитектуры системы команд;
- _CS_DOMAIN — сетевой домен этого узла;
- _CS_HOSTNAME — сетевое имя этого узла;

При изменении конфигурационной строки _CS_HOSTNAME необходимо также изменять переменную окружения **HOSTNAME**. Утилита hostname всегда возвращает значение конфигурационной строки _CS_HOSTNAME.

- _CS_HW_SERIAL — серийный номер оборудования;
- _CS_HW_PROVIDER — название производителя оборудования;
- _CS_LIBPATH — путь по умолчанию для размещения разделяемых объектов. Более подробные сведения см. в разд. "Задание переменных PATH и LD_LIBRARY_PATH" ранее в этом разделе;

_CS_LOCALE — название локали (набор региональных настроек, включающий в себя такие параметры, как часовой пояс, единицы измерения веса, наименование валюты и т. п.) (locale string);

- _CS_MACHINE — тип оборудования, на котором работает операционная система;
- _CS_PATH — строка по умолчанию для поиска системных утилит. Более подробные сведения см. в разд. "Задание переменных PATH и LD_LIBRARY_PATH" ранее в этом разделе;
- _CS_RELEASE — текущий номер выпуска (релиз) операционной системы;

- _CS_RESOLVE — версия файла /etc/resolv.conf (исключая доменное имя), хранимая в оперативной памяти;

- _CS_SRPC_DOMAIN — защищенный RPC-домен (Remote Procedure Call, удаленный вызов процедур);

- _CS_SYSNAME — название операционной системы;

- _CS_TIMEZONE — источник информации о часовых поясах, альтернативный переменной окружения TZ. Более подробные сведения см. в разд. "Задание часового пояса" далее в этом разделе.

- _CS_VERSION — версия операционной системы.

9.6. Задание часового пояса

Самый простой способ задать часовой пояс в графической оболочке Photon — воспользоваться утилитой phlocale. Достаточно просто выбрать нужный часовой пояс, а программа phlocale выполнит все остальные действия.

Если графическая оболочка Photon не используется, то необходимо задать переменную **TZ** или конфигурационную строку _CS_TIMEZONE. Чтобы часовой пояс устанавливался при загрузке компьютера, требуется поместить эту же информацию в файл /etc/TIMEZONE (см. описание файла /etc/system/sysinit в разделе 8).

Примечание. Если переменная окружения **TZ** не установлена, система использует значение конфигурационной строки _CS_TIMEZONE. Переменная TZ включена в стандарты POSIX, а конфигурационная строка _CS_TIMEZONE реализована в операционной системе ЗОСРВ «Нейтрино». Описание, которое приведено далее, относится и к переменной окружения TZ, и к конфигурационной строке _CS_TIMEZONE.

Различные функции, которые работают со временем, используют информацию о часовых поясах для вычисления времени относительно пояса UTC (Universal Time Coordinated, всеобщее скоординированное время, которое раньше называлось Greenwich Mean Time (GMT — время по Гринвичу)).

Как правило, время на компьютере устанавливается в соответствии с часовым поясом UTC. Если оборудование компьютера не сохраняет время автоматически, следует воспользоваться командой date.

Переменную окружения **TZ** можно задать с помощью утилиты env или команды командного интерпретатора export. Конфигурационная строка _CS_TIMEZONE задается с помощью утилиты setconf. Пример:

```
env TZ=PST8PDT
export TZ=PST8PDT
setconf _CS_TIMEZONE PST8PDT
```

Переменная окружения **TZ** и конфигурационная строка _CS_TIMEZONE имеют следующий формат (пробелы использованы только для удобочитаемости):

```
std offset dst offset, rule
```

Расширенный формат имеет вид:

```
stdoffset[dst[offset][,start[/time],end[/time]]]
```

Здесь:

- std и dst — три или более буквы, которые обозначают поясное или летнее время. Обязательным является только параметр std. Если опустить параметр dst, летнее время не будет применяться в этой локали. Разрешается использовать буквы как в верхнем, так и в нижнем регистре. Можно применять любые символы за исключением двоеточия (:) в качестве первого символа, а также цифр, запятой (,), минуса (-), плюса (+) и нуль-символа стандарта ASCII (\0);
- offset — величина, которую требуется прибавить к местному времени, чтобы получить всеобщее скоординированное время (UTC). Параметр offset имеет следующую форму:

```
hh[:mm[:ss]]
```

Минуты (mm) и секунды (ss) являются необязательными. Необходимо задавать лишь часы (hh), которые могут быть указаны в виде одной цифры.

Параметр offset, следующий за параметром std, является обязательным. Если параметр offset после параметра dst опущен, то считается, что летнее время находится на один час впереди поясного.

При задании элементов параметра offset можно использовать одну или несколько цифр. Значения всегда интерпретируются как десятичные числа. Часы могут находиться в пределах от 0 до 24, а минуты и секунды, если они указаны — от 0 до 59. Если перед часовым поясом указан символ "-", то часовой пояс находится восточнее нулевого меридиана, иначе – западнее (в этом случае можно указать необязательный символ "+").

- rule — определяет моменты перехода на летнее время и обратно. Параметр rule имеет следующую форму:

```
date/time,date/time
```

где первый элемент date задает дату перехода с поясного времени на летнее, а второй элемент date — дату перехода с летнего времени на поясное. Поля time определяют время перехода с одного режима времени на другой по текущему местному времени.

Элемент date может иметь один из следующих форматов:

- Jn — юлианский день n (1 x n x 365). Дни 29 февраля не учитываются. Таким образом, в каждом году, в том числе в високосном, 28 февраля считается днем 59, а 1 марта — днем 60. Этот формат не позволяет явно указать дату 29 февраля;
- n — юлианский день с отсчетом от нуля (0 x n x 365). Високосные годы учитываются и можно указать дату 29 февраля;
- Mm.n.d — день d (0 x d x 6) недели n месяца m текущего года (1 x n x 5, 1 x m x 12; неделя 5 означает "последний день недели d в месяце m", который может находиться в четвертой или пятой неделе). Неделя 1 — это первая неделя, в которой имеется день недели d. Днем с номером 0 является воскресенье.

Форматы элементов time и offset совпадают за исключением того, что в элементе time не разрешается указывать первый знак ("+" или "-"). Если поле time опущено, то используется значение по умолчанию 02:00:00.

Ограничения

Утилита phlocale список часовых поясов берет из /etc/timezone/uc_tz_t, но его соответствие текущему делению планеты на часовые пояса не гарантируется; часовые пояса регулируются местным законодательством стран и могут отличаться от указанных в данном файле. Аббревиатуры, обозначающие часовые пояса в данном файле, так же не являются стандартными и могут неоднозначно идентифицировать часовой пояс.

Примеры

В этом разделе рассматривается несколько примеров настроек часовых поясов.

Московское время

Определение московского времени выглядит следующим образом:

```
MSK-4
```

- московское время на 4 часа опережает всеобщее скоординированное время;
- в этой локали не применяется летнее время.

Красноярское время

Определение часового пояса Красноярска выглядит следующим образом:

```
KRAT-8
```

- красноярское время на 8 часов опережает всеобщее скоординированное время;
- в этой локали не применяется летнее время.

Восточное время

Восточное время является часовым поясом по умолчанию. Его краткое определение имеет следующий вид:

```
EST5EDT
```

- восточное поясное время на 5 часов опережает всеобщее скоординированное время (UTC). К этой локали применяется как поясное, так и летнее время;
- по умолчанию восточное летнее время (Eastern Daylight Time, EDT) находится на один час впереди поясного (EDT4);

- если даты перехода на летнее время и обратно не указаны, то летнее время начинается в 2 часа ночи первого воскресенья апреля и заканчивается в 2 часа ночи последнего воскресенья октября.

Полное определение выглядит следующим образом:

```
EST5EDT4,M4.1.0/02:00:00,M10.5.0/02:00:00
```

В этом определении явно указано, что летнее время начинается в 2 часа ночи первого (1) воскресенья (0) апреля (4) и заканчивается в 2 часа ночи последнего (5) воскресенья (0) октября (10).

Центральноевропейское время

Определение центральноевропейского времени выглядит следующим образом:

```
Central Europe Time-2:00
```

- центральноевропейское поясное время на 2 часа отстает от всеобщего скоординированного времени;
- в этой локали не применяется летнее время.

Использование часовых поясов в программировании

В программе переменной окружения **TZ** можно присвоить значение, вызвав функцию setenv() или putenv():

```
setenv( "TZ", "PST08PDT07,M3.2.0/2,M11.1.0/2", 1 );
putenv( "TZ=PST08PDT07,M3.2.0/2,M11.1.0/2" );
```

Чтобы считать значение переменной **TZ**, следует воспользоваться функцией getenv():

```
char *tzvalue;
...
tzvalue = getenv( "TZ" );
```

Значение конфигурационной строки _CS_TIMEZONE можно считать, вызвав функцию confstr():

```
confstr(_CS_TIMEZONE, buff, BUFF_SIZE);
```

или задать с помощью оператора

```
confstr(_CS_SET | _CS_TIMEZONE, "JST-9", 0);
```

Функция tzset() считывает текущее значение переменной окружения **TZ** или конфигурационной строки _CS_TIMEZONE, если значение переменной TZ не определено, и задает следующие глобальные переменные:

- daylight — показывает, поддерживается ли летнее время в данной локали;
- timezone — разница во времени между текущим часовым поясом и всеобщим скоординированным временем, выраженная в секундах;
- tzname — вектор, который состоит из двух указателей на символьные строки, содержащие имена часовых поясов поясного и летнего времени.

При вызове функции ctime(), ctime_r(), localtime() или mktime() библиотека присваивает значение переменной tzname так же, как и при вызове функции tzset(). Это же действие выполняется, если вызывается функция strftime() с директивой %Z.

9.7. Настройка графической оболочки Photon

Автоматический запуск приложений

Если необходимо, чтобы при каждом запуске графической оболочки Photon запускалось какое-либо Photon-приложение, следует поместить его в файл $HOME/.ph/phapps. Каждая команда должна располагаться на отдельной строке. Например, для запуска редактора графической оболочки Photon вместе с графической оболочкой Photon следует ввести строку:

```
ped &
```

Примечание. Поскольку этот файл не является сценарием командного интерпретатора, в нем не следует задавать какие-либо переменные окружения.

Настройка шрифтов

Графическая оболочка Photon поддерживает широкий набор типов шрифтов. В среде Photon должны работать любые шрифты стандарта Unicode.

Файлы шрифтов системы хранятся в каталоге /usr/photon/font_repository. Этот каталог содержит следующие файлы:

- *.phf — файлы шрифтов графической оболочки Photon. Эти файлы включают в себя растровые шрифты. Каждый файл содержит информацию о шрифте одного размера и стиля;
- *.TTF, *.ttf — файлы шрифтов TrueType;
- *.pfr — файлы переносимых шрифтовых ресурсов (Portable Font Resource, PFR) формата TrueDoc компании Bitstream, которые содержат хинтованные масштабируемые определения шрифтов. Каждый файл может включать в себя множество шрифтов и стилей. Эта более старая технология, которая поддерживается для обеспечения совместимости;
- fontdir — каталог известных шрифтов. Каждая запись этого файла содержит имя и тип шрифта, его размер и стиль, текстовое описание семейства шрифтов и диапазон символов, которые определены в шрифте. В этом конфигурационном файле должен быть определен, как минимум, один шрифт, чтобы быть доступным приложению. Записи файла fontdir являются статическими; их нельзя загружать динамически;
- fontext — набор правил расширения для обработки пропущенных (отсутствующих) символов;
- fontmap — правила подстановки шрифтов систем. Подробные сведения о формате этого файла см. в описании phfont;
- fontopts — командно-строковые параметры, которые используются для вызова соответствующего сервера шрифта, по одному параметру в строке.

Чтобы установить в систему новый шрифт, следует поместить файлы шрифтов в каталог font_repository и запустить утилиту mkfontdir для создания нового файла fontdir. Затем нужно перезапустить менеджер шрифтов, который обычно представлен разделяемым объектом, загруженным в io-graphics. При использовании отдельного сервера phfont его так же следует перезапустить.

Возможности ввода текста

Графическая оболочка Photon позволяет вводить текст на китайском, японском и корейском языках. Чтобы воспользоваться этими возможностями, следует запустить утилиту cpim (ввод на китайском языке), vpim (ввод на японском языке) или kpim (ввод на корейском языке). Символы этих языков можно вводить с обычной клавиатуры в любое приложение, которое принимает обычный текст. Более подробные сведения см. в комплекте документации "Photon Multilingual Input".

9.8. Типы терминалов

Чтобы указать тип используемого терминала консоли или утилите pterm, следует задать переменную окружения TERM. Каталоги, которые содержат базу данных терминалов, расположены в каталоге /usr/lib/terminfo. Для изменения правил подстановки шрифтов в базе данных можно воспользоваться утилитами tic и infocmp.

Например, если запустить утилиту infocmp для /usr/lib/terminfo/q/qansi-m, то для этой базы данных будет сгенерирован файл в текстовой форме, доступный для чтения человеком. Затем этот файл можно модифицировать и снова скомпилировать с помощью утилиты tic в формат базы данных. Файл /etc/termcap обеспечивает совместимость с программами, которые используют более раннюю однофайловую модель базы данных, а не новую библиотечную модель.

9.9. Устранение неполадок

Далее рассмотрены некоторые распространенные проблемы, которые могут возникнуть при конфигурировании среды.

- сценарий, который помещен в файл /etc/profile.d, не запускается.

Следует убедиться в том, что:

- имя сценария имеет расширение .ksh или .sh;
- атрибут исполнения сценария установлен;
- сценарий начинается со строки:

```
#! /bin/sh
```

- как установить время таким образом, чтобы оно было правильным одновременно в ЗОСРВ «Нейтрино» и в Microsoft Windows?

Если на одном разделе компьютера установлена операционная система Windows, а на другом – ЗОСРВ «Нейтрино», то установка часов в одной операционной системе изменяет время в другой операционной системе.

В ЗОСРВ «Нейтрино» аппаратные часы обычно устанавливают по всеобщему скоординированному времени (Universal Time Coordinated, UTC), а затем задают часовой пояс. В Windows аппаратные часы устанавливаются по местному времени.

Чтобы задать время, которое является корректным для обеих операционных систем, следует установить на аппаратных часах местное время, находясь в ЗОСРВ «Нейтрино». Более подробные сведения см. в описании файла /etc/system/sysinit в разделе 8.

- каким образом правильно проверить, что файл .kshrc выполняется как сценарий, а не как сеанс терминала?

Если параметр i задан, то файл .kshrc работает в интерактивном режиме. Следующий код проверяет, задан ли этот параметр:

```
case $- in
*i*)
set -o emacs
export EDITOR=vi
export VISUAL=vi
export PS1='`hostname -s`:`/bin/pwd` >'
bind ^[[z=list
bind ^I=complete
...
esac
```

Параметр $- является объединением всех односимвольных параметров, которые переданы сценарию. Более подробные сведения см. в разделе «Параметры» к описанию ksh в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

10. Написание сценариев командного интерпретатора

10.1. Что такое сценарий?

Создание сценариев командного интерпретатора по существу представляет собой помещение последовательности команд, которые можно ввести в командной строке в файл для того, чтобы выполнить эти команды в будущем или многократно запускать их без повторного ввода.

Сценарии можно использовать для автоматизации повторяющихся задач и решения сложных задач, которые трудно выполнить без многократных попыток и/или внесения изменений в код. Примеры сценариев:

- файл /etc/config/sysinit, который запускается при загрузке компьютера с операционной системой ЗОСРВ «Нейтрино» (см. раздел 8);
- файл /usr/bin/ph, который запускает графическую оболочку Photon (см. раздел 5).

10.2. Доступные командные интерпретаторы

В операционной системе ЗОСРВ «Нейтрино» чаще всего используется командный интерпретатор ksh, который является общедоступной реализацией командного интерпретатора Korn. Команда sh обычно представляет собой символьную ссылку на командный интерпретатор ksh. Более подробные сведения об этом интерпретаторе см. в:

- разделе 4;
- описании утилиты ksh в справочнике по утилитам.

Операционная система ЗОСРВ «Нейтрино» также предоставляет или использует несколько других сред исполнения сценариев.

- файл построения образа операционной системы включает в себя раздел файла сценария, который помечен атрибутом +script. Утилита mkifs выполняет разбор этого сценария, однако его исполнение осуществляет модуль procnto на этапе загрузки операционной системы. Модуль procnto обеспечивает очень

простую среду исполнения сценариев, которая позволяет выполнять последовательности команд и обладает примитивными возможностями синхронизации;

- командный интерпретатор esh обеспечивает среду исполнения простых сценариев во встроенных системах, где полный функциональный набор интерпретатора ksh является излишним. Встроенный командный интерпретатор esh поддерживает исполнение утилит, простое перенаправление, удлинение файловых имен, псевдонимы и действия над окружением;

- расширенный командный интерпретатор fesh, как и esh, предоставляет ограниченную среду исполнения сценариев, однако поддерживает дополнительные встроенные команды для часто используемых утилит, что позволяет сэкономить ресурсы, не включая эти утилиты во встраиваемую систему. Командный интерпретатор fesh включает в себя встроенные аналоги команд cp, df, ls, mkdir, rm и rmdir, однако в большинстве случаев эти аналоги обеспечивают только базовые функции соответствующих утилит и не являются полными;

- python представляет собой мощный объектно-ориентированный язык, который может быть использован для обработки файлов, манипуляций со строками, синтаксического разбора HTML и многого другого;

- утилита sed представляет собой потоковый редактор, что делает его полезным для многократного изменения файла или множества файлов. Программа sed часто используется для редактирования сценариев или в качестве утилиты в других сценариях;

- утилита gawk (GNU awk) представляет собой язык программирования, который предназначен для сравнения с шаблонами и работы с содержимым файлов. Эту утилиту также можно использовать для работы со сценариями и вызывать из самих сценариев.

В целом сценарии командного интерпретатора наиболее полезны при запуске программ и изменении файлов в контексте файловой системы, а утилиты sed, gawk и perl предназначены в первую очередь для работы с содержимым файлов. Дополнительные сведения об утилитах sed и gawk см. в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01;

Запуск сценария командного интерпретатора

Сценарий командного интерпретатора можно запустить следующими способами:

- запустить другой командный интерпретатор, указав имя сценария в качестве аргумента:

```
sh myscript
```

- загрузить сценарий в текущий командный интерпретатор с помощью команды "точка":

```
. myscript
```

- сделать сценарий исполняемым с помощью утилиты chmod, а затем вызвать его следующим образом:

```
chmod 744 myscript
./myscript
```

В этом примере текущий командный интерпретатор вызывает новый командный интерпретатор для исполнения сценария.

Первая строка

Первая строка многих, если не большинства сценариев имеет следующую форму:

```
#! interpreter [arg]
```

Например, сценарии Korn shell начинаются с такой строки:

```
#! /bin/sh
```

Эта строка начинается с символа #, который указывает на то, что она является комментарием и должна игнорироваться командным интерпретатором, который обрабатывает сценарий. Первая пара символов, #!, не имеет значения для командного интерпретатора, но код загрузчика в модуле procnto распознает их как команду загрузить следующий за ними исполняемый файл, /bin/sh, и передать ему:

- путь к данному интерпретатору;
- опциональные аргументы, заданные в первой строке сценария;
- путь к сценарию;

- аргументы этого сценария, заданные в качестве командно-строковых параметров.

Например, сценарий называется my_script и пользователь вызывает его следующей командой:

```
./my_script my_arg1 my_arg2 ...
```

В этом случае procnto загружает:

```
interpreter [arg] ./my_script my_arg1 my_arg2 ...
```

Примечание.

- интерпретатор не может быть иным, нежели заданный #!.
- ядро игнорирует атрибуты setuid и getuid у сценария; дочерний процесс будет иметь такие же идентификаторы пользователя и группы, что и родительский процесс. (Для получения дополнительной информации см. раздел “Setuid и setgid” в разделе “Работа с файлами”.)

Некоторые интерпретаторы корректируют список получаемых аргументов:

- ksh удаляет себя из списка аргументов;
- gawk заменяет свое путевое имя просто на имя “gawk“;
- perl удаляет из аргументов себя и имя сценария, и добавляет имя сценария в переменную $0.

Для примера рассмотрим простые сценарии, печатающие полученные ими аргументы.

Аргументы сценария ksh

Пусть имеется следующий сценарий ksh_script:

```
#! /bin/sh

echo $0

for arg in "$@" ; do

echo $arg

done
```

Если вызвать его с помощью команды «./ksh_script one two three», то загрузчик вызовет его как «/bin/sh ./ksh_script one two three», а затем ksh удалит себя из списка аргументов. Вывод будет иметь следующий вид:

```
./ksh_script
one
two
three
```

Аргументы сценария gawk

Рассмотрим версию предыдущего сценария для интерпретатора gawk с именем gawk_script, который будет выглядеть следующим образом:

```
#!/usr/bin/gawk -f
BEGIN {
for (i = 0; i < ARGC; i++)
print ARGV[i]
}
```

Аргумент -f важен, т.к. он указывает, что бы утилита gawk прочитала сценарий из заданного файла. Без опции -f данный сценарий не будет работать ожидаемым образом.

Если этот сценарий запустить командой «./gawk_script one two three», то загрузчик вызовет его как «/usr/bin/gawk -f ./gawk_script one two three», а затем gawk заменит полное путевое имя на gawk. Вывод будет иметь следующий вид:

```
gawk
one
two
three
```

Аргументы сценария perl

Версия рассматриваемого сценария на языке perl будет выглядеть следующим образом:

```
#! /usr/bin/perl
for ($i = 0; $i <= $#ARGV; $i++) {
print "$ARGV[$i]\n";
}
```

Если этот сценарий запустить командой «./perl_script one two three» то загрузчик вызовет его как «/usr/bin/perl ./perl_script one two three», а затем perl удаляет себя и имя сценария из списка аргументов. Вывод будет иметь следующий вид:

```
one
two
three
```

10.3. Пример сценария командного интерпретатора Korn

Чтобы кратко ознакомиться с интерпретатором Korn, рассмотрим сценарий, который ищет строку, переданную ему в командной строке, в исходных и заголовочных С-файлах, которые расположены в текущем дереве каталогов.

```
#!/bin/sh
#
# tfind:
# сценарий, который ищет строки в различных файлах и выводит
# эти строки с помощью утилиты less

case $# in
1)
   find . -name '*.[ch]' | xargs grep $1 | less
   exit 0 # успешное завершение
esac
```

```
echo " Введите команду tfind строка_для_поиска,                    "
echo " где строка_для_поиска — искомая строка                      "
echo "                                                               "
echo " Например, команда tfind console state просматривает все файлы"
echo " в текущем каталоге и его подкаталогах и отображает все      "
echo " экземпляры фразы console state.                              "
exit 1 # неудачное завершение
```

Как было описано выше, первая строка указывает программу, /bin/sh, для интерпретации сценария.

Несколько следующих строк являются комментариями, которые описывают действия, выполняемые сценарием. Далее следует конструкция:

```
case $# in
1)
   ...
esac
```

Команда case...in является встроенной командой интерпретатора Korn и представляет собой одну из структур ветвления, эквивалентную оператору switch языка C.

Последовательность символов $# является переменной командного интерпретатора. Чтобы обратиться к переменной в командном интерпретаторе, ее имя следует предварить символом $, тогда командный интерпретатор отличит имя переменной от символьной строки. $# – особая переменная командного интерпретатора, которая содержит количество командно-строковых аргументов, переданных сценарию.

Константа 1) является возможным значением переменной ветвления, которое эквивалентно оператору case в языке C. Этот код проверяет, был ли передан командному интерпретатору ровно один параметр.

Строка esac завершает оператор case. В командах case и if для указания конца структуры ветвления используется имя команды, записанное справа налево.

Внутри оператора case имеется конструкция:

```
find . -name '*.[ch]' | xargs grep $1 | less
```

Эта строка выполняет несколько действий и включает в себя следующие компоненты:

- find . -name '*.[ch]';
- xargs grep $1;
- less.

Эти команды объединены при помощи символа конвейера |. Конвейер – одна из самых мощных возможностей командного интерпретатора; он принимает выходные данные программы, которая указана слева, и передает их в качестве входных данных программе, которая указана справа. Конвейер позволяет строить сложные действия из более простых компонентов. Более подробные сведения см. в разд. "Перенаправление ввода и вывода" раздела 4.

Первый компонент конвейера, find . -name '*.[ch]', использует еще одну мощную и распространенную команду. Большинство файловых систем структурировано в виде рекурсивной иерархии каталогов, а утилита find выполняет рекурсивный поиск в этой иерархии. В этом примере утилита find ищет файлы, которые заканчиваются на .c или на .h (т. е. исходные и заголовочные файлы языка C), и распечатывает их имена.

Групповые символы имен файлов заключаются в одиночные кавычки, поскольку командный интерпретатор интерпретирует их особым образом. В отсутствие кавычек командный интерпретатор раскрывает групповые символы в текущий каталог, однако в этом примере необходимо, чтобы интерпретацию групповых символов выполнила утилита find, поэтому групповые символы скрыты от командного интерпретатора с помощью кавычек. Более подробные сведения см. в разд. "Применение кавычек со специальными символами" раздела 4.

Следующий компонент конвейера, xargs grep $1, выполняет два действия:

- grep представляет собой утилиту для поиска в содержимом файлов. Она выполняет поиск первого аргумента в файлах, которые указаны в ее командной

строке. $1 – еще одна специальная переменная командного интерпретатора, которая содержит первый аргумент, переданный сценарию (т. е. искомую строку);

- xargs представляет собой утилиту, которая преобразует свои входные данные в командно-строковые параметры другой переданной ей команды. В этом примере утилита xargs принимает список файлов от утилиты find и превращает его в командно-строковые аргументы команды grep. Здесь утилита xargs используется в первую очередь для эффективности. Аналогичные действия можно было бы выполнить при помощи только утилиты find:

```
find . -name '*.[ch]' -exec grep $i {} | less
```

Эта команда загружает и запускает программу grep для каждого найденного файла. Команда, которая была использована в примере:

```
find . -name '*.[ch]' | xargs grep $1 | less
```

запускает утилиту grep только тогда, когда программа xargs накапливает достаточно файлов для заполнения командной строки, что в общем случае сокращает число вызовов утилиты grep и делает сценарий эффективнее.

Последний компонент конвейера less является утилитой страничного представления вывода. Команда способна сгенерировать большой объем выходных данных, которые выходят за пределы экрана, а утилита less представляет вывод постранично и позволяет перелистывать данные вперед и назад.

Конструкция case также включает в себя команду:

```
exit 0 # успешное завершение
```

которая следует за вызовом утилиты find. Эта команда завершает сценарий и возвращает значение 0. В программировании для командного интерпретатора 0 означает истину или успех, а любое ненулевое значение указывает на ложь или ошибку (в противоположность языку C).

Последний блок команд просто содержит справочную информацию. Если передать сценарию неправильные аргументы, он выведет описание с правилами своего использования, а затем вернет код ошибки.

```
echo " Введите команду tfind строка_для_поиска,                    "
```

```
echo " где строка_для_поиска - искомая строка                    "
echo "                                                            "
echo " Например, команда tfind console state просматривает все файлы"
echo " в текущем каталоге и его подкаталогах и отображает все    "
echo " экземпляры фразы console state.                           "
exit 1 # неудачное завершение
```

10.4. Эффективность

Сценарии обычно менее эффективны, чем специальные программы, написанные на языке C или C++, поскольку сценарии:

- интерпретируются, а не компилируются;
- выполняют большую часть своей работы, запуская другие программы.

Тем не менее разработка сценария может оказаться быстрее, чем написание программы, особенно при построении сценария из конвейеров и существующих утилит.

10.5. Рекомендации разработчикам сценариев

Далее приведены некоторые рекомендации, которые следует учитывать при написании сценариев.

- чтобы запускать сценарий так же, как утилиту, необходимо сделать его исполняемым с помощью команды chmod. Например, чтобы любой пользователь мог запускать сценарий, следует выполнить команду:

```
chmod a+x имя_сценария
```

Сценарий не обязан быть исполняемым, если планируется вызывать его путем передачи в качестве аргумента командному интерпретатору:

```
ksh имя_сценария
```

или запускать его с помощью команды "точка":

```
. имя_сценария
```

- как и в случае с любым исполняемым файлом, если сценарий находится в каталоге, который не указан в переменной окружения **PATH**, то для его запуска необходимо указать путь к сценарию:

~/bin/my_script

- запущенный сценарий наследует окружение от родительского процесса. Если сценарий исполняет команду, которая может отсутствовать в переменной окружения **PATH**, необходимо указать путь к этой команде или добавить этот путь в переменную **PATH** сценария;
- если сценарий запущен не командой "точка", он не может изменять окружение и текущий каталог своего родительского командного интерпретатора;
- сценарий не сработает при наличии в нем символов конца строки, которые используются в операционной системе DOS. При редактировании сценария для ЗОСРВ «Нейтрино» в операционной системе Windows следует преобразовать файл сценария в формат, который используется файловой системой QNX 4, с помощью утилиты textto с параметром -l.

11. Работа с файловыми системами

11.1. Общие сведения

Операционная система ЗОСРВ «Нейтрино» предоставляет множество файловых систем, что позволяет без труда работать с дисками, на которых используются файловые системы DOS, Linux и QNX 4. Глава 9 документа «Описание применения. Часть 1. Системная архитектура» КПДА.10964-01 31 01 описывает классы и возможности этих файловых систем.

В операционной системе ЗОСРВ «Нейтрино»:

- можно динамически запускать и останавливать файловые системы;
- несколько файловых систем могут работать одновременно;
- приложениям предоставляется единое универсальное пространство путевых имен и интерфейс независимо от конфигурации и количества базовых файловых систем.

На настольном компьютере с операционной системой ЗОСРВ «Нейтрино» необходимые блочные файловые системы запускаются при загрузке; другие файловые системы запускаются пользователем как отдельные администраторы. Блочной файловой системой по умолчанию является QNX 4.

11.2. Настройка, запуск и остановка блочной файловой системы

При загрузке компьютера операционная система обнаруживает разделы на устройствах блочного ввода/вывода и автоматически запускает нужную файловую систему для каждого раздела (см. раздел 8).

Скорее всего, пользователю вообще не потребуется останавливать или перезапускать блочную файловую систему. Чтобы обновить файловую систему при изменении ее параметров, можно воспользоваться командой mount с параметром -e или -u.

Если требуется изменить какие-либо параметры устройства блочного ввода/вывода, можно остановить соответствующий драйвер devb-* с помощью

команды slay (будьте внимательны, чтобы не выбить себе почву из-под ног) и перезапустить его, однако после этого необходимо явно смонтировать к нему все файловые системы.

Для определения свободного пространства, имеющегося в вашей файловой системе, используйте команду df. Более подробные сведения см. в справочнике по утилитам.

Некоторые файловые системы поддерживают возможность маркировать их как «грязные». Это позволяет пропускать интенсивные проверки логической целостности при последующих загрузках. В файловых системах QNX4 и Ext2 для этой цели служит бит, выполняющий функцию флага. В файловых системах FAT для этого предназначены сигнатуры из нескольких бит. Идея заключается в том, что при монтировании файловой системы с правами по чтению и записи выставляется этот флаг, а после корректного отмонтирования файловой системы флаг сбрасывается. Между этими двумя событиями файловая система считается «грязной» и может нуждаться в проверке. Файловая система Power-Safe такого флага не имеет — она просто откатывается к последнему целостному «снимку» (snapshot). Для отключения данного маркирования используется опция «blk marking=none», подробнее см. описание io-blk.so в справочнике по утилитам.

11.3. Монтирование и демонтирование файловых систем

С файловыми системами работают следующие утилиты:

- mount — монтирование специального блочного устройства или удаленной файловой системы;
- umount — демонтирование устройства или файловой системы.

Например, если процесс fs-cifs уже работает, то к нему можно смонтировать файловую систему следующим образом:

```
mount -t cifs -o guest,none //SMB_SERVER:10.0.0.1:/QNX_BIN /bin
```

По умолчанию файловые системы монтируются в режиме доступа по «чтению-записи» (read-write), если это позволяет данный физический носитель. Для

монтирования в режиме «только чтение» (read only) используется опция -r утилиты mount. Библиотека io-blk.so так же поддерживает опцию ro для монтирования в режиме «только чтения» блок-ориентированных файловых систем.

Для временного изменения режима монтирования файловой системы используется опция -u утилиты mount. Например, если некоторая файловая система обычно монтируется в режиме «только чтение», но к ней необходимо получить доступ в режиме «чтение-запись», то следует воспользоваться опциями -uw. Например:

```
mount -uw /
```

Для возврата в режим «только чтение» используются опции -ur:

```
mount -ur /
```

Перед удалением или извлечением носитеся, смонтированного в режиме «чтение-запись» следует предварительно воспользоваться утилитой umount.

Более подробные сведения об использовании и синтаксисе утилит mount и umount см. в справочнике по утилитам.

11.4. Файловая система образа

В данном контексте под термином образ (image) понимается образ операционной системы – файл, который содержит операционную систему, исполняемые файлы и файлы данных, связанные с программами, и предназначен для использования во встраиваемой системе. Образ можно уподобить небольшой "файловой системе", поскольку он имеет каталоговую структуру, в которой содержатся файлы.

Образ включает в себя небольшую структуру каталогов, которая указывает модулю procnto имена и расположение файлов, и сами файлы. Во время работы встраиваемой системы доступ к образу осуществляется так же, как и к любой другой файловой системе, предназначенной только для чтения:

```
# cd /proc/boot
# ls
```

```
.script     cat      data1       data2      devc-ser8250
esh         ls       procnto
# cat data1
```

Это файл данных с именем data1, который находится в образе. Обратите внимание на то, что это удобный способ связывания файлов данных с программами.

Пример, который приведен выше, демонстрирует два аспекта функционирования образа операционной системы как файловой системы. При вводе команды ls операционная система загружает утилиту ls (путевое имя /proc/boot/ls) из файловой системы своего образа. Затем при вводе команды cat операционная система загружает утилиту cat также из файловой системы образа и открывает файл data1.

Конфигурирование образа операционной системы

Создать образ операционной системы можно с помощью утилиты mkifs (от англ. MaKe Image FileSystem — сборка файловой системы образа). Более подробные сведения см. в руководстве "Building Embedded Systems" и описании утилиты mkifs в «Описании программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

11.5. "Файловая система" в ОЗУ: каталог /dev/shmem

Операционная система ЗОСРВ «Нейтрино» обеспечивает простую файловую систему в оперативной памяти (ОЗУ), которая позволяет помещать файлы в каталог /dev/shmem для выполнения операций считывания и записи.

Примечание. На самом деле /dev/shmem не является файловой системой. Это окно к именам объектов разделяемой памяти, которое случайно обладает некоторыми характеристиками файловой системы.

Эта файловая система не является полноценной, поскольку в ней отсутствуют такие возможности, как поддержка подкаталогов. Кроме того, файловая система в ОЗУ не включает в себя элементы . и .. для текущего и родительского каталогов.

Файлы, которые находятся в каталоге /dev/shmem, помечены как именованные специальные файлы (S_IFNAME). Это обманывает многие утилиты (например, gzip и more), которые работают с обычными файлами (S_IFREG). По этой причине многие утилиты могут не работать с файловой системой в ОЗУ.

Примечание. При сжатии и разжатии файлов в /dev/shmem с помощью gzip необходимо задавать опцию -f.

Файловая система в оперативной памяти в основном используется подсистемой разделяемой памяти модуля procnto. В особых ситуациях (например, при отсутствии файловой системы) файловую систему в ОЗУ можно использовать для хранения файловых данных. Средств, предотвращающих поглощение всей оперативной памяти одним файлом, не существует. Такое поглощение может привести к проблемам в других процессах.

Файловая система в ОЗУ, как правило, применяется в компактных встраиваемых системах, где необходима небольшая и быстрая файловая система для временного хранения данных, а сохранение данных после перезагрузки не требуется.

Файловая система в оперативной памяти входит в состав модуля procnto и не требует настройки и драйверов устройств. Можно просто создавать файлы в каталоге /dev/shmem и увеличивать их до любого размера (в зависимости от ресурсов оперативной памяти).

Несмотря на то, что сама файловая система в ОЗУ не поддерживает жесткие, гибкие ссылки и каталоги, можно создать на нее ссылку администратора процессов. Например, чтобы создать ссылку на каталог /tmp в ОЗУ, следует ввести команду:

```
ln -sP /dev/shmem /tmp
```

Эта команда указывает модулю procnto создать ссылку администратора процессов /tmp на каталог /dev/shmem. После этого большинство прикладных программ сможет открывать файлы каталога /tmp так, как будто он является обычной файловой системой.

Примечание. Для того чтобы файловая система в ОЗУ имела как можно меньший объем кода в администраторе процессов, она намеренно лишена таких возможностей "больших" файловых систем, как блокирование файлов и создание каталогов.

11.6. Файловая система QNX 4

Файловая система QNX 4 используется в операционной системе ЗОСРВ «Нейтрино» по умолчанию и реализована в виде разделяемого объекта fs-qnx4.so и автоматически загружается драйверами devb-* при монтировании.

Для создания дискового раздела ЗОСРВ «Нейтрино» можно воспользоваться утилитами fdisk и dinit.

Эта файловая система имеет надежную архитектуру, в которой память выделяется с использованием экстентов, битовой карты и контрольные структуры данных, обеспечивающих защиту данных от потерь, а также восстановление данных.

Файловая система QNX 4 включает в себя следующие возможности:

- соответствие стандарту POSIX и хранение файлов в виде экстентов;
- надежность – вся важная информация файловой системы сбрасывается на диск;
- дисковые "сигнатуры" и специальную ключевую информацию, которая обеспечивает быстрое восстановление данных при повреждении диска;
- имена файлов длиной до 505 символов;
- многопоточную архитектуру;
- приоритет, управляемый клиентом;
- формат диска, аналогичный файловой системе QNX 4;
- поддержку файлов размером до 2 Гбайт минус 1 байт.

Дополнительные сведения о реализации файловой системы QNX 4 см. в поразд. "Дисковая структура файловой системы QNX 4" раздела 18.

Экстенты

В файловой системе QNX 4 обычные файлы и каталоги хранятся в виде последовательности экстентов (extents) — непрерывных последовательностей дисковых блоков. Экстенты файла регистрируются в его каталоговом элементе. Если для хранения файла файловой системе требуется более чем один экстент, она использует связанный список блоков экстентов (extent blocks), который содержит информацию об экстентах.

Когда файлу необходимо дополнительное пространство, файловая система пытается расширить файл с сохранением его непрерывности. Если это невозможно, файловая система выделяет новый экстент и по необходимости новый блок экстентов. Когда файловая система выделяет или расширяет экстент, она может выделить избыточное пространство в предположении, что процесс продолжит записывать данные в файл и заполнит это пространство. При закрытии файла все дополнительное пространство освобождается.

Эта архитектура обеспечивает максимальную непрерывность файлов даже при записи данных в несколько файлов одновременно. Поскольку в большинстве жестких дисков реализовано кэширование дорожек, экстентная архитектура не только гарантирует максимально быстрое чтение файлов с диска, но и сводит к минимуму фрагментацию данных на диске.

Более подробные сведения о производительности см. в разделе 20.

Имена файлов

Исходная файловая система QNX 4 поддерживала имена файлов длиной не более 48 символов. Благодаря обратно-совместимому расширению, которое используется по умолчанию, этот предел увеличился до 505 символов. Дисковый формат остался неизменным. Новые системы распознают расширенное имя, однако старые системы используют урезанное имя длиной 48 символов.

При создании файловой системы QNX 4 по умолчанию поддерживаются длинные имена файлов. Чтобы отключить длинные имена, следует указать параметр –N в вызове утилиты dinit. Чтобы добавить поддержку длинных имен файлов в существующую файловую систему QNX 4, необходимо войти в

операционную систему как пользователь root и создать в корневом каталоге файловой системы пустой файл с именем .longfilenames, доступный только для чтения, владельцем которого является пользователь root:

```
cd корневой_каталог
touch .longfilenames
chmod a=r .longfilenames
chown root:root .longfilenames
```

Примечание. После создания файла .longfilenames необходимо перезапустить файловую систему, чтобы включить поддержку длинных имен файлов.

Определить максимальную длину имени файла, поддерживаемую файловой системой, можно с помощью утилиты getconf:

```
getconf _PC_NAME_MAX корневой_каталог
```

где корневой_каталог — корневой каталог файловой системы.

В именах файлов нельзя использовать символы 0x00—0x1F, 0x7F и 0xFF. Кроме того, символ / (0x2F) является разделителем путевого имени и не может входить в компонент файловой системы. Можно использовать пробелы, но их необходимо заключать в кавычки в командной строке. В кавычки также требуется заключать все групповые символы, которые поддерживает командный интерпретатор. Более подробные сведения см. в подразд. "Применение кавычек со специальными символами" раздела 4.

Ссылки и индексные дескрипторы

Данные файла хранятся отдельно от его имени. На одни и те же данные может ссылаться несколько имен. Каждое имя файла называется ссылкой (link) и указывает на реальные данные файла. На самом деле существуют два вида

ссылок — жесткие ссылки (hard links), или просто "ссылки", и символьные ссылки (symbolic links), которые описываются в следующем разделе.

Для того чтобы обеспечить возможность создавать ссылки на любой файл, имя файла отделяется от остальной информации, которая описывает файл. Эта информация хранится в таблице, которая называется индексным дескриптором (inode, information node — информационный узел).

Если существует только одна ссылка на файл (иначе говоря, файл имеет единственное имя), то информация индексного дескриптора (т. е. информация о файле, отличная от его имени) хранится в каталоговом элементе файла. Если у файла имеется более чем одна ссылка, индексный дескриптор хранится в виде записи в специальном файле с именем /.inodes, на которую указывает каталоговый элемент файла (рис. 11.1).

Рис. 11.1. Две ссылки, которые указывают на один файл

Следует обратить внимание на то, что создать ссылку на файл можно лишь в случае, если ссылка и файл находятся в одной файловой системе.

Существуют две другие ситуации, в которых файл может иметь запись в файле /.inodes:

- если имя файла имеет длину более 16 символов, то информация индексного дескриптора хранится в файле /.inodes, освобождая место для 48-символьного файлового имени в каталоговом элементе файла. Имена файлов длиной более 48 символов хранятся в файле .longfilenames, который способен содержать

имена длиной до 505 символов. Урезанное 48-символьное имя также помещается в каталоговый элемент файла для использования существующими системами;

- если в какой-то момент у файла было несколько ссылок, а затем все ссылки, кроме одной, были удалены, запись, соответствующая этому файлу в файле /.inodes, сохраняется. Это делается из-за слишком большой трудоемкости поиска каталогового элемента, который указывает на запись индексного дескриптора (в записях индексных дескрипторов нет обратных ссылок, которые указывают на каталоговые элементы).

Удаление ссылок

При создании файла его счетчик ссылок (link count) устанавливается равным единице. По мере добавления и удаления ссылок на этот файл счетчик ссылок увеличивается и уменьшается. Дисковое пространство, которое занимают данные файла, не освобождается и не помечается в битовой карте как неиспользуемое до тех пор, пока значение счетчика ссылок не становится равным нулю и все программы, которые используют этот файл, не закрывают его. Это позволяет использовать открытый файл даже после того, как все ссылки на него уничтожены. Такое поведение соответствует требованиям стандарта POSIX и является обычным для UNIX-систем.

Ссылки на каталоги

Несмотря на то, что создавать жесткие ссылки на каталоги нельзя, каждый каталог имеет две жестко закодированные встроенные ссылки:

. ("точка")

.. ("две точки")

Файловое имя "точка" указывает на текущий каталог, а "две точки" — на предыдущий (или родительский) каталог в иерархии.

Следует иметь в виду, что в отсутствие родительского каталога ссылка "две точки" указывает на текущий каталог. Например, запись "две точки" каталога / указывает на каталог /, поскольку перейти на более высокий путевой уровень невозможно.

Примечание. Стандарт POSIX не требует, чтобы файловая система включала в себя записи . и ... Например, эти записи отсутствуют в файловых системах флэш-устройств и файловой системе /dev/shmem.

Символьные ссылки

Символьная ссылка (symbolic link) представляет собой специальный файл, данные которого обычно содержат путевое имя. Когда имя символьной ссылки используется в запросе ввода/вывода, например, в функции open(), ссылочная часть путевого имени заменяется "данными" ссылки и путь определяется заново.

Символьные ссылки являются гибким способом косвенного обращения к файлу по путевому имени и часто используются для создания нескольких путей к одному файлу. В отличие от жестких ссылок, символьные ссылки могут указывать на другие файловые системы и каталоги.

В следующем примере каталоги /net/node1/usr/fred и /net/node2/usr/barney связаны друг с другом, несмотря на то, что они находятся в разных файловых системах и даже на разных узлах (рис. 11.2). Эту связь нельзя создать с помощью жестких ссылок, но можно посредством символьной ссылки следующим образом:

```
ln -s /net/node2/usr/barney /net/node1/usr/fred
```

Следует обратить внимание на то, что символьная ссылка и целевой каталог не обязательно используют одно и то же имя. Как правило, символьная ссылка создается для того, чтобы связать один каталог с другим. Тем не менее символьные ссылки можно применять и к файлам, например:

```
ln -s /net/node1/usr/src/game.c /net/node1/usr/eric/src/sample.c
```

Рис. 11.2. Символьные ссылки

Примечание. При удалении символьной ссылки уничтожается только сама ссылка, но не ее цель.

Несколько функций работает с символьными ссылками непосредственно. В этих функциях символьный элемент путевого имени не заменяется на целевой. К этим функциям относится unlink(), которая удаляет символьную ссылку, а также lstat() и readlink().

Поскольку символьные ссылки могут указывать на каталоги, их некорректная настройка способна привести к проблемам, например, к циклическим ссылкам на каталоги. Чтобы выйти из бесконечного ссылочного цикла, система устанавливает максимальное число переходов, которое задается переменной SYMLOOP_MAX в заголовочном файле <limits.h>.

Надежность файловой системы

Файловая система QNX 4 обеспечивает высокую производительность без ущерба для надежности. Это достигается несколькими способами.

Несмотря на то, что большая часть данных содержится в буферном кэше и записывается лишь после короткой задержки, запись критически важных данных файловой системы выполняется немедленно. Обновленные каталоги, индексные дескрипторы, блоки экстентов и битовые карты сбрасываются на диск для того,

чтобы гарантированно защитить структуру файловой системы на диске от повреждений (т. е. внутренняя согласованность дисковых данных никогда не должна нарушаться).

Иногда возникает необходимость обновления всех перечисленных выше структур. Например, если файл перемещается в каталог, последний экстент которого заполнен, каталог требуется расширить. Для таких ситуаций предусмотрена тщательно продуманная последовательность действий, при которой файловая система не повреждается после перезагрузки в случае катастрофической аварии (например, сбоя электропитания) во время выполнения операции. В худшем случае некоторые выделенные блоки могут оказаться неиспользованными. Их можно освободить с помощью утилиты chkfsys. Более подробные сведения см. в разделе 18.

11.7. Файловая система Power-Safe

Файловая система Power-Safe, поддерживаемая разделяемым объектом fs-qnx6.so, обеспечивает сохранность данных от потери и повреждений при аварийных отключениях электропитания. Многие ее характеристики такие же, как у файловой системы QNX4, например следующие:

- имена файлов 510 байт (кодировка UTF-8);
- метод обновлений «copy-on-write» (COW), который позволяет избежать повреждения данных при отключении питания в момент записи;
- «снимок» (snapshot), который содержит целостное представление файловой системы.

Информацию о файловой системе Power-Safe см. так же в главе 9 руководства по системной архитектуре.

Предупреждение. Если дисковод не поддерживает синхронизацию, то fs-qnx6.so не может гарантировать устойчивость файловой системы по питанию. Перед тем как использовать данную файловую систему на каком-либо устройстве кроме традиционных вращающихся жестких дисков — такое как устройство USB/Flash — убедитесь, что данные устройство удовлетворяет

требованиям файловой системы. Подробнее см. подраздел справочника по утилитам, посвященный модулю fs-qnx6.so.

Для создания файловой системы Power-Safe используется утилита mkqnx6fs. Например:

```
mkqnx6fs /dev/hd0t76
```

Для указания размеров логических блоков, порядка следования байт, количества логических блоков и тому подобных данных используются опции утилиты mkqnx6fs.

После того, как файловая система отформатирована, ее можно смонтировать утилитой mount. Например:

```
mount -t qnx6 /dev/hd0t76 /mnt/psfs
```

Подробнее об опциях файловой системы Power-Safe см. описание fs-qnx6.so в справочнике по утилитам.

Для проверки логической целостности этой файловой системы (чего обычно не требуется делать) используется chkqnx6fs.

Примечание. Утилита chkfsys, используемая для проверки целостности файловой системы QNX4, будет выдавать сообщение о том, что файловая система Power-Safe повреждена.

Загрузка

В настоящее время для ЭВМ на базе x86 поддерживается загрузка, которая основана на таблицах разделов и использует BIOS, поддерживающие INT13X (LBA).

Утилита mkqnx6fs создает каталог .boot в корне новой файловой системы. Этот каталог всегда присутствует в файловой системе и всегда имеет вторую inode-запись (сам корневой каталог имеет первую inode-запись). Так же утилита mkqnx6fs устанавливает новый вторичный загрузчик в первые 8 Кбайт данного раздела (и прописывает в него расположение и смещение новой файловой системы).

В время функционирования fs-qnx6.so защищает этот каталог — в частности он не может быть удален или переименован, не может превысить размер 4096 байт (128 записей). Файлы, помещаемые в каталог .boot, считаются загружаемыми образами, созданными утилитой mkifs. Поэтому для удобства целесообразно, что бы имя файла описывало загружаемый образ (например, 6.3.2SP3, 6.4.0_with_diskboot или SafeMode_1CPU).

Данный каталог может содержать до 126 элементов. В нем могут создаваться другие типы объектов (например, каталоги или символьные ссылки), на начальный загрузчик будет их игнорировать. Так же загрузчик игнорирует регулярные файлы, если их размер 0 или больше 2 Гбайт, а так же если их имена содержат более 27 символов.

Файловая система автоматически приостанавливает создание «снимков» когда загрузочный образ открыт по записи. Это позволяет гарантировать то, что начальный загрузчик никогда не будет «видеть» частично записанный образ. Обычно образ создается в каком-либо другом месте и затем копируется в каталог .boot, поэтому файлы образов остаются открытыми в течение короткого интервала времени. Тем не менее эта схема работает так же и в том случае, когда вывод утилиты mkifs направляется прямо в финальный загрузочный файл.

> **Примечание**. Что бы воспрепятствовать использованию этой особенности для DoS-атак загрузочный каталог по умолчанию имеет такие права доступа:root rwx------. Эти права можно изменить с помощью chmod и chown, но необходимо помнить, что если вы предоставите какому-либо пользователю доступ в данный каталог, то это позволит ему как добавлять собственные образы, так и удалять имеющиеся образы.

Загрузка с файловой системы Power-Safe рассматривается далее в подразделе «Управление загрузкой ЗОСРВ «Нейтрино».

«Снимки»

«Снимок» (snapshot)- это сохраненное представление стабильного состояния файловой системы Power-Safe. Каждая смонтированная файловая система имеет

один стабильный «снимок» и одно рабочее представление, в котором выполняются модификации COW (copy-on-write) в стабильный «снимок».

Как только создан новый «снимок» файловая система приостанавливает свои операции (для обеспечение целостности системы), обновляются битовые карты, все «грязные» блоки принудительно записываются на диск и записывается альтернативный суперблок файловой системы с увеличенным порядковым номером. Затем файловая система продолжает обычную работу, при этом на месте старого суперблока создается другое рабочее представление.

Когда файловая система монтируется заново после сбоя по питанию, она восстанавливает последний стабильный «снимок». «Снимки» выполняются:

- явно, в случае глобального вызова sync() для всех файловых систем;
- явно, в случае вызова fsync() для любого файла в файловой системе Power-Safe;
- явно, в случае переключения в режим «только чтение» с помощью команды mount -ur;
- периодически, при срабатывании таймера, заданного опцией snapshot= утилиты mount (по умолчанию 10 секунд).

Создание «снимков» может быть отключено на глобальном или локальном уровне. В случае отключения новый суперблок записываться не будет, а попытка создания снимка будет завершаться с ошибкой EAGAIN (либо без выставления кода ошибки — в случае использования sync() или таймеров). Если в момент отмонтирования (неявного или при сбое по питанию) снимки будут оставаться отключенными, то все ожидающие модификации будут сброшены (потеряны).

Кроме того снимки автоматически отключаются на постоянной основе после неустранимой ошибки, приводящей к нарушению целостности файловой системы. Например, переполнение кэша при модификации битовой карты или ошибка дискового ввода-вывода при записи снимка. В этом случае файловая система принудительно переводится в режим «только чтение», а все текущие и последующие снимки пропускаются. Цель этого состоит в том, что бы сохранить нетронутым и доступным для следующего монтирования/загрузки последний

стабильный снимок (т.е. файловая система будет в гарантированно стабильном состоянии, даже если оно несколько устарело). Это предназначено для противодействия некоторым ситуациям, когда возникают редко случающиеся серьезные сбои.

Ручное отключение снимком может быть полезным в случае, когда необходимо обеспечить атомарность последовательности высокоуровневых операций, которые либо должны быть выполнены все, либо ни одна из них (в случае, например, когда сбой по питанию произошел при выполнении этой последовательности). К таким случаям можно отнести обновление программного обеспечения или дефрагментацию файловой системы.

Для отключения снимков на глобальном уровне необходимо сбросить флаг FS_FLAGS_COMMITTING на соответствующей файловой системе. Для этого используется вызов devctl() с командой DCMD_FSYS_FILE_FLAGS:

```
struct fs_fileflags flags;
memset( &flags, 0, sizeof(struct fs_fileflags));
flags.mask[FS_FLAGS_GENERIC] = FS_FLAGS_COMMITTING;
flags.bits[FS_FLAGS_GENERIC] = disable ? 0 : FS_FLAGS_COMMITTING;
devctl( fd, DCMD_FSYS_FILE_FLAGS, &flags,
sizeof(struct fs_fileflags), NULL);
```

Примечание. Этот флаг у всей файловой системы только один, и он может быть выставлен или сброшен любым клиентом, выполняющимся в суперпользовательском режиме доступа. Поэтому приложения должны координировать между ними использование этого флага.

В качестве альтернативы можно воспользоваться утилитой chattr (как интерфейсом к указанному выше вызову devctl()):

```
# chattr -snapshot /fs/qnx6
/fs/qnx6: -snapshot
...
```

```
# chattr +snapshot /fs/qnx6
/fs/qnx6: +snapshot
```

Для отключения снимков на локальном уровне необходимо с помощью той же команды DCMD_FSYS_FILE_FLAGS функции devctl() скорректировать счетчик QNX6FS_SNAPSHOT_HOLD для файлового дескриптора. Каждый открытый файл имеет свой счетчик удержаний, а сумма всех счетчиков удержаний является глобальным счетчиком удержания, отключающим снимки в случае, если его значение ненулевое. Следовательно, если какой-либо клиент устанавливает счетчик удержаний, то снимки отключаются до тех пор, пока все клиенты не сбросят свои счетчики.

Счетчик удержаний представляет собой 32-битное значение и может быть инкриментирован более одного раза (т.е. должен быть сбалансированным с помощью соответствующего количества декрементов). Если файловый дескриптор закрыт или процесс, его открывший, завершен, то все выполненные на нем локальные удержания отменяются.

Достоинство этой схемы заключается в том, что она не требует специальной координации между клиентами; каждый из них может создавать собственные атомарные последовательности операций используя свой независимы счетчик удержаний:

```
struct fs_fileflags
flags;
memset( &flags, 0, sizeof(struct fs_fileflags));
flags.mask[FS_FLAGS_FSYS] = QNX6FS_SNAPSHOT_HOLD;
flags.bits[FS_FLAGS_FSYS] = QNX6FS_SNAPSHOT_HOLD;
devctl( fd, DCMD_FSYS_FILE_FLAGS, &flags,
sizeof(struct fs_fileflags), NULL);
...
memset( &flags, 0, sizeof(struct fs_fileflags));
flags.mask[FS_FLAGS_FSYS] = QNX6FS_SNAPSHOT_HOLD;
```

```
flags.bits[FS_FLAGS_FSYS] = 0;
devctl( fd, DCMD_FSYS_FILE_FLAGS, &flags,
sizeof(struct fs_fileflags), NULL);
```

В этом случае утилита chattr не подходит для манипулирования отключением снимков, поскольку счетчик удержаний восстанавливается сразу после завершения утилиты (т.к. его файловый дескриптор закрывается).

Однако, она подходит для получение информации о текущем статусе файловой системы, поскольку она отдельно отображает глобальные и локальные флаги:

```
# chattr /fs/qnx6
/fs/qnx6: +snapshot +contiguous +used +hold
```

Если «+snapshot» не отображается, то снимки были отключены с помощью глобального флага. Если отображается «+hold», то снимки были отключены из-за ненулевого значения глобального счетчика удержаний (установленного неспецифицированным количеством клиентов). Если постоянно (даже после sync()) отображается «+dirty», то снимки отключены или из-за возможно фатальной ошибки, или дисковая аппаратура не поддерживает полной синхронизации данных («track cache flush»).

Примечание. Само по себе включение снимков не приводит к созданию снимка — для этого нужно явно вызвать fsync(). Целесообразно вызывать fsync() как перед выключением, так и после включения снимков (утилита chattr делает этот вызов самостоятельно).

11.8. Файловая система DOS

Файловая система операционной системы DOS обеспечивает прозрачный доступ к DOS-дискам, что позволяет работать с файловыми системами DOS так же, как и с файловыми системами ЗОСРВ «Нейтрино» (POSIX). Эта прозрачность дает процессам возможность оперировать DOS-файлами без какой-либо специальной информации и дополнительных действий.

Разделяемый объект fs-dos.so (см. руководство "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01) позволяет монтировать файловые системы DOS (FAT12, FAT16 и FAT32) в операционной системе ЗОСРВ «Нейтрино». Этот разделяемый объект автоматически загружается драйверами devb-* при монтировании файловой системы FAT операционной системы DOS. Если требуется выполнять операции чтения и записи над гибким диском DOS, его следует смонтировать с помощью команды

```
mount -t dos /dev/fd0 /fd
```

Сведения о символах, которые можно использовать в файловых именах в файловой системе DOS см. на сайте сети разработчиков Microsoft (Microsoft Developer Network) **http://msdn.microsoft.com**. Самые жесткие ограничения налагаются на имена файлов в формате FAT 8.3. В них можно использовать только буквы в верхнем регистре, цифры, а также символы $ % ' — _ @ { } ~ # ( ). Формат VFAT несколько смягчает эти ограничения и позволяет использовать буквы в нижнем регистре и символы [ ] ; , = +. Файловая система DOS операционной системы ЗОСРВ «Нейтрино» автоматически переводит файловые имена в формате FAT 8.3 в верхний регистр, чтобы сымитировать возможность использования букв в нижнем регистре (информация о регистре, тем не менее, не сохраняется).

Более подробные сведения об администраторе файловой системы DOS см. в описании файла fs-dos.so в справочнике по утилитам и в главе "Файловые системы" руководства по системной архитектуре.

11.9. Файловая система для устройств CD-ROM

Файловая система для устройств CD-ROM в операционной системе ЗОСРВ «Нейтрино» обеспечивает прозрачный доступ к компакт-дискам, что позволяет работать с их файловыми системами так же, как с файловыми системами стандарта POSIX. Эта прозрачность дает процессам возможность манипулировать файлами компакт-дисков без какой-либо специальной информации и дополнительных действий.

Разделяемый объект fs-cd.so обеспечивает поддержку файловой системы стандарта ISO 9660, а также различных расширений, в том числе Rock Ridge (RRIP), Joliet (компании Microsoft) и многосеансовых дисков (стандарта Kodak Photo CD, расширенных аудиодисков). Этот разделяемый объект автоматически загружается драйверами devb-* при монтировании файловой системы ISO 9660.

Файловая система для устройств CD-ROM принимает любые символы, которые присутствуют в имени файла. Она допускает только чтение, поэтому все надлежащие ограничения на имя файла должны налагаться программой, которая подготавливает образ компакт-диска. Для строгого соответствия стандарту ISO 9660 необходимо использовать только символы 0, 1, ..., 9, A, ..., Z и _, однако расширения Joliet и Rock Ridge значительно менее требовательны.

Сведения о записи компакт-дисков см. в разделе 18.

Примечание. Модуль fs-cd.so является устаревшим и более не развивается. Для его замены предназначен модуль fs-udf.so, который поддерживает файловые системы ISO-9660 в дополнение к UDF. Информация о UDF представлена ниже, в соответствующем подразделе этого раздела.

11.10. Файловая система Linux Ext2

Файловая система Ext2, которая входит в состав операционной системы ЗОСРВ «Нейтрино», обеспечивает прозрачный доступ к дисковым разделам операционной системы Linux. Эта файловая система не поддерживает некоторые возможности Ext2, в том числе:

- фрагменты файлов (выделение подблоков);
- крупные файлы размером более 2 Гбайт;
- расширение типа файла;
- сжатие;
- каталоги со структурой в виде сбалансированного дерева.

Поддержка файловой системы Ext2 обеспечивается разделяемым объектом fs-ext2.so. Этот разделяемый объект автоматически загружается драйверами devb-* при монтировании файловой системы Ext2.

Предупреждение.

Несмотря на то, что Ext2 является основной файловой системой для операционной системы Linux, не рекомендуется использовать файл fs-ext2.so в качестве замены файловой системы QNX4. В настоящее время загрузка с разделов Ext2 не поддерживается. Кроме того, целостность файловой системы Ext2 в значительной степени обеспечивается специальной проверочной программой. Эта программа и другие утилиты поддержки (например, mke2fs) в настоящее время не реализованы в ЗОСРВ «Нейтрино».

Если демонтирование файловой системы Ext2 не выполнено должным образом, то проверочная программа обычно отвечает за его завершение при следующем монтировании. Несмотря на то, что модуль fs-ext2.so выполняет краткий тест файловой системы, он автоматически монтирует ее только для чтения при обнаружении серьезных проблем, которые должны быть исправлены программой проверки файловой системы.

Файловая система Ext2 разрешает использовать в имени файла те же символы, что и файловая система QNX 4 (см. подразд. "Имена файлов" ранее в этом разделе).

11.11. Файловые системы флэш-памяти

Операционная система ЗОСРВ «Нейтрино» включает в себя POSIX-совместимые драйверы файловых систем для устройств флэш-памяти типа NOR. Эти драйверы являются самостоятельными исполняемыми файлами, которые содержат код как файловой системы, так и флэш-устройства. Существуют версии драйвера файловой системы флэш-памяти для встраиваемых систем с различным оборудованием, а также для карт памяти стандарта PCMCIA.

Примечание. Файловые системы флэш-памяти не включают в себя элементы . и .. для текущего и родительского каталогов.

По соглашению об именах драйвер имеет название вида devf-система, где элемент система описывает встраиваемую систему. Например, драйвер devf-800fads предназначен для демонстрационной платы 800FADS PowerPC. Более подробные сведения об этих драйверах см. в описаниях утилит devf-* в справочнике по утилитам.

Сведения о работе операционной системы ЗОСРВ «Нейтрино» с файловыми системами флэш-памяти см. в следующих источниках:

- в описаниях утилит mkefs и flashctl в справочнике по утилитам;
- в главе 9 руководства по системной архитектуре;
- в электронном руководстве "Building Embedded Systems" справочной системы QNX SDP.

11.12. Файловая система CIFS

Протокол CIFS (Common Internet File System, общая файловая система Интернета) обеспечивает клиентской рабочей станции прозрачный сетевой доступ к Windows- и UNIX-системам с SMB-сервером. Этот протокол раньше назывался SMB (Server Message Block, блок серверных сообщений) и предназначен для контролируемого доступа к ресурсам локальной сети. Запросы доступа к файлам, которые посылает клиент, преобразуются в запросы протокола CIFS и передаются на сервер по сети. Сервер получает запрос, выполняет фактическую операцию над файловой системой и посылает ответ клиенту. Протокол CIFS функционирует на основе стека протоколов TCP/IP и использует протокол DNS.

Администратор файловой системы fs-cifs является клиентом протокола CIFS и работает по протоколу TCP/IP. Чтобы воспользоваться этим администратором, необходимо наличие SMB-сервера и действительного регистрационного имени на нем. Утилита fs-cifs предназначена в основном для использования в качестве клиента для связи с компьютерами с операционной системой Windows, хотя она работает с любыми SMB-серверами, например OS/2 Peer, LAN Manager и SAMBA.

Администратору файловой системы fs-cifs необходим транспортный уровень TCP/IP. Этот уровень предоставляется администратором сетевого ввода/вывода io-pkt*.

Дополнительные сведения о паролях и несколько связанных с ними примеров см. в разделе fs-cifs в справочнике по утилитам.

Если необходимо запускать файловую систему CIFS при загрузке компьютера, следует добавить соответствующую команду в файл /etc/host_cfg/$HOSTNAME/rc.d/rc.local или /etc/rc.d/rc.local. Более подробную информацию см. в описании файла /etc/rc.d/rc.sysinit в раздел 8.

11.13. Файловая система NFS

Протокол NFS (Network File System, сетевая файловая система) представляет собой TCP/IP-приложение, которое поддерживает сетевые файловые системы. Оно обеспечивает прозрачный межсетевой доступ к разделяемым файловым системам.

Протокол NFS позволяет клиентской рабочей станции работать с файлами, которые расположены на серверах с различными NFS-совместимыми операционными системами. Клиентские вызовы доступа к файлам преобразуются в запросы протокола NFS (см. спецификации RFC 1094 и RFC 1813) и передаются на сервер по сети. Сервер получает запрос, выполняет фактическую операцию над файлом и отсылает ответ клиенту.

По существу, протокол NFS позволяет включать удаленные файловые системы или их фрагменты в локальное пространство имен. Каталоги удаленных систем представляются как часть локальной файловой системы, а все утилиты перечисления файлов и управления ими (например, ls, cp и mv) работают с удаленными файлами так же, как с локальными.

Файловая система NFS допускает использование в имени файла тех же символов, что и файловая система QNX 4 (см. подразд. "Имена файлов" ранее в этом разделе).

Настройка NFS

Протокол NFS состоит из:

- клиента, который запрашивает включение удаленной файловой системы в локальное пространство имен;
- сервера, который отвечает на запросы клиентов и обеспечивает клиентам доступ к файловым системам как к точкам монтирования NFS.

Примечание. Процедуры, которые используются в операционной системе ЗОСРВ «Нейтрино» для настройки клиентов и серверов, могут отличаться от аналогичных процедур в других операционных системах. Для настройки клиентов и серверов в операционной системе, отличной от ЗОСРВ «Нейтрино», следует обратиться к документации ее поставщика, а также изучить сценарии инициализации, чтобы определить, как в этой системе запускаются программы.

Фактически работа по преобразованию обобщенных методов доступа к файлам, которые предоставляют серверы, в методы доступа к файлам, которые полезны приложениям и пользователям, выполняется клиентами протокола NFS.

Чтобы запускать файловую систему NFS при загрузке компьютера, следует поместить соответствующую команду в файл /etc/host_cfg/$HOSTNAME/rc.d/rc.local или /etc/rc.d/rc.local. Более подробную информацию см. в описании файла /etc/rc.d/rc.sysinit в разделе 8.

NFS-сервер

NFS-сервер обрабатывает запросы NFS-клиентов, которые пытаются получить доступ к файловым системам как к точкам монтирования NFS. Для того чтобы сервер работал, необходимо запустить следующие программы (табл. 11.1).

Таблица 11.1

Имя	**Назначение**
rpcbind	Сервер протокола RPC (Remote Procedure Call, удаленный вызов процедур)
nfsd	Сервер NFS и сервис mountd

Сервер rpcbind сопоставляет номера портов протоколов TCP и UDP номерам и версиям программ RPC. Клиенты могут выполнять RPC-вызовы только в том случае, если на сервере работает утилита rpcbind.

Сервис nfsd считывает файл /etc/exports, который содержит список файловых систем, доступных для экспорта, и необязательный список клиентов, которые могут экспортировать эти файловые системы. Если ни один клиент не указан, доступ к файловой системе предоставляется любому клиенту.

Сервис nfsd обслуживает запросы монтирования файловой системы NFS и запросы, которые адресованы самой файловой системе NFS, в соответствии с файлом exports. При загрузке программа nfsd считывает файл /etc/exports.имя_хоста или при его отсутствии файл /etc/exports, чтобы определить, какие точки монтирования требуют обслуживания. Изменения, которые вносятся в этот файл, начинают действовать только после перезапуска сервиса nfsd.

NFS-клиент

NFS-клиент запрашивает включение файловой системы, которую экспортирует NFS-сервер, в свое локальное пространство имен. Для того чтобы клиент работал, необходимо сначала запустить вторую или третью версию администратора файловой системы NFS (fs-nfs2 или fs-nfs3). Во второй версии описатель файла представляет собой 32-байтовый массив фиксированного размера, а в третьей версии — 64-байтовый массив переменной длины.

Примечание. При возможности вместо fs-nfs2 следует использовать fs-nfs3.

Администраторы файловой системы fs-nfs2 и fs-nfs3 также являются клиентскими сервисами протоколов NFS2 и NFS3 и работают по протоколу TCP/IP. Для использования этих администраторов необходим NFS-сервер и функционирующий транспортный уровень стека протоколов TCP/IP, например, тот, который обеспечивает администратор сетевого ввода/вывода io-net с модулем npm-ttcpip.so. Кроме того, администраторам fs-nfs2 и fs-nfs3 требуются модули socket.so и libc.so.

Создавать точки монтирования NFS можно с помощью команды mount, указывая nfs в качестве типа файловой системы и -o ver3 в качестве параметра. Чтобы создавать точки монтирования таким способом, необходимо предварительно запустить утилиту fs-nfs2 или fs-nfs3. Если администратор fs-nfs2 или fs-nfs3 запускается без аргументов, он работает в фоновом режиме и позволяет использовать команду mount.

Для подачи запроса клиент применяет утилиту mount так, как показано в следующих примерах:

- монтирование клиентской файловой системы NFS 2 (утилита fs-nfs2 должна быть предварительно запущена):

```
mount -t nfs 10.1.0.22:/home /mnt/home
```

- монтирование клиентской файловой системы NFS 3 (утилита fs-nfs3 должна быть предварительно запущена):

```
mount -t nfs -o ver3 server_node:/qnx_bin /bin
```

В первом примере клиент запрашивает монтирование каталога /home, который находится на хосте с заданным IP-адресом, в локальное пространство имен как каталог /mnt/home. Во втором примере протокол NFS версии 3 используется для сетевой файловой системы.

Следующая командная строка запускает и монтирует клиента:

```
fs-nfs3 10.7.0.197:/home/bob /homedir
```

Примечание. Несмотря на то, что файловая система NFS 2 появилась раньше стандарта POSIX, она была создана для имитации семантики файловой системы UNIX и относительно схожа со стандартом POSIX.

11.14. Файловая система UDF

Файловая система UDF (Universal Disk Format) обеспечивает доступ к таким записываемым носителям как CD, CD-R, CD-RW и DVD. Она предназначена для видео DVD, но может использоваться для резервного копирования данных на CD и других задач.

Файловая система UDF реализуется разделяемым объектом fs-udf.so. Драйверы devb-* автоматически загружают fs-udf.so при монтировании файловой системы UDF.

11.15. Файловые системы HFS и HFS Plus

Файловые системы HFS (Hierarchical File System) и HFS Plus используются в системах Apple Macintosh.

Доступ по чтению к файловым системам HFS и HFS Plus в системах ЗОСРВ «Нейтрино» обеспечивается разделяемым объектом fs-mac.so. Этот ресурс-менеджер способен распознавать следующие варианты: HFS, HFS Plus, HFS Plus в «обертке» HFS, HFSX и гибрид HFS/ISO-9660. Он может так же распознавать HFSJ (HFS Plus с журналированием), но только только в том случае, когда журнал «чистый», а не «грязный» вследствие некорректного останова системы. В традиционной таблице таблице файловых систем персональных ЭВМ для HFS используется код 175.

Драйверы devb-* автоматически загружают fs-mac.so при монтировании файловой системы HFS и HFS Plus.

11.16. Файловая система NTFS

Файловая система NTFS используется в системах Microsoft Windows NT и более новых ОС корпорации Microsoft. Доступ по чтению к файловым системам NTFS в системах ЗОСРВ «Нейтрино» обеспечивается разделяемым объектом fs-nt.so.

Драйверы devb-* автоматически загружают fs-nt.so при монтировании файловой системы NTFS. Если необходимо, что бы модуль fs-nt.so эмулировал элементы каталогов «.» и «..», необходимо задать ему опцию dots=on. По умолчанию данные элементы не эмулируются.

11.17. Файловая система с распаковкой сжатых данных "на лету"

Операционная система ЗОСРВ «Нейтрино» обеспечивает виртуальную файловую систему с распаковкой сжатых данных "на лету" (inflator). Она представляет собой администратор ресурсов, который находится перед другими файловыми системами и распаковывает файлы, ранее сжатые с помощью утилиты deflate.

Обычно утилита inflator используется в случае, если базовой файловой системой является файловая система флэш-памяти. С помощью программы inflator можно увеличить эффективный объем флэш-памяти почти в 2 раза.

Более подробные сведения см. в справочнике по утилитам.

11.18. Устранение неполадок

Далее рассмотрены некоторые проблемы, которые могут возникать при работе с файловыми системами.

- как сделать раздел флэш-памяти доступным только для чтения?

Нужно демонтировать, а затем заново смонтировать раздел следующим образом:

`flashctl -p` специальный_файл_устройства -u

`mount -t flash -r` специальный_файл_устройства /точка_монтирования

где параметр специальный_файл_устройства обозначает раздел (например, /dev/fs0px).

- как определить, какие драйверы работают в текущий момент?

Создать список точек монтирования:

```
find /proc/mount \
-name '[-0-9]*,[-0-9]*,[-0-9]*,[-0-9]*,[-0-9]*' \
-prune -print >mountpoints
```

Отобразить драйверы:

```
cut -d, -f2 < mountpoints | sort -nu > pidlist
```

Отобразить точки монтирования для заданного идентификатора процесса:

```
grep pid mountpoints
```

Отобразить идентификатор, длинное имя и обработчики прерываний каждого из процессов, указанных в списке pidlist:

```
xargs -i pidin -p {} -F "%a %n %Q" < pidlist | less
```

Отобразить точки монтирования процесса с идентификатором pid:

```
grep pid mountpoints
```

Отобразить дату указанного драйвера:

```
use -i имя_драйвера
```

Эта процедура схожа с алгоритмом команды driverquery операционной системы Windows XP и отображает только драйверы (т. е. программы, которые имеют точки монтирования в пространстве путевых имен), работающие в текущий момент, а не драйверы, установленные в операционную систему.

12. Прозрачная распределенная обработка с помощью Qnet

12.1. Что такое Qnet?

Собственная сеть ЗОСРВ «Нейтрино» представляет собой группу соединенных между собой рабочих станций, каждая из которых работает под управлением операционной системы ЗОСРВ «Нейтрино». В этой сети программа имеет прозрачный доступ к любому ресурсу, который расположен на любом другом узле (компьютере или рабочей станции) локальной подсети, будь то файл, устройство или процесс. Собственная сеть ЗОСРВ «Нейтрино» даже позволяет запускать программы на других узлах.

В сущности, протокол Qnet прозрачно распространяет действие механизма межзадачного взаимодействия (IPC) на сеть микроядер, используя принцип передачи сообщений ЗОСРВ «Нейтрино» для реализации собственной сети.

Записи всех узлов локальной подсети, на которых работает протокол Qnet, расположены в пространстве имен /net (в операционной системе QNX 4 для обращения к другому узлу применяется двойная косая черта, за которой следует номер узла).

Более подробные сведения о протоколе Qnet см. в главе 13 руководства "Описание применения. Часть 1. Системная архитектура" КПДА.10964-01 31 01".

12.2. Когда использовать протокол Qnet?

Когда следует использовать протокол Qnet, а когда — TCP или какой-либо другой протокол? Все зависит от того, какие компьютеры требуется объединить в сеть.

Протокол Qnet предназначен для сети доверенных компьютеров, которые работают под управлением операционной системы ЗОСРВ «Нейтрино» и имеют один и тот же порядок следования байтов. В этих условиях компьютеры могут совместно использовать ресурсы с небольшими накладными расходами. Протокол

Qnet дает возможность пользоваться утилитами операционной системы ЗОСРВ «Нейтрино» (cp, mv и т. д.) для работы с файлами, которые находятся в любом месте сети Qnet так же, как если бы они находились на локальном компьютере.

Поскольку протокол Qnet предназначен для группы доверенных компьютеров (например, входящих в состав встроенной системы), он не выполняет аутентификацию запросов. Защита файлов осуществляется с помощью обычных прав доступа, которые применяются к пользователям и группам (см. подразд. "Владение файлами и права доступа" раздела 6), однако ключи maproot и mapany позволяют в некоторых пределах управлять действиями удаленных пользователей на компьютере. В отличие от NFS, протокол Qnet является протоколом с установлением соединения; сообщения о сетевых ошибках передаются клиентскому процессу.

Протокол TCP/IP предназначен для компьютеров, которые связаны между собой не столь тесно и могут работать под управлением разных операционных систем. Для управления доступом к компьютеру протокол TCP/IP выполняет процедуру аутентификации; он полезен для соединения с компьютерами, которые не обязательно являются доверенными. Протокол TCP/IP используется как основа для специализированных протоколов, например FTP и Telnet, и обеспечивает высокую производительность потоковой передачи данных. Более подробные сведения см. в разделе 13.

Протокол NFS предназначен для выполнения операций над файловыми системами на любых хостах с любым порядком следования байтов и пользуется широкой поддержкой. NFS является протоколом без установления соединения; сервер можно остановить и перезапустить, а клиент автоматически возобновляет взаимодействие с сервером. Кроме того, протокол NFS использует аутентификацию и управляет доступом к каталогам. Более подробные сведения см. в подразд. "Файловая система NFS" раздела 11.

12.3. Правила именования узлов

Для разрешения имен узлов протокол Qnet использует следующие правила.

- имя узла (node name) – символьная строка, идентифицирующая узел, с которым осуществляется связь. Это имя должно быть уникальным в домене и не может содержать косых черт и точек;

По умолчанию именем узла является значение конфигурационной строки _CS_HOSTNAME. Если хост имеет имя localhost, которое назначается по умолчанию при первой загрузке, то протокол Qnet обеспечивает связь между узлами, используя имя хоста на основе MAC-адреса сетевого адаптера.

- домен узла (node domain) – символьная строка, которая добавляется в конец имени узла библиотекой npm-qnet.so. Имя и домен узла в совокупности должны составлять строку, которая является уникальной для всех узлов, взаимодействующих друг с другом по сети. По умолчанию домен узла задается значением конфигурационной строки _CS_DOMAIN;

- полностью определенное имя узла (fully qualified node name, FQNN) — строка, которая формируется путем объединения имени и домена узла. Например, если узел имеет имя matvei и доменное имя kpda.ru, то полностью определенным именем узла является matvei.kpda.ru;

- сетевой каталог (network directory) — каталог в пространстве путевых имен, который создается библиотекой npm-qnet.so. С каждым сетевым каталогом (их может быть несколько на одном узле) связан домен узла. Сетевым каталогом по умолчанию является /net; он используется в примерах, которые приведены в этом разделе.

> **Примечание**. Элементы каталога /net, которые соответствуют узлам, расположенным в том же домене, что и локальный компьютер, не включают в себя доменное имя. Например, если локальный компьютер находится в домене kpda.ru, то элемент узла matvei имеет вид /net/matvei. Если компьютер находится в другом домене, то этот элемент имеет вид /net/matvei.kpda.ru.

- разрешение имени (name resolution) — процесс, с помощью которого модуль lsm-qnet.so преобразует полностью определенное имя узла в список адресов назначения, доступных транспортному уровню;

- разрешитель имени (name resolver) — элемент кода, который реализует конкретный метод преобразования полностью определенного имени узла в список адресов назначения. Каждый сетевой каталог имеет список разрешителей имен, которые по очереди используются для разрешения полностью определенного имени узла. Разрешителем имени по умолчанию является протокол обнаружения узлов (Node Discovery Protocol, NDP).

12.4. Программные компоненты сети Qnet

Для функционирования сети Qnet наряду с оборудованием требуются следующие программные компоненты (рис. 12.1):

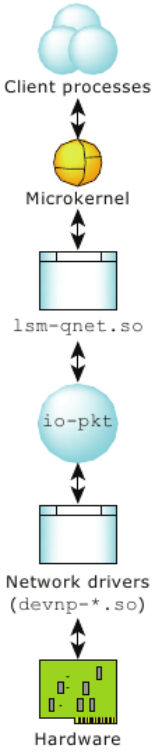


Рис. 12.1. Компоненты сети Qnet

- io-pkt* — администратор, который обеспечивает поддержку динамически загружаемых сетевых модулей;
- сетевые драйверы (devn-*, devnp-*) — администраторы, которые образуют интерфейс с оборудованием;

- lsm-qnet.so — собственный сетевой администратор операционной системы ЗОСРВ «Нейтрино», который реализует протоколы Qnet.

12.5. Запуск Qnet

Протокол Qnet можно запустить одним из следующих способов:

- создать файл useqnet и перезагрузить компьютер;

или

- явным образом запустить администратор сети, протоколы и драйверы, как описано далее.

Примечание. Если протокол Qnet работает на локальном компьютере, то пользователь любого компьютера сети с работающим протоколом Qnet может просматривать файлы и процессы локального компьютера при наличии прав доступа к ним. Более подробные сведения см. в следующих источниках:

- подразд. "Владение файлами и права доступа" раздела 6;
- подразд. "Qnet" раздела 19.

Создание файла useqnet

Чтобы автоматически запускать протокол Qnet при загрузке системы, следует войти в нее как пользователь root и создать пустой файл useqnet следующим образом:

```
touch /etc/system/config/useqnet
```

Если этот файл существует, сценарий /etc/system/sysinit запускает протокол Qnet при загрузке компьютера. Чтобы передать протоколу Qnet какие-либо параметры, следует отредактировать файл sysinit, изменив следующие его строки:

```
# Запустить протокол qnet, если пользователь включил его.
if test -r /etc/system/config/useqnet -a -d /dev/io-net; then
        mount  -Tio-pkt lsm-qnet.so
fi
```

Например, при работе с ненадежным оборудованием целесообразно включить проверку циклического избыточного кода пакетов. Для этого приведенные выше строки необходимо изменить следующим образом:

```
# Запустить протокол qnet, если пользователь включил его.
if test -r /etc/system/config/useqnet -a -d /dev/io-net; then
        mount  -Tio-pkt -o do_crc=1 lsm-qnet.so
fi
```

Более подробные сведения о том, что происходит при загрузке системы, см. в разделе 8.

Запуск сетевого администратора, протоколов и драйверов

Администратор io-pkt* — это процесс, который выполняет ключевую функцию загрузки набора разделяемых объектов. Он является основой всего стека протоколов и обеспечивает передачу данных между модулями. Множество разделяемых объектов собственной сети ЗОСРВ «Нейтрино» включает в себя библиотеку lsm-qnet.so и сетевые драйверы, например devn-ppc800-ads.so. Разделяемые объекты организованы в виде иерархии, где вершиной является конечный пользователь, а нижним уровнем – оборудование.

Предупреждение. Программа распознавания устройств автоматически запускает процесс io-pkt* при загрузке компьютера и загружает драйверы обнаруженных устройств. Можно запустить несколько экземпляров администратора io-pkt*, однако для этого необходима специальная настройка. Если требуется запустить процесс io-pkt* вручную, то сначала следует завершить работающий процесс io-pkt* с помощью команды slay.

Процесс io-pkt* можно запустить из командной строки, указав ему, какие драйверы и протоколы следует загрузить:

```
$ io-pkt* -del900 -p npm-qnet &
```

В этом примере администратор io-pkt-v4 загружает Ethernet-драйвер devn-el900.so и стек протоколов Qnet.

Модули можно запускать и останавливать динамически при помощи команд mount и umount следующим образом:

```
$ io-pkt-v6-hc &
$ mount -Tio-pkt devn-el900.so
$ mount -Tio-pkt lsm-qnet.so
```

Чтобы выгрузить драйвер, следует ввести команду:

```
umount /dev/io-net/en0
```

Примечание. Демонтировать стеки протоколов, такие как TCP/IP и Qnet, нельзя.

12.6. Просмотр сетевого окружения

После запуска протокола Qnet каталог /net содержит элементы для всех других узлов локальной подсети, на которых работает протокол Qnet. Доступ к файлам и процессам, которые расположены на других компьютерах, можно осуществлять так, как будто эти файлы и процессы находятся на локальном компьютере (по крайней мере, в той степени, в которой позволяют права доступа).

Например, чтобы отобразить содержимое файла, который находится на другом компьютере, можно воспользоваться утилитой less, указав путь с префиксом /net:

```
less /net/alonzo/etc/TIMEZONE
```

Чтобы считать системную информацию обо всех удаленных узлах, которые перечислены в каталоге /net, следует воспользоваться утилитой pidin с аргументом net:

```
$ pidin net
```

Можно вызвать утилиту pidin с параметром -n, чтобы получить информацию о процессах на другом компьютере:

```
pidin -n alonzo | less
```

Можно даже запустить процесс на другом компьютере, используя собственную консоль для ввода и вывода, с помощью команды on с параметром -f:

```
on -f alonzo date
```

Заполнение каталога /net

После загрузки и запуска Qnet вместе с сетевым драйвером, если приложения на узле не пытаются взаимодействовать с другими приложениями через Qnte, то каталог /net будет медленно заполняться остальными узлами по мере получения от них сведений, отправленных ими в широковещательном режиме.

По умолчанию для этого задан интервал 30 секунд, и он может изменяться с помощью опции auto_add=X модуля lsm-qnet.so. Другими словами, через 30 секунд после загрузки каталог /net, возможно, будет заполнен настолько, насколько это возможно.

Примечание. Для взаимодействия с удаленным узлом не требуется ждать 30 секунд; приложения могут обращаться через Qnet к удаленным узлам сразу после инициализации Qnet и сетевого драйвера.

Если существует некоторый элемент каталога /net, то это значит только то, что Qnet отображает представленное в ASCII-символах текстовое имя узла на MAC-адрес Ethernet. Это лишь немного замедляет процесс распознавания имен, но это позволяет пользователям увидеть, какие еще узлы доступны в данной сети.

Элементы каталога /net не удаляются пока кто-либо пытается их использовать, поэтому они могут быть некорректными.

Например, какой-либо узел мог загрузиться час назад, проработать одну минуты и завершить работу. Он будет по-прежнему иметь запись в каталогах тех узлов Qnet, которые не взаимодействовали с ним. На тех узлах, которые взаимодействовали с ним установив соединение, записи будут удаляться после истечения таймаутов сесий.

Для удаления неправильных элементов из /net можно воспользоваться командой:

```
ls -l /net &
```

Для полной очистки /net необходимо выполнить команду:

```
rmdir /net/*
```

12.7. Устранение неполадок

Для того чтобы собственная сеть операционной системы ЗОСРВ «Нейтрино» функционировала, все программные компоненты протокола Qnet должны корректно взаимодействовать с оборудованием. Если сеть Qnet не работает, то при поиске проблем с оборудованием и сетью можно воспользоваться различными Qnet-утилитами для получения диагностической информации.

- работает ли протокол Qnet?

Протокол Qnet создает каталог /net. Чтобы убедиться в том, что этот каталог существует, следует воспользоваться командой

```
$ ls /net
```

Если каталог /net не содержит ни одного подкаталога, то протокол Qnet не работает. Этот каталог должен содержать как минимум один элемент с именем локального компьютера, которое выводит команда hostname. Если используется привязка протокола Ethernet, то все остальные доступные компьютеры также отображаются в каталоге /net. Пример:

```
joseph/ eileen/
```

- работают ли процесс io-pkt* и драйверы?

Как было отмечено ранее, процесс io-pkt* является связующим звеном между драйверами и протоколами. Чтобы проверить наличие неполадок в нем, следует воспользоваться командой pidin:

```
$ pidin -P io-pkt-v4-hc mem
```

Затем следует выполнить поиск разделяемого объекта протокола Qnet в полученном выводе:

```
pid   tid  name      prio STATE     code data stack
118802 1 sbin/io-pkt-v4-hc  21o SIGWAITINFO  876K            672K
4096(516K)*
```

```
118802 2 sbin/io-pkt-v4-hc  21o RECEIVE  876K 672K   8192(132K)
118802 3 sbin/io-pkt-v4-hc  21r RECEIVE  876K 672K   4096(132K)
118802 4 sbin/io-pkt-v4-hc  21o RECEIVE  876K 672K   4096(132K)
118802 5 sbin/io-pkt-v4-hc  20o RECEIVE  876K 672K   4096(132K)
118802 6 sbin/io-pkt-v4-hc 10o RECEIVE   876K 672K   4096(132K)
libc.so.2             @b0300000     436K 12K
devnp-shim.so              @b8200000     28K 4096
devn-pcnet.so              @b8208000     40K 4096
lsm-qnet.so                @b8213000     168K 36K
```

Если в выводе присутствует разделяемый объект lsm-qnet.so, то протокол Qnet работает.

- работает ли сетевая плата?

Чтобы определить, работает ли сетевая плата (другими словами, выполняет ли она прием и передачу пакетов), следует воспользоваться командой nicinfo. Переменная окружения **PATH** пользователя root включает в себя каталог, в котором находится исполняемый файл nicinfo; всем остальным пользователям необходимо указывать полный путь к этому файлу:

```
$ /usr/sbin/nicinfo
```

Теперь можно извлечь диагностическую информацию из следующего вывода:

```
en0:
AMD PCNET-32 Ethernet Controller
Physical Node ID ..........................000C29 DD3528
Current Physical Node ID ..................000C29 DD3528
Current Operation Rate .....................10.00 Mb/s
Active Interface Type ......................UTP
Maximum Transmittable data Unit ............1514
Maximum Receivable data Unit ...............1514
Hardware Interrupt .........................0x9
I/O Aperture ...............................0x1080 - 0x10ff
Memory Aperture ............................0x0
Promiscuous Mode ...........................Off
Multicast Support ..........................Enabled

Packets Transmitted OK ..................... 588
```

```
Bytes Transmitted OK ..................... 103721
Memory Allocation Failures on Transmit ..... 0

Packets Received OK ...................... 11639
Bytes Received OK ........................ 934712
Memory Allocation Failures on Receive ...... 0

Single Collisions on Transmit ..............0
Deferred Transmits .........................0
Late Collision on Transmit errors ..........0
Transmits aborted (excessive collisions) ....0
Transmit Underruns .........................0
No Carrier on Transmit .....................0
Receive Alignment errors ...................0
Received packets with CRC errors ...........0
Packets Dropped on receive .................0
```

Следует обратить особое внимание на значения счетчиков Total Packets Txd OK и Total Packets Rxd OK. Если они равны нулю, то, возможно, драйвер не работает или отсутствует подключение к сети. Следует проверить запись Current Operation Rate и убедиться в том, что драйвер корректно автоматически определил скорость передачи данных.

- как получить диагностическую информацию?

Диагностическая информация находится в файле /proc/qnetstats. Если этот файл не существует, протокол Qnet не работает.

Файл qnetstats содержит много диагностической информации, которая имеет смысл для разработчиков, использующих протокол Qnet, но бесполезна для других пользователей. Тем не менее, можно воспользоваться утилитой grep, чтобы извлечь определенные поля из файла /proc/qnetstats:

```
# cat /proc/qnetstats | grep "compiled"

**** Qnet compiled on Jun 3 2008 at 14:08:23 running on EAdd3528
```

Еще один пример:

```
# cat /proc/qnetstats | grep -e "ok" -e "bad"
txd  ok     930
txd  bad    0
```

```
rxd  ok     2027
rxd  bad dr 0
rxd  bad L4 0
```

- уникально ли имя хоста?

Чтобы узнать имя хоста, следует воспользоваться утилитой hostname. Сеть Qnet работает только в случае, если имя хоста является уникальным.

- находятся ли узлы в одном и том же домене?

Если узлы находятся в разных доменах, то для доступа к узлу необходимо указывать домен. Пример:

```
ls /net/kostya.kpda.ru
```

13. Сеть TCP/IP

13.1. Обзор протокола TCP/IP

Термин "TCP/IP" обозначает два разных протокола — TCP и IP. Поскольку эти протоколы обычно используются совместно, понятие "TCP/IP" стало стандартным в современном Интернете. В сущности, оно характеризует сетевые средства связи, в которых данные передаются по IP-сетям с помощью протокола TCP.

Эта глава включает в себя сведения о том, как настроить протокол TCP/IP в сети на основе ЗОСРВ «Нейтрино». Кроме того, она содержит информацию о разрешении неполадок и другие подробности, которые важны с точки зрения системного администрирования. Сеть TCP/IP на основе ЗОСРВ «Нейтрино» обеспечивает доступ к ресурсам, которые расположены на любом удаленном компьютере, поддерживающем стек протоколов TCP/IP.

Клиенты и серверы

Существуют два типа TCP/IP-хостов: клиенты и серверы. Клиент запрашивает службу TCP/IP, а сервер предоставляет ее. При планировании сети необходимо решить, какие хосты будут серверами, а какие – клиентами.

Например, компьютер, который посылает запросы на соединение через протокол telnet, должен быть настроен как клиент, а компьютер, который принимает эти запросы – как сервер.

Хосты и шлюзы

В терминологии протокола TCP/IP компьютеры, которые доступны по сети, называются хостами (host) или шлюзами (gateway).

- хост – узел с работающим протоколом TCP/IP, который не пересылает IP-пакеты в другие TCP/IP-сети. Как правило, хост имеет единственный интерфейс (сетевую плату) и является адресатом или источником TCP/IP-пакетов;

- шлюз – узел с работающим протоколом TCP/IP, который пересылает IP-пакеты в другие TCP/IP-сети в соответствии со своей таблицей маршрутизацией. Эти системы включают в себя два и более сетевых интерфейса. Если TCP/IP-хост имеет доступ к Интернету, то в сети этого хоста должен присутствовать шлюз.

Примечание. Для того чтобы воспользоваться протоколом TCP/IP, необходимо знать IP-адрес локального хоста и IP-адрес хоста, к которому требуется подключиться. Обычно обращение к удаленному хосту осуществляется по текстовому имени, которое разрешается в IP-адрес с помощью сервера имен.

Серверы имен

Сервер имен (nameserver) представляет собой базу данных, которая содержит имена и IP-адреса хостов. Доступ к TCP/IP- и интернет-хостам обычно осуществляется через текстовое имя (например, **www.kpda.ru**) и механизм, который преобразует имя в IP-адрес (например, 209.226.137.1).

Самый простой способ создать такую привязку — воспользоваться таблицей в файле /etc/hosts. Этот метод эффективен в сетях небольшого и среднего масштаба. Если сеть включает в себя много хостов или не является внутренней, то необходимо использовать сервер имен (например, для подключения к поставщику услуг Интернета).

При подключении к TCP/IP-хосту с использованием имени у сервера имен запрашивается соответствующий IP-адрес, а затем осуществляется соединение с этим IP-адресом. Сервер имен может быть задан следующими способами:

- записью в конфигурационной строке _CS_RESOLVE, которая считывается из конфигурационного файла (по умолчанию /etc/net.cfg);
- записью в файле /etc/resolv.conf, например:

```
nameserver 10.0.0.2
nameserver 10.0.0.3
```

Для того чтобы сконфигурировать сеть и задать серверы имен, можно воспользоваться инструментом настройки протокола TCP/IP и коммутируемых

соединений графической оболочки Photon под названием phlip. Утилита phlip присваивает значение конфигурационной строке _CS_RESOLVE. Эту конфигурационную строку можно также задавать вручную. Если строка _CS_RESOLVE существует, то она всегда используется для поиска сервера имен вместо файла /etc/resolv.conf.

Более подробные сведения о поиске имен хостов и серверов имен в TCP/IP-сетях см. в разделах /etc/hosts и /etc/resolv.conf справочника по утилитам.

Примечание. Если сервер имен не отвечает, то существует таймаут по 1,5 минуты на каждый сервер. Этот таймаут не может быть изменен, но многие утилиты TCP/IP имеют опцию -n, которую можно использовать, что бы избежать поиска имени.

Маршрутизация

Маршрутизация определяет, как пакет передается адресату. Существуют следующие категории маршрутизации.

- минимальная маршрутизация.

Хост устанавливает соединения только с теми хостами, которые находятся в его сети. Например, минимальная маршрутизация используется в изолированной сети.

- статическая маршрутизация.

Если сеть включает в себя небольшое и постоянное число шлюзов, то можно настроить и сохранить таблицы маршрутизации протокола TCP/IP вручную с помощью команды route.

Эта конфигурация используется очень часто. Если хост имеет доступ к Интернету, то он обычно добавляет один статический маршрут, который называется маршрутом по умолчанию (default route). Этот маршрут направляет все TCP/IP-пакеты, которые отправляются данным хостом и не адресованы какому-либо хосту локальной сети, на шлюз, который обеспечивает доступ к Интернету.

- динамическая маршрутизация.

Если в сети существует более одного пути к одному и тому же хосту, то, возможно, требуется использовать динамическую маршрутизацию. При динамической маршрутизации информация об изменении состояния сети распространяется с помощью протоколов маршрутизации. Если необходимо реагировать на эти изменения, то следует запустить утилиту routed, которая реализует протоколы RIP (Routing Information Protocol, протокол маршрутной информации) и RIPv2.

Маршрутизацию и протоколы маршрутизации часто путают друг с другом. Маршрутизация в стеке TCP/IP определяется с помощью таблиц маршрутизации, а протоколы маршрутизации позволяют изменять эти таблицы.

13.2. Программные компоненты сети TCP/IP

Для использования протокола TCP/IP необходимы следующие программные компоненты (рис. 13.1):

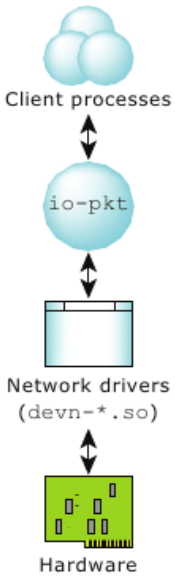
- io-pkt* — администратор, который обеспечивает поддержку динамически загружаемых сетевых модулей. Он включает полнофункциональный стек TCP/IP основанный на коде base.io-net операционной системы NetBSD;
- сетевые драйверы (devn-*, devnp-*) — администраторы, которые реализуют интерфейс с оборудованием.

Параметры конфигурации задаются с помощью утилит ifconfig и route.

Если используется протокол динамического конфигурирования хостов (Dynamic Host Configuration Protocol, DHCP), то, независимо от версии стека TCP/IP, можно воспользоваться утилитой dhcp.client для того, чтобы автоматически задать конфигурационные параметры, которые предоставляет DHCP-сервер.

Примечание. Программа распознавания устройств автоматически запускает администратор io-pkt* при загрузке операционной системы, а затем загружает полнофункциональный стек TCP/IP и запускает драйверы обнаруженных устройств. Если при загрузке требуется использовать компактный стек или задавать какие-либо параметры (например, включать протокол IPSec), то необходимо отредактировать файлы распознавания

устройств. Более подробные сведения и пример см. в



подразд. "Распознавание устройств" раздела 8.

Рис. 13.1. Компоненты протокола TCP/IP в операционной системе ЗОСРВ «Нейтрино»

Стек TCP/IP, основан на стеке TCP/IP операционной системы NetBSD и поддерживает схожие с ним возможности. Если для конфигурирования полнофункционального стека не используется утилита phlip (инструмент настройки протокола TCP/IP и коммутируемых соединений графической оболочки Photon), то необходимо воспользоваться утилитами ifconfig и route, как показано далее, поскольку конфигурировать полнофункциональный стек с помощью командно-строковых аргументов по аналогии с компактным стеком нельзя.

Для задания IP-адреса интерфейса следует воспользоваться утилитой ifconfig. Чтобы назначить сетевому интерфейсу IP-адрес 10.0.0.100, необходимо ввести команду:

```
ifconfig if_name 10.0.0.100
where if_name is the interface name that the driver uses.
```

Если требуется также указать шлюз, следует ввести команду route:

```
route add default 10.0.0.1
```

Эта команда устанавливает IP-адрес шлюза 10.0.0.1.

Чтобы затем просмотреть сетевую конфигурацию, следует воспользоваться командой netstat (команда netstat -in отображает информацию о сетевых интерфейсах):

```
Name Mtu   Network Address           Ipkts Ierrs Opkts Oerrs Coll
lo0  32976 <Link>                    0     0     0     0     0
lo0  32976 127     127.0.0.1         0     0     0     0     0
en0  1500  <Link>  00:50:da:c8:61:92 21    0     2     0     0
en0  1500  10      10.0.0.100        21    0     2     0     0
```

Чтобы отобразить информацию о таблице маршрутизации в полном стеке, следует воспользоваться командой netstat -rn. Ее результат выглядит так:

```
Routing tables

Internet:
Destination Gateway    Flags Refs Use Mtu Interface
default     10.0.0.1   UGS   0    0   —  en0
10          10.0.0.100 U     1    0   —  en0
10.0.0.100  10.0.0.100 UH    0    0   —  lo0
127.0.0.1   127.0.0.1  UH    0    0   —  lo0
```

Эта таблица показывает, что маршрут по умолчанию задан и его IP-адрес равен 10.0.0.1.

13.3. Запуск интернет-серверов

Если хост является сервером, то он запускает соответствующий сервис для обслуживания клиентских запросов. Сервер TCP/IP обычно запускает сервис inetd, который также называют "суперсервером Интернета". Сервис inetd можно запустить из файла rc.local (см. описание файла /etc/rc.d/rc.sysinit в разделе 8).

Предупреждение. Запуск сервиса inetd дает внешним пользователям возможность подключаться к локальному компьютеру, поэтому его ненадлежащая настройка способна вызвать проблемы с безопасностью.

В сети TCP/IP сервис inetd ожидает соединений через порты со стандартными номерами, которые указаны в файле /etc/inetd.conf. После получения запроса утилита inetd запускает соответствующий серверный сервис. Например, если клиент запрашивает удаленное подключение к хосту с помощью утилиты rlogin, то для обслуживания этого запроса сервер inetd запускает сервис удаленного подключения rlogind. В большинстве случаев клиентские запросы обслуживаются таким способом.

Сервисы, которые может запускать утилита inetd, задаются в конфигурационном файле суперсервера /etc/inetd.conf. Версия этого файла, которая входит в дистрибутив ЗОСРВ «Нейтрино», содержит все сервисы TCP/IP, имеющиеся в дистрибутиве, а также некоторые нестандартные службы утилиты pidin. Редактировать файл /etc/inetd.conf необходимо лишь тогда, когда требуется добавить или удалить определения сервисов. Утилита inetd считывает конфигурационную информацию из этого файла при запуске. Файл /etc/inetd.conf включает в себя следующие часто используемые сервисы.

- ftpd — передача файлов;
- rlogind — удаленный вход в систему;
- rshd — удаленный командный интерпретатор;
- telnetd — удаленный терминальный сеанс;
- tftpd — простейший протокол передачи файлов, который разработан Управлением перспективных исследовательских проектов (DARPA).

Примечание. Сервисы, которые перечислены в этом файле, не следует запускать вручную; они должны запускаться сервером inetd.

Запуск сервисов rshd и rlogind открывает доступ к локальному компьютеру извне. Для задания доверенных пользователей необходимо воспользоваться файлом /etc/hosts.equiv или ~/.rhosts, однако следует быть очень внимательным.

Существуют и другие резидентные сервисы, которые могут работать независимо от сервера inetd (их описания приведены в руководстве "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01):

- bootpd — сервер протокола интернет-загрузки;
- dhcpd — сервис протокола динамического конфигурирования хостов;
- lpd — сервис построчного принтера (см. раздел 14);
- mrouted — сервис протокола дистанционной векторной групповой многоадресной маршрутизации (Distance-Vector Multicast Routing Protocol, DVMRP);
- named — сервер доменных имен Интернета;
- ntpd — сервис сетевого протокола времени (Network Time Protocol);
- routed — сервис протоколов маршрутизации RIP и RIPv2;
- rwhod — база данных состояний системы;
- slinger — компактный Web-сервер;
- snmpd — SNMP-агент;
- nfsd — NFS-сервер.

Эти сервисы прослушивают свои TCP-порты и управляют собственными транзакциями. Они обычно запускаются при загрузке компьютера и затем работают постоянно. Тем не менее, для экономии системных ресурсов можно запускать сервис bootpd с помощью сервера inetd только при получении запроса на загрузку.

13.4. Запуск нескольких экземпляров стека TCP/IP

В некоторых ситуациях необходимо запускать несколько экземпляров стека протоколов TCP/IP.

- первый экземпляр стека TCP/IP запускается с помощью вызова утилиты io-pkt* следующим образом:

```
io-pkt-v4 -del900 pci=0x0
```

- второй экземпляр стека TCP/IP запускается с помощью вызова утилиты io-pkt* следующим образом:

```
io-pkt-v4 -i1 -del900 pci=0x1 -ptcpip prefix=/sock2
```

Чтобы получить PCI-индексы сетевых плат, можно воспользоваться командой pci -vvv. Если используются различные типы сетевых плат, то указывать PCI-индекс не обязательно.

Параметр -i во втором экземпляре стека TCP/IP указывает утилите io-pkt-v4 зарегистрироваться под префиксом io-pkt1. Параметр prefix указывает на то, что второй стек должен быть зарегистрирован под префиксом /sock2/dev/socket вместо префикса по умолчанию /dev/socket. Приложения, которым необходимо использовать второй стек TCP/IP, должны задать переменную окружения **SOCK**. Пример:

```
SOCK=/sock2 telnet 10.59
```

или

```
SOCK=/sock2 netstat -in
```

или

```
SOCK=/sock2 ifconfig en0 192.168.2.10
```

Если переменная **SOCK** не задана, то команда использует первый стек TCP/IP.

13.5. Динамически назначаемые параметры TCP/IP

Когда хост добавляется в сеть или подключается к Интернету, необходимо задать его адрес и некоторые другие конфигурационные параметры. Для этого существует несколько распространенных механизмов:

- поставщики сетевых услуг с коммутируемым доступом используют протокол соединения "точка-точка" (Point-to-Point Protocol, PPP);
- поставщики, которые предоставляют широкополосный доступ такими методами, как цифровая абонентская линия (Digital Subscriber Line, DSL) или кабельное подключение, используют протокол соединения "точка-точка" через среду Ethernet (Point-to-Point Protocol over Ethernet, PPPoE) или протокол DHCP;
- как правило, в корпоративных сетях используется протокол DHCP.

Серверы, которые реализуют эти протоколы в корпоративной сети, предоставляют клиенту IP-адрес, шлюз, маску сети, серверы имен и даже принтер. Пользователям не обязательно конфигурировать свои хосты вручную для того, чтобы использовать сеть.

Операционная система ЗОСРВ «Нейтрино» реализует еще один протокол автоматического конфигурирования под названием AutoIP, который является проектом комитета IETF по автоматической настройке. Этот протокол используется в небольших сетях для назначения хостам IP-адресов, локальных для канала (link-local). Протокол AutoIP самостоятельно определяет IP-адрес, локальный для канала, используя схему согласования с другими хостами и не обращаясь к центральному серверу.

Использование протокола PPPoE

Сокращение PPPoE расшифровывается как "Point-to-Point Protocol over Ethernet" (протокол соединения "точка-точка" через среду Ethernet). Этот протокол инкапсулирует данные для передачи через сеть Ethernet с мостовой топологией.

PPPoE представляет собой спецификацию подключения пользователей сети Ethernet к Интернету через широкополосное соединение, например, выделенную цифровую абонентскую линию, беспроводное устройство или кабельный модем. Использование протокола PPPoE и широкополосного модема обеспечивает пользователям локальной компьютерной сети индивидуальный аутентифицированный доступ к высокоскоростным сетям передачи данных.

Протокол PPPoE объединяет технологию Ethernet с протоколом PPP, что позволяет эффективно создавать отдельное соединение с удаленным сервером для каждого пользователя. Управление доступом, учет соединений и выбор поставщика услуг определяется для пользователей, а не для узлов сети. Преимущество этого подхода заключается в том, что ни телефонная компания, ни поставщик услуг Интернета не должен обеспечивать для этого какую-либо специальную поддержку.

В отличие от коммутируемых соединений, соединения через цифровую абонентскую линию и кабельный модем всегда активны. Поскольку физическое соединение с удаленным поставщиком услуг совместно используется несколькими пользователями, необходим метод учета, который регистрирует отправителей и адресатов трафика, а также производит начисления пользователям. Протокол PPPoE позволяет пользователю и удаленному узлу, которые участвуют в сеансе

связи, узнавать сетевые адреса друг друга во время начального обмена, который называется обнаружением (discovery). После того как сеанс между отдельным пользователем и удаленным узлом (например, поставщиком услуг Интернета) установлен, за этим сеансом можно вести наблюдение для того, чтобы производить начисления. Во многих домах, гостиницах и корпорациях общий доступ к Интернету предоставляется через цифровые абонентские линии с использованием технологии Ethernet и протокола PPPoE.

Соединение через протокол PPPoE состоит из клиента и сервера. Клиент и сервер работают с использованием любого интерфейса, который близок к спецификациям Ethernet. Этот интерфейс применяется для выдачи клиентам IP-адресов с привязкой этих IP-адресов к пользователям и, по желанию, к рабочим станциям, вместо аутентификации на основе только рабочей станции. Сервер PPPoE создает соединение "точка-точка" для каждого клиента.

Установка сеанса PPPoE

Для того чтобы создать сеанс PPPoE, следует воспользоваться сервисом pppoed. Модуль io-pkt-* предоставляет службы протокола PPPoE. Сначала необходимо запустить io-pkt-* с подходящим драйвером. Пример:

```
io-pkt-v6-hc -del900
```

Затем следует создать сеанс связи с любым сервером с помощью файла /etc/ppp/pppoe-up для того, чтобы запустить сервис pppd:

```
pppoed
```

После этого следует создать сеанс связи с сервером, который имеет имя PPPOE_GATEWAY:

```
pppoed name=PPPOE_GATEWAY
```

Когда сеанс PPPoE установлен, сервис pppoed использует сервис pppd для создания соединения "точка-точка" через этот сеанс. Сервис pppd получает локальную конфигурацию протокола TCP/IP от сервера (поставщика услуг Интернета).

Запуск соединения "точка-точка" через сеанс PPPoE

Для установления каналов связи "точка-точка" через протокол TCP/IP программе pppoed необходим сервис pppd. При запуске сервиса pppd необходимо воспользоваться несколькими параметрами, которые специфичны для его работы через сеанс PPPoE. Пример файла /etc/ppp/pppoe-up:

```
#!/bin/sh
pppd debug /dev/io-net/ppp_en -ac -pc -detach defaultroute \
require-ns mtu 1492 name username
```

Для работы сервиса pppd с сервисом pppoed обязательны следующие параметры.

- /dev/io-net/ppp_en – устройство, которое создает модуль io-pkt;
- -ac -pc — обязательные параметры, которые отключают сжатие пакетов;
- -detach — параметр, который не позволяет утилите pppd работать в режиме сервиса. Это дает программе pppoed возможность получить информацию о том, что сеанс утилиты pppd завершен. Данный параметр можно опустить, если воспользоваться параметром scriptdetach сервиса pppoed;
- mtu 1492 — следует задать максимальный размер передаваемых данных, который поддерживается протоколом PPPoE. Этот размер равен разности размера пакета Ethernet и размера служебных данных инкапсуляции протокола PPPoE.

Использование протокола DHCP

Хост TCP/IP использует протокол DHCP (Dynamic Host Configuration Protocol, динамический протокол конфигурирования хостов) для получения своих конфигурационных параметров (IP-адреса, шлюза, серверов имен и т. д.) у DHCP-сервера, который содержит конфигурационные параметры всех хостов сети.

Утилита dhcp.client, которая является DHCP-клиентом операционной системы ЗОСРВ «Нейтрино», получает эти параметры и автоматически конфигурирует хост для использования Интернета или локальной сети.

Если утилита dhcp.client не может применить конфигурационные параметры, которые предоставлены DHCP-сервером, она передает эти параметры сценарию,

который она исполняет. Этот сценарий можно использовать для применения любых параметров, которые не устанавливаются утилитой dhcp.client. Более подробные сведения см. в разделе о dhcp.client справочника по утилитам.

Использование протокола AutoIP

AutoIP представляет собой модуль, который необходимо смонтировать в администратор io-pkt*. Этот модуль используется для быстрого конфигурирования хостов в небольшой сети. Протокол AutoIP назначает своему интерфейсу IP-адрес, локальный для канала, из сети 169.254/16, если этот адрес не используется никаким другим хостом. Преимущество протокола AutoIP состоит в том, что необходимость в центральном конфигурационном сервере отсутствует. Хосты самостоятельно определяют, какие IP-адреса свободны для использования, и отслеживают возникновение конфликтов.

Обычно хост использует протоколы DHCP и AutoIP одновременно. Когда хост впервые подключается к сети, он не располагает информацией о наличии в сети DHCP-сервера. Если запустить программу dhcp.client с параметром –a (использовать IP-адрес как псевдоним), то интерфейсу одновременно присваивается IP-адрес, локальный для канала, и IP-адрес, который выдается протоколом DHCP. Если DHCP-сервер отсутствует, то наступает тайм-аут, в результате которого программа dhcp.client завершается и оставляет активным IP-адрес, локальный для канала. Если позднее DHCP-сервер становится доступным, то утилиту dhcp.client можно перезапустить и применить IP-адрес, который выдает протокол DHCP, без помех для TCP/IP-соединений, использующих IP-адрес, локальный для канала.

Одновременная активность адреса, который назначается протоколом DHCP, и IP-адреса, локального для канала, обеспечивает связь с хостами, которые имеют как IP-адреса, локальные для канала, так и обычные IP-адреса. Более подробные сведения см. в разделах о lsm-autoip.so и dhcp.client в руководстве "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01.

13.6. Устранение неполадок

Для обнаружения неполадок, которые возникают в сети TCP/IP (например, невозможность передачи пакетов по сети), необходимо использовать несколько утилит. Эти утилиты опрашивают хосты, серверы и шлюзы с целью получения диагностической информации для локализации неисправностей и отвечают на следующие типичные вопросы.

- работают ли администратор io-pkt* и драйверы?

Как было сказано ранее, администратор io-pkt* является связующим звеном между драйверами и протоколами. Чтобы проверить наличие неполадок в нем, следует воспользоваться командой pidin:

```
$ pidin -P io-pkt-v4 mem
```

Затем надо выполнить поиск разделяемого объекта протокола TCP/IP в выводе этой команды:

```
  pid tid name              prio STATE     code data   stack
126996   1 sbin/io-pkt-v4-hc     21o SIGWAITINFO   872K 904K     8192(516K)*
126996   2 sbin/io-pkt-v4-hc     21o RECEIVE         872K 904K   8192(132K)
126996   3 sbin/io-pkt-v4-hc     21r RECEIVE         872K 904K   4096(132K)
126996   4 sbin/io-pkt-v4-hc     21o RECEIVE         872K 904K   4096(132K)
126996   5 sbin/io-pkt-v4-hc     20o RECEIVE         872K 904K   4096(132K)
126996   6 sbin/io-pkt-v4-hc     9o RECEIVE     872K 904K   4096(132K)
      libc.so.3       @b0300000      444K 16K
      devnp-shim.so         @b8200000      28K 8192
      devn-epic.so          @b8209000      40K 4096
      lsm-qnet.so           @b8214000      168K 36K
```

Вывод должен содержать разделяемый объект сетевого драйвера (в данном случае это драйвер devnp-shim.so, который позволяет io-pkt использовать унаследованный из устаревшего стека io-net драйвер devn-epic.so).

- какова информация о серверах имен?

Для того чтобы получить информацию о серверах имен, следует воспользоваться командой:

```
getconf _CS_RESOLVE
```

Если конфигурационная строка не используется, то надо ввести команду:

```
cat /etc/resolv.conf
```

- как имена хостов связаны с IP-адресами?

Файл /etc/hosts содержит информацию, которая относится к известным хостам сети. Каждому хосту должна соответствовать одна строка, которая включает в себя следующие данные:

интернет_адрес официальное_имя_хоста псевдонимы

Чтобы отобразить содержимое этого файла, следует воспользоваться командой:

```
cat /etc/hosts
```

- как получить сведения о состоянии сети?

Сведения о состоянии сети можно получить с помощью следующих команд:

- netstat –in — перечисляет интерфейсы, в том числе их MAC- и IP-адреса;
- netstat –rn — отображает сетевые таблицы маршрутизации, которые определяют, как стек может связаться с другим хостом. Если путь к другому хосту не существует, то отображается сообщение об ошибке "no route to host";
- netstat –an — отображает информацию о входящих и исходящих TCP/IP-соединениях локальной системы. Эта информация включает в себя состояние соединения или количество данных, которые ожидают передачи по соединению. Кроме того, команда netstat –an отображает IP-адрес и порт локальной и удаленной стороны соединения.

Более подробные сведения о команде netstat см. в справочнике по утилитам.

- как убедиться в том, что локальный хост подключен к другим хостам?

Чтобы определить, подключен ли локальный хост к другим хостам, следует воспользоваться утилитой ping. Пример:

```
ping isp.com
```

Если подключение существует, утилита ping отображает приблизительно следующее:

```
PING isp.com (10.0.0.1): 56 data bytes
64 bytes from 10.0.0.1: icmp seq=0 ttl=255 time=0 ms
64 bytes from 10.0.0.1: icmp seq=1 ttl=255 time=0 ms
64 bytes from 10.0.0.1: icmp seq=2 ttl=255 time=0 ms
64 bytes from 10.0.0.1: icmp seq=3 ttl=255 time=0 ms
64 bytes from 10.0.0.1: icmp seq=4 ttl=255 time=0 ms
64 bytes from 10.0.0.1: icmp seq=5 ttl=255 time=0 ms
64 bytes from 10.0.0.1: icmp seq=6 ttl=255 time=0 ms
```

Это сообщение продолжает выводиться до тех пор, пока утилита ping не завершится, например, с помощью нажатия комбинации клавиш <Ctrl>+<C>.

- как отобразить информацию о контроллере интерфейса?

Следует воспользоваться командой nicinfo:

```
/usr/sbin/nicinfo устройство
```

Примечание. Все пользователи, кроме root, должны указывать полный путь к утилите nicinfo.

Эта утилита отображает информацию о соединении заданного сетевого интерфейса или интерфейса /dev/io-net/en0 (если интерфейс не указан). Эта информация включает в себя число переданных и полученных пакетов, коллизий и других ошибок:

```
3COM (90xC) 10BASE-T/100BASE-TX Ethernet Controller
Physical Node ID ................. 000103 E8433F
Current Physical Node ID ......... 000103 E8433F
Media Rate ....................... 10.00 Mb/s half-duplex UTP
MTU .............................. 1514
Lan .............................. 0
I/O Port Range ................... 0xA800 -> 0xA87F
Hardware Interrupt ............... 0x7
Promiscuous ...................... Disabled
Multicast ........................ Enabled

Total Packets Txd OK ............. 1585370
Total Packets Txd Bad ............ 9
```

```
Total Packets Rxd OK ..... 11492102
Total Rx Errors ..... 0

Total Bytes Txd ..... 102023380
Total Bytes Rxd ..... 2252658488

Tx Collision Errors ..... 39598
Tx Collisions Errors (aborted) ... 0
Carrier Sense Lost on Tx ..... 0
FIFO Underruns During Tx ..... 0
Tx deferred ..... 99673
Out of Window Collisions ..... 0
FIFO Overruns During Rx ..... 0
Alignment errors ..... 0
CRC errors ..... 0
```

14. Печать

14.1. Обзор систем печати

Самый простой способ напечатать файл – это послать его непосредственно на принтер. Например, если принтер подключен к параллельному порту компьютера, то для вывода на печать нужно просто ввести строку:

```
cat файл > /dev/par
```

Однако при таком способе возникают некоторые проблемы:

- если в конце команды не ввести знак амперсанда (&), то другую команду будет невозможно ввести до полного окончания печати файла;
- если принтер уже выполняет печать или если невозможно обработать отправленный тип файла, то результат печати может быть искажен, и вы просто зря израсходуете бумагу.

При печати лучше всего использовать функцию спулинга (spooling). При использовании функции спулинга задание на печать помещается в очередь, ожидая момента, когда принтер будет в состоянии его напечатать.

В ЗОСРВ «Нейтрино» существуют два различных механизма спулинга печати:

- с использованием стандартной, аналогичной UNIX утилиты lpr (см. разд. "Печать с помощью утилиты lpr" далее в этом разделе);
- с использованием утилиты spooler (см. подразд. "Печать с помощью утилиты spooler" далее в этом разделе).

Вы можете использовать любой из способов, либо тот и другой в зависимости от начальных настроек вашей машины и сети. Далее перечислены некоторые особенности подключения принтера.

- при подключении USB-принтера к компьютеру, необходимо запустить USB-стек и devu-prn (см. подразд. «USB-устройства» в разделе «Подключение оборудования»), после этого необходимо использовать команды семейства lpr или spooler:

- если принтер подключен к последовательному порту машины, то нужно использовать команды семейства lpr;
- если принтер подключен к параллельному порту машины, то можно использовать как команды семейства lpr, так и утилиту spooler.

В этом случае при работе программ распознавания устройств в момент загрузки системы происходит автоматический запуск программы spooler (см. подразд. "Распознавание устройств" раздела 8). Конфигурационные файлы для большинства часто используемых принтеров находятся в каталоге /etc/printers.

Если вы хотите использовать семейство утилит lpr, то вам нужно настроить конфигурационный файл принтера /etc/printcap.

Если используется сетевой принтер или принтер, который присоединен к параллельному порту другого узла, то для работы с семейством утилит lpr необходимо использовать сеть TCP/IP. Для печати на удаленных принтерах утилита spooler может работать посредством Qnet, SAMBA, NCFTP или даже семейства утилит lpr.

Для выполнения удаленной печати необходимо изменить параметры настроек в некоторых конфигурационных файлах в зависимости от того, используются для печати семейство утилит lpr или утилита spooler.

Если печать производится из графической оболочки Photon (например, с помощью программ Helpviewer или Voyager), то необходимо использовать spooler.

Еще одно отличие состоит в том, что сервис lpd управляет всеми определенными в конфигурационном файле принтерами, в то время как программа spooler управляет только одним принтером, однако несколько экземпляров этой программы может работать одновременно.

14.2. Печать с помощью утилиты lpr

Система печати lpr поддерживает:

- работу нескольких принтеров;
- несколько очередей спулинга;

- печать как на локальном, так и на удаленном принтере;
- возможность подключения принтеров через последовательный канал, для которого требуется инициализация (например, установка скорости передачи в бодах).

Для работы через систему печати lpr необходимо иметь:

- интерфейс пользователя и способ организации и подготовки заданий на печать;
- каталоги спулинга, т. е. место для хранения файлов, ожидающих печати;
- способ предотвращения неавторизованного доступа;
- программу сетевого администратора (для удаленной печати), с помощью которой происходит передача печатаемых файлов;
- некоторые данные об используемом принтере.

Примечание. Для настройки системы lpr нужно войти в систему под учетной записью root.

Интерфейс пользователя

В систему стандартной печати входят следующие основные файлы и команды (рис. 14.1):

- lpd — сервис принтера, с помощью которого и выполняется вся реальная работа;
- lpr — программа постановки в очередь задания на печать;
- lprq — программа для просмотра очереди спулера;
- lprrm — программа для удаления заданий из очереди;
- lprc — программа для управления принтерами и очередями спулера; эта утилита доступна только пользователю с правами root;
- /etc/printcap — главная база данных, где находятся описания принтеров, непосредственно подключенных к машине и доступных через сеть. В базе содержатся данные о доступных принтерах и способе взаимодействия с ними, а также значения важных параметров (например, местонахождение каталога спулинга печати).

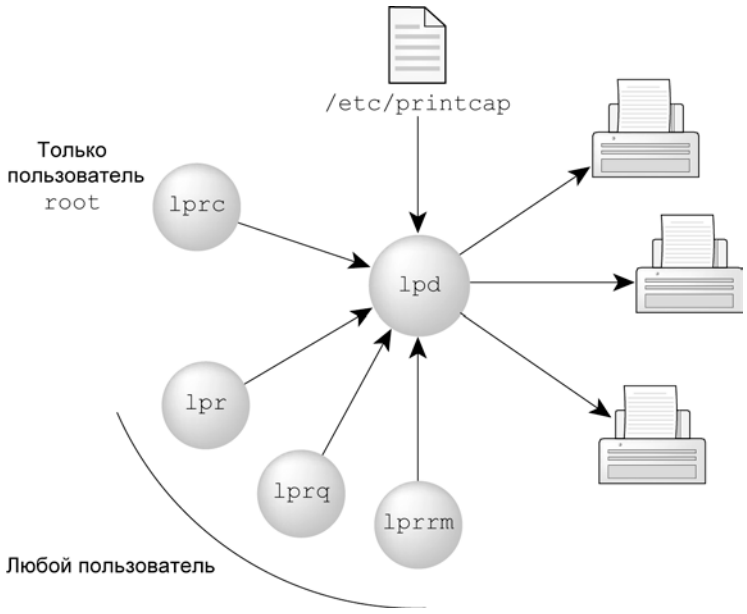


Рис. 14.1. Печать с помощью утилит семейства lpr

lpd — сервис принтера

Программа lpd, которая обычно запускается на этапе загрузки системы из файла /etc/rc.d/rc.local (см. раздел 8), выполняет роль главного сервера координации и управления очередью спулера, параметры которой заданы в файле /etc/printcap. При запуске программы lpd происходит однократный просмотр базы данных /etc/printcap и перезапуск всех принтеров, для которых назначены задания на печать. При нормальной работе программа lpd принимает запросы на обслуживание сокета в домене Интернета (со спецификацией обслуживания "printer") при запросах на доступ к принтеру.

Для обработки запроса сервис принтера порождает копию собственного процесса, при этом в главном процессе продолжается анализ появления новых запросов. Для синхронизации сервисов используются простые текстовые файлы в качестве блокировки. Родительский процесс сервиса использует файл /usr/spool/output/lpd.lock, а дочерний процесс — файл .lock в каталоге спулинга принтера в соответствии с записью в файле printcap.

Взаимодействие клиентов с программой lpd происходит посредством обычного транзактного протокола. Аутентификация удаленных клиентов

осуществляется на основе схемы "привилегированного порта" ("privileged port"), используемой rshd (см. подразд. "Управление доступом" далее в этом разделе).

lpr — запустить задание на печать

С помощью команды lpr задание на печать ставится в локальную очередь, после чего локальному сервису lpd отправляется уведомление о том, что новое задание ожидает запуск в спулинге. Сервис либо отправляет задание для печати на локальный принтер, либо (в случае удаленной печати) делает попытку перенаправления задания на соответствующую машину. Если невозможно открыть принтер или отсутствует доступ к удаленной машине, задание остается в очереди до тех пор, пока не будет возможности его выполнить.

lprq — показать очередь принтера

Программа lprq работает рекурсивно от конца к началу, отображая очередь заданий для машины с заданным принтером, а затем очереди машин, которые направляют задания на данный принтер.

Имеется два формата отображения выходной информации:

- короткий формат (принят по умолчанию — на каждое задание из очереди отводится одна строка вывода;
- длинный формат (определяется с помощью ключа –l) — выводится список и размеры файлов, составляющих задание.

lprrm — удалить задания из очереди

Команда lprrm удаляет задания из очереди спулера. При необходимости вначале по команде lprrm происходит уничтожение процесса сервиса, обслуживающего очередь, а затем перезапуск этого процесса после удаления указанных файлов. Если удаляется задание, назначенное удаленному принтеру, то команда lprrm работает так же, как и команда lprq с одним исключением: сначала проверяются удаляемые задания на локальной машине, затем делается попытка удаления файлов из очереди, находящихся на удаленных машинах.

Примечание. Из очереди можно удалить только собственные задания на печать.

lprc — программа управления принтером

Программа lprc используется для управления системой печати lpr. Для каждого принтера, заданного в файле /etc/printcap, команда lprc может:

- разрешить или запретить работу какого-нибудь принтера;
- разрешить или запретить работу очереди спулера принтера;
- изменить порядок заданий в очереди спулера;
- получить информацию о статусе принтеров, очередях спулеров и состоянии сервисов принтеров.

Программа lprc предоставляет пользователю root возможность локального управления принтерами. Далее приведен список основных команд программы и их назначение. Подробное описание формата каждой команды и полный список команд можно найти в руководстве "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01.

- start — разрешает печать и передает программе lpd запрос на запуск заданий на печать.
- abort — немедленно завершает работу активного сервиса спулера на локальной машине, а затем запрещает печать (тем самым предотвращая запуск командой lpr новых сервисов). Обычно эта команда используется для принудительного перезапуска зависшего сервиса принтера (например, в случае, когда команда lprq сообщает, что сервис запущен, но ничего не происходит).

Команда abort не удаляет из очереди спулера каких-либо заданий на печать, для этого существует команда lprrm.

- enable и disable — команды включают или выключают работу локальной очереди спулера для того, чтобы через команду lpr нельзя было поместить в очередь спулера новые задания. Например, вы можете использовать команду disable при тестировании новых фильтров принтера, поскольку при этом разрешается печать пользователю root, но запрещается для всех остальных

пользователей. Другая распространенная ситуация применения этой команды — запретить пользователям помещать задания в очередь, если ожидается недоступность принтера в течение длительного времени.

- restart — обычным пользователям разрешается перезапустить сервисы печати, когда от программы lprq поступает сообщение об отсутствии запущенных сервисов.

- stop — команда останавливает сервис печати после завершения текущего задания. Эта команда также запрещает печать. Использование этой команды является самым правильным способом отключения принтера для проведения техобслуживания. Имейте в виду, что при остановке принтера этой командой пользователи по-прежнему имеют возможность посылать задания в очередь спулера.

- topq — команда перемещает выбранные задания в начало очереди печати. Эта команда может использоваться для продвижения на печать приоритетных заданий (программа lpr помещает задания в очередь в порядке их поступления).

Каталоги спулинга

Каждый узел, с которого может производиться печать, должен иметь каталог спулинга, в котором сохраняются файлы для печати. По умолчанию для этого каталога назначается путь /usr/spool/output/lpd (этот путь может быть изменен в файле /etc/printcap). Если этот каталог не существует, его нужно создать для всех узлов.

> **Примечание**. Сервис lpd не работает без каталога спулинга и не выдает никаких сообщений об этом. Поэтому весьма полезно при отладке проблем с принтером запускать работу системного журнала (см. описание утилиты syslogd в руководстве "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01). Тогда вы сможете прочесть сообщения об ошибках в файле /var/log/syslog.

Управление доступом

В системе управления принтером организуются защищенные области спулинга, чтобы пользователь не мог изменить относящиеся к принтеру учетные данные или удалить файлы для печати, кроме собственных. Правила, которые соблюдаются при этом, перечислены далее:

- только сервис управления печатью может помещать задания в спулер печати. Запись в область спулинга могут производить только программы, запущенные пользователем сервиса или группой сервиса;
- утилита lpr запускается с идентификатором пользователя (root) и идентификатором группы (daemon). При запуске от имени root утилита lpr может читать любой файл. Идентификатор группы используется для установки соответствующих прав доступа на файлы в зоне спулинга для программы lprrm;
- пользователи не могут изменять управляющие файлы. Управляющие файлы в области спулинга принадлежат пользователю, от имени которого запущен спулер, или группе daemon. Им соответствует режим 0660. Это дает гарантии того, что пользователи не смогут изменить управляющие файлы и что никакой пользователь не сможет удалить файлы никаким способом, кроме применения утилиты lprrm;
- пользователи могут изменять файлы в каталоге спулинга только с помощью утилит печати. Программы работы со спулером (lpd, lprq и lprrm) для получения доступа к файлам спулера и принтерам устанавливают бит setuid для пользователя root и бит setgid для группы daemon;
- управление локальным доступом к очередям осуществляется через запись rg в файле /etc/printcap:

```
:rg=lprgroup:
```

Для передачи задания на печать конкретному принтеру пользователи должны относиться к группе lprgroup. По умолчанию доступ разрешен всем пользователям. Имейте в виду, что после того как файлы попали в локальную очередь, они могут быть напечатаны на локальном принтере или перенаправлены на другой хост (в зависимости от конфигурации);

- в администраторе печати (print manager) происходит аутентификация всех удаленных клиентов. Для аутентификации используется тот же метод, что и в утилите rshd (см. руководство "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01).

Хост, на котором находится клиент, должен быть указан в файле /etc/hosts.equiv или /etc/hosts.lpd, а сообщение о запросе должно исходить из зарезервированного порта.

Примечание. В других утилитах (например, rlogin) для определения принадлежности к хосту тоже используется информация из файла /etc/hosts.equiv. Файл /etc/hosts.lpd служит только для того, чтобы установить, какие хосты имеют доступ к принтерам.

Для разрешения доступа только тем удаленным пользователям, которые имеют учетные записи на локальном хосте, используйте поле rs в записи принтера в файле /etc/printcap.

```
:rs:
```

Сетевой администратор

Если вы хотите печатать на удаленном принтере, то в ЗОСРВ «Нейтрино» необходимо запустить сетевой администратор io-pkt*. Эта программа загружает разделяемые объекты (библиотеки DLL), которые обеспечивают поддержку требуемых протоколов и драйверов устройств.

Например, чтобы загрузить полный стек протоколов TCP/IP npm-tcpip.so и драйвер адаптеров Ethernet, совместимых с NE-2000 (devn-ne2000.so), нужно ввести команду io-pkt* в таком формате:

```
io-pkt-v4 -dne2000
```

Примечание. Чтобы использовать стек TCP/IP таким образом, возможно, потребуется задать конфигурацию своего сетевого интерфейса с указанием типа и номера сетевой интерфейсной платы (NIC), IP-адреса и сетевой маски интерфейса TCP/IP. Более подробные сведения об этом приведены в разделе 13.

Параметры принтера: файл /etc/printcap

До начала печати на узлах должна присутствовать информация о конкретном используемом принтере (как минимум, должно быть известно, где находится принтер). Описание принтера на каждом узле хранится в файле /etc/printcap. В базе данных /etc/printcap для каждого принтера имеется одна или несколько записей.

Примечание. Этот файл отсутствует после первоначальной установки ЗОСРВ «Нейтрино», поэтому при необходимости его нужно создать вручную.

В данном разделе приводится описание основных полей файла. Сведения об остальных полях приведены в руководстве "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01, в разделе, посвященном файлу /etc/printcap.

Типовые начальные установки

Вот как выглядит типичный файл /etc/printcap, который вы можете модифицировать:

```
lpt1|tpptr|printer in Docs department:\
  :lp=/dev/par1:\
  :sd=/usr/spool/output/lpt1:\
  :lf=/usr/adm/lpd-errs:\
  :mx#0:\
  :sh:
```

Каждая запись в файле /etc/printcap описывает принтер. Комментарии начинаются с символа (#). Запись состоит из некоторого числа полей, разделенных символами двоеточия (:). В приведенном примере каждое поле размещено на отдельной строке, но все поля можно располагать и на одной строке при условии, что они начинаются и заканчиваются двоеточием.

Далее приводятся пояснения, касающиеся каждой строки вышеприведенной записи.

- lpt1|tpptr|printer in Docs department:\

Перечисляются известные имена принтера. Имена разделяются символом вертикальной черты (|). Самое последнее имя — это длинное имя, полностью идентифицирующее принтер; только в нем допускается использование пробелов.

Записи могут размещаться на нескольких строках. Для указания продолжения на следующей строке используется символ обратной косой черты (\), при этом он должен быть последним символом на переносимой строке. Для удобства чтения можно включать пустые поля.

- :lp=/dev/par1:\

Имя устройства, которое открывается для вывода (имя по умолчанию /dev/lp).

- :sd=/usr/spool/output/lpt1:\

Каталог спулинга (путь по умолчанию /usr/spool/output/lpd). Для каждого принтера должен быть определен отдельный каталог спулинга. При его отсутствии задание на печать будет передаваться на другие принтеры в зависимости от того, какой из них будет запущен сервисом раньше. Каталог спулинга принято называть таким же именем, что и соответствующий ему принтер.

Примечание. Перед печатью убедитесь в том, что каталог спулинга существует.

- :lf=/usr/adtn/lpd-errs:\

Файл для сохранения выводимых сообщений об ошибках печати (по умолчанию сообщения об ошибках выводятся на консоль).

Примечание. Иногда сообщения об ошибках, которые посылаются на стандартное устройство для вывода ошибок, не сохраняются в файле отчетов. Настоятельно рекомендуется использовать программу-сервис для формирования системных журнальных записей syslogd.

- :mx#0:\

Удалить принимаемые по умолчанию ограничения на размер буфера для спулинга.

- :sh:

Подавить печать пакетного заголовка (burst header) — страницы, на которой приводится список идентификаторов пользователей и информация о задании на печать.

Принтеры на последовательных портах

При подключении принтера через последовательный порт необходимо установить соответствующую скорость и режимы терминала. Далее приводится пример, где заданы параметры принтера DecWriter III, подключенного локально к последовательному порту с пропускной способностью 1200 бод.

```
lp|LA-180 DecWriter III:\
  :lp=/dev/lp:br#1200:fs#06320:\
  :tr=\f:of=/usr/lib/lpf:lf=/usr/adm/lpd-errs:
```

Здесь:

- lp — имя файла, который нужно открыть для вывода;
- br — скорость передачи данных по линии терминала tty;
- fs — флаги, которые устанавливают параметры: CRMOD, без проверки четности, XTABS;
- tr=\f — печатать символ подачи страницы (formfeed) при пустой очереди. Это удобно, когда в принтер вставлена рулонная бумага. Тогда по завершении задания на печать можно просто оторвать бумагу вместо того, чтобы отключать принтер от линии и вручную подавать бумагу;
- of=/usr/lib/lpf — использовать для печати файлов программу фильтрации с именем lpf (см. разд. "Фильтры" далее в этом разделе);
- lf=/usr/adm/lpd-errs — выводить все сообщения об ошибках в файл /usr/adm/lpd-errs, а не на консоль.

Удаленные принтеры

Для принтеров, которые подключены к удаленным хостам, запись lp следует оставить пустой. Например, приводимая далее запись в файле /etc/printcap направляет вывод на принтер lp, подключенный к машине ucbvax.

```
lp|default line printer:\
```

```
:lp=:rm=ucbvax:rp=lp:sd=/usr/spool/vaxlpd:
```

Поле rm задает имя удаленной машины, соединение с которой нужно установить. Это имя должно быть известным именем хоста для компьютера в сети. Поле rp указывает, что имя удаленного принтера — lp (в данном случае это можно не указывать, поскольку такое значение устанавливается по умолчанию). Поле sd определяет, что в качестве каталога спулинга задается /usr/spool/vaxlpd, а не задаваемый по умолчанию каталог /usr/spool/output/lpd.

Фильтры

Фильтры используются для управления взаимозависимостями устройств и функциями учета.

- выходные фильтры.

Применяются, когда не нужен учет или когда через фильтр должны пропускаться все текстовые данные.

Выходной фильтр не годится для ведения учета, потому что его запуск происходит однократно, через него фильтруются все текстовые файлы, он не пропускает имена зарегистрированных пользователей и не идентифицирует начало или конец задания.

- входные фильтры.

Запускаются для каждого печатаемого файла и выдают учетную информацию, если присутствует поле af в записи для принтера файла printcap. Если такое поле присутствует как во входном, так и выходном фильтрах, то выходной фильтр используется только для печати заглавной страницы (banner page). После этого выходной фильтр останавливается, давая возможность работать входным фильтрам.

- другие фильтры.

Служат для преобразования файлов из одного формата в другой. Например:

```
va|varian|Benson-Varian:\
  :lp=/dev/va0:sd=/usr/spool/vad:of=/usr/lib/vpf:\
  :tf=/usr/lib/rvcat:mx#2000:pl#58:px=2112:py=1700:tr=\f:
```

Поле tf определяет, что файл /usr/lib/rvcat используется в качестве фильтра при печати с выходным форматом troff. Этот фильтр необходим для установки устройства в режим печати текста при работе с текстовыми файлами и в режим плоттера при печати файлов типа troff или растровых изображений. Обратите внимание на то, что запись pl определяет размер страницы 58 строк при размере бумаги 8,5 □ 11 дюймов (□ 27 см).

Для разрешения ведения учета нужно добавить фильтр af в записи varian, например, таким образом:

```
va|varian|Benson-Varian:\
  :lp=/dev/va0:sd=/usr/spool/vad:of=/usr/lib/vpf:\
  :if=/usr/lib/vpf:tf=/usr/lib/rvcat:af=/usr/adm/vaacct:\
  :mx#2000:pl#58:px=2112:py=1700:tr=\f:
```

Примечание. В состав ЗОСРВ «Нейтрино» не входят фильтры печати. Эти фильтры нужно либо перенести из другой ОС типа UNIX, либо написать самостоятельно. Если вы не хотите это делать, то можно использовать систему спулинга, в которой имеются драйверы печати для конкретных семейств наиболее популярных принтеров. См. раздел, касающийся утилиты spooler, в руководстве "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01, и подразд. "Печать с помощью утилиты spooler" далее в этом разделе.

Фильтры вызываются сервисом lpd. Стандартными входными данными для фильтров являются данные для печати, стандартным выходом — принтер. Стандартные сообщения об ошибках добавляются в файл lf, который является журналом ошибок (в качестве альтернативы можно использовать систему syslogd). При отсутствии ошибок фильтр должен возвращать код 0, при необходимости перепечатки — код 1, а при необходимости снятия задания на печать — код 2.

Когда утилита lprrm посылает сигнал SIGINT процессу lpd, который управляет печатью, сигнал SIGINT посылается всем фильтрам и их потомкам. Этот сигнал может перехватываться фильтрами, для которых необходимо выполнить операции очистки (например, удаление временных файлов).

Аргументы, передаваемые из программы lpd фильтру, зависят от типа фильтра.

- выходные (of) фильтры вызываются со следующими аргументами:

фильтр -wширина -lдлина

Значения ширина и длина берутся из записей pw и pl в базе данных /etc/printcap.

- входные (if) фильтры вызываются со следующими аргументами:

фильтр [-c] -wширина -lдлина -iотступ –nвходное_имя -hхост файл_учета

Необязательный ключ -c используется лишь в случае, когда управляющие символы должны пересылаться принтеру без их интерпретации, т. е. когда для печати в команде lpr используется ключ -l. Параметры -w и -l имеют тот же смысл, что и в выходном (of) фильтре. Параметры -n и -h применяются для задания входного имени пользователя и имени хоста владельца задания. Последний аргумент является именем файла учета из записи в файле /etc/printcap.

- все остальные фильтры вызываются со следующими аргументами:

фильтр -wширина –удлина –nвходное_имя -hхост файл_учета

Ключи -x и -y определяют горизонтальный и вертикальный размеры страницы в пикселах (значения берутся из записей px и py в файле /etc/printcap). Остальные аргументы имеют тот же смысл, что и во входном (if) фильтре.

Примеры содержимого файла /etc/printcap

В этом разделе приводится несколько примеров, показывающих, как задаются описания принтеров. См. также описание файла /etc/printcap в руководстве "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01.

USB принтер

При подключении USB-принтера к компьютеру и запуске USB стека и devu-prn, как было описано в подразделе «Устройства USB» раздела «Подключение оборудования», необходимо, чтобы содержимое файла /etc/printcap было следующим:

```
hpps: \
:lp=/dev/usbpar0
:sd=/usr/spool/output/hpps
```

Данный файл устанавливает имя hpps для USB-принтера, идентифицирует файл для открытия /dev/usbpar0 (или любого другого устройства, созданного devu-prn), идентифицирует каталог спулинга /usr/spool/output/hpps.

Для доступа к принтеру необходимо установить lpr -Phpps или задать hpps переменную окружения **PRINTER**.

Примечание. Необходимо убедиться, что каталог спулинга существует.

Единственный принтер

Пусть имеются два узла: node1 и node2, при этом в узле node1 имеется принтер, присоединенный к /dev/par1 (рис. 14.2).

Тогда файл /etc/printcap на узле node1 должен быть таким:

```
lpt1:\
  :lp=/dev/par1:
```

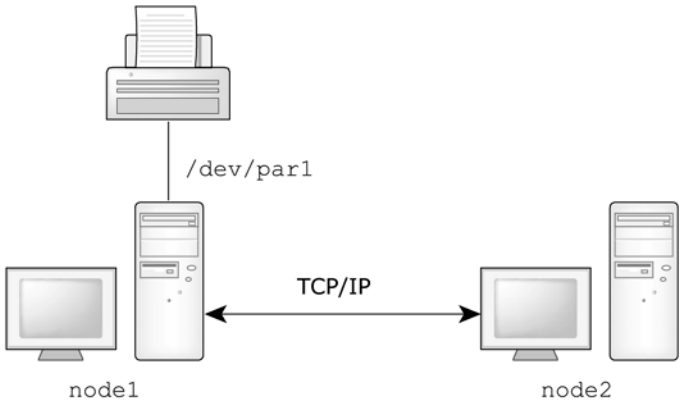


Рис. 14.2. Единственный принтер

В файле просто задается имя lpt1 для принтера, присоединенного к /dev/par1. Нет необходимости описывать что-нибудь еще, поскольку установки по умолчанию выполняют все необходимое. Для получения доступа к принтеру с узла node1 нужно выдать команду lpr -Plpt1 или установить для переменной окружения **PRINTER** значение lpt1.

Примечание. Убедитесь в том, что существует каталог спулинга и что для узла node2 в файле /etc/hosts.lpd имеется запись, указывающая на узел node1.

Файл /etc/printcap на узле node2 должен быть таким:

```
rlpt1:\
  :rm=node1:rp=lpt1:lp=:
```

В этом файле определяется, что удаленному хосту с принтером, имеющим имя lpt1, соответствует узел node1. Имя локального принтера (rlpt1) используется локальными клиентами и может быть аналогичным имени удаленного принтера lpt1.

Следует проверить, что в файле /etc/hosts имеется запись для узла node1.

Несколько принтеров

Теперь добавим к узлу node1 другой принтер, на этот раз он подключается к /dev/par2 (рис. 14.3).

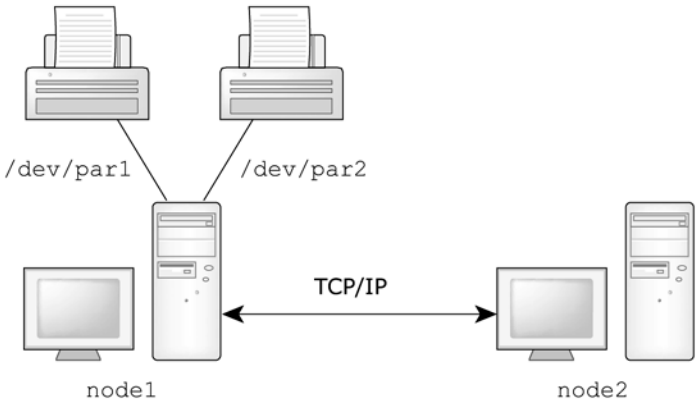


Рис. 14.3. Несколько принтеров

Нужно быть внимательным при определении нескольких принтеров, поскольку предлагаемые по умолчанию параметры не подходят для всех принтеров. Например, используйте поле sd для задания каждому принтеру уникального каталога спулинга.

Файл /etc/printcap на узле node1 имеет следующее содержимое:

```
lpt1:\
  :lp=/dev/par1:sd=/usr/spool/output/lpt1:
```

```
lpt2:\
  :lp=/dev/par2:sd=/usr/spool/output/lpt2:
```

В приведенном описании определяются следующие принтеры:

- lpt1 (присоединен к /dev/par1 и использует для спулинга каталог /usr/spool/output/lpt1);
- lpt2 (присоединен к /dev/par2 и использует для спулинга каталог usr/spool/output/lpt2).

Проверьте, чтобы в файле /etc/hosts.lpd на узле node1 имелась запись для узла node2.

Для того чтобы обращаться в удаленном режиме к этим принтерам с узла node2, нужно на узле node2 в файле /etc/printcap создать такие записи:

```
lpt1:\
  :rm=node1:rp=lpt1:sd=/usr/spool/output/lpt1:lp=:
lpt2:\
  :rm=node1:rp=lpt2:sd=/usr/spool/output/lpt2:lp=:
```

Таким образом, были определены два принтера, которые размещаются на узле node1 с именами, которые можно использовать на узле node2. Проверьте, чтобы в файле /etc/hosts имелась запись для узла node1.

Локальные и удаленные принтеры

Что нужно сделать, если мы захотим переместить один из двух принтеров (например, lpt2) с узла node1 на узел node2 (рис. 14.4)?

На обоих узлах нужно изменить содержимое файла /etc/printcap. Аналогичным образом нужно изменить содержимое файла /etc/printcap на любом другом узле, с которого нужно производить печать. Конкретные изменения для каждого узла таковы:

- на узле node1:

```
lpt1:\
  :lp=/dev/par1:sd=/usr/spool/output/lpt1:
lpt2:\
```

```
  :rm=node2:rp=lpt2:sd=/usr/spool/output/lpt2:
```

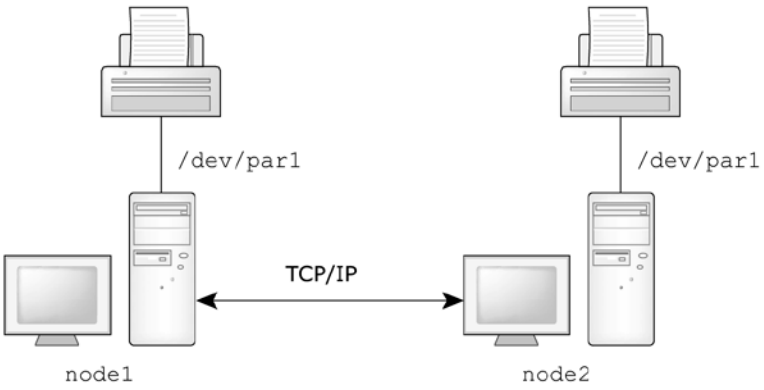


Рис. 14.4. Локальный и удаленный принтеры

- на узле node2:

```
lpt1:\
  :rm=node1:rp=lpt1:sd=/usr/spool/output/lpt1:
lpt2:\
  :lp=/dev/par1:sd=/usr/spool/output/lpt2:
```

- на других узлах:

```
lpt1:\
  :rm=node1:rp=lpt1:sd=/usr/spool/output/lpt1:
lpt2:\
  :rm=node2:rp=lpt2:sd=/usr/spool/output/lpt2:
```

Проверьте, чтобы на каждом узле в файле /etc/hosts имелись бы записи для узлов node1 и node2. Также необходимо проверить наличие записей на узлах node1 и node2 в файле /etc/hosts.lpd для каждого узла, с которого могут использоваться принтеры.

Если вы произведете настройки своей сети для выполнения удаленной печати в соответствии с приведенными примерами, то вы сможете послать файл, находящийся в каталоге /tmp/test на узле node2, на принтер, подключенный к узлу node1, с помощью такой команды:

```
lpr -h -Plpt1 /tmp/test
```

В этом случае происходит следующее.

- вы вводите команду lpr для выполнения печати файла на удаленном принтере;
- утилита lpr отправляет запрос службе печати;
- сервис lpd на узле node2 принимает запрос, создает копию своего процесса для обслуживания запроса на печать и создает подкаталог спулинга для хранения печатаемых файлов;
- порожденный процесс сервиса lpd помещает в спулер задание на печать в виде двух файлов: собственно печатаемый файл данных и заголовочный файл, содержащий информацию о задании на печать (она может быть отпечатана в виде отдельной титульной страницы);
- порожденный процесс сервиса lpd производит обработку заданий на печать, помещенных в спулер, в порядке их поступления и начинает передачу пакетов данных с заданиями на печать удаленному сервису lpd;
- сервис lpd на узле node1 принимает пакеты данных как запросы на печать, проверяет, от разрешенного ли узла поступил запрос, затем порождает копию своего процесса для обслуживания запроса и создает подкаталог спулинга для хранения печатаемых файлов (если запрос поступает от неразрешенного узла, то в адрес источника запроса отсылается сообщение об отказе);
- порожденный процесс сервиса lpd принимает пакеты данных, помещая в очередь спулера сформированное задание на печать, после чего задания на печать пересылаются на указанный принтер в порядке их поступления.

Удаленная печать на принтер, находящийся в другой сети

С помощью протокола TCP/IP и утилиты lpr можно выполнять печать файла на удаленном принтере, присоединенном к серверу другой сети. Нужно просто сделать соответствующую настройку сетевого узла ЗОСРВ «Нейтрино» для удаленной печати, а удаленный сервер настроить на работу по протоколу TCP/IP и на обработку принтеров, совместимых с требованиями утилиты lpr.

Например, пусть необходимо напечатать PostScript-файл /root/junk.ps с некоторого узла сети ЗОСРВ «Нейтрино», но единственный доступный PostScript-принтер (windows_printer) присоединен к серверу Windows с IP-адресом 10.2.1.8.

Вначале нужно убедиться, что Windows-сервер сконфигурирован для печати через протокол TCP/IP и что принтер совместим с утилитой lpr. После этого, войдя под учетной записью root с узла сети ЗОСРВ «Нейтрино», нужно выполнить следующие действия.

- добавьте в файл /etc/printcap описание принтера в следующем виде:

```
rlpt4:\
  :rm=windows_server:lp=:rp=windows_printer:\
  :sd=/usr/spool/output/lpd/rlpt4:
```

- в файле /etc/hosts добавьте новую строку такого вида:

```
10.2.1.8    windows_server
```

- создайте каталог спулинга:

```
mkdir /usr/spool/output/lpd/rlpt4
```

- запустите утилиту lpd.

Для запуска печати PostScript-файла на принтере введите следующую команду:

```
lpr -Prlpt4 junk.ps
```

Удаленная печать на TCP/IP-принтере с использованием утилиты lpr

Принтер, имеющий возможность работать напрямую через протокол TCP/IP, для выполнения печати не требует подключения к какому-либо компьютеру. Все запросы принтер обрабатывает самостоятельно. Поэтому для работы с таким принтером необходимо проделать все те же шаги по настройке, что описаны выше, со следующими небольшими изменениями.

- в файле /etc/hosts узла, с которого будет производиться печать, нужно добавить запись с именем и IP-адресом удаленного принтера, например:

```
10.2.0.4    tcpip_printer
```

- в файле /etc/printcap на том же узле нужно добавить запись с описанием принтера:

```
rlpt2:\
  :rm=tcpip_printer:rp=/ps:sd=/usr/spool/output/lpd/rlpt2:
```

В этом примере видно, что tcpip_printer используется в качестве имени удаленной машины (в данном случае роль машины выполняет непосредственно принтер), а каталог спулинга определен как /usr/spool/output/lpd/rlpt2. Обратите внимание на то, что удаленный принтер определен как /ps. В некоторых сетях это имя назначается принтерам, которые могут работать с файлами формата PostScript. Нужно отыскать то имя, которое необходимо для работы с принтером в требуемом формате. Для работы с разными форматами печатаемых файлов могут требоваться разные имена (например, для печати PostScript-файлов и обычных текстовых файлов).

Еще раз проверьте, что создан каталог спулинга. Следуйте процедуре, описанной в разд. "Локальные и удаленные принтеры" ранее в этом разделе, и тогда вы сможете выполнить печать на удаленном принтере с помощью команды:

```
lpr –Prlpt2 /root/junk.ps
```

По этой команде PostScript-файл с именем /root/junk.ps посылается на принтер с именем tcpip_printer, имеющий IP-адрес 10.2.0.4.

Примечание. В приведенном примере мы выбрали простой вариант отправки PostScript-файла на PostScript-принтер. Это легко реализовать, поскольку форматирование встроено в текст PostScript-файла. Иногда может потребоваться фильтрация содержимого файла, отправленного для печати с помощью утилиты lpr. Для принтера можно определить фильтр посредством добавления соответствующей записи в файл /etc/printcap (более подробно об этом см. в разд. "Фильтры" ранее в этом разделе).

14.3. Печать с помощью утилиты spooler

В качестве альтернативы стандартному механизму печати с использованием UNIX-подобного семейства программ lp* в ЗОСРВ «Нейтрино» служит утилита spooler. Она применяется в приложениях графической оболочки Photon для печати. Для преобразования выходного графического потока (draw-stream) оболочки Photon (phs) в формат, воспринимаемый принтером, используется специальный фильтр.

Начальные установки spooler

Утилита spooler обычно запускается программой распознавания устройств (enumerator) при старте ЗОСРВ «Нейтрино» (см. раздел 8). Утилита управляет работой одного принтера, однако можно запускать более одного экземпляра утилиты spooler.

После запуска утилиты spooler (вручную или на системном уровне) происходит следующее (рис. 14.5):

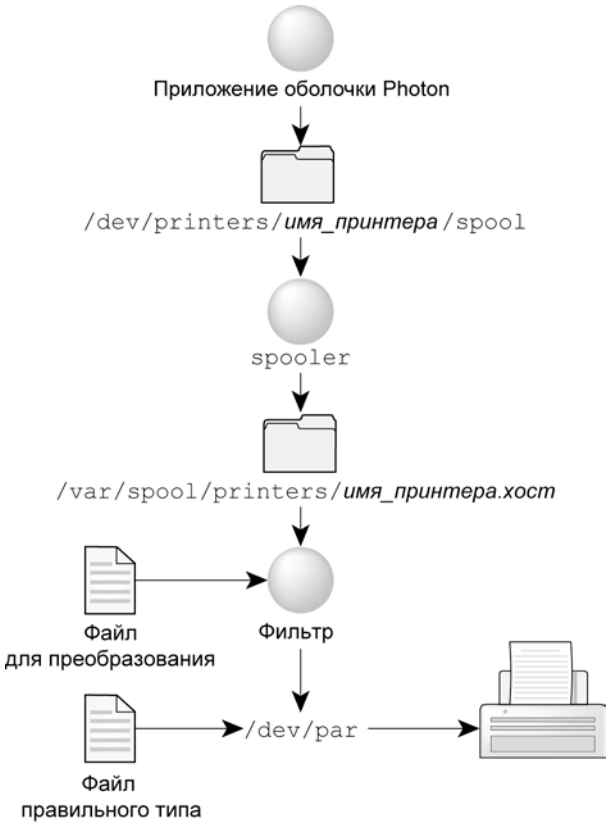


Рис. 14.5. Печать с помощью утилиты spooler

- в каталоге /dev создается необходимая запись для принтера:
  /dev/printers/имя_принтера/spool
- из программы spooler на принтер выдается запрос для определения его типа, создается файл свойств данного принтера на основе системного файла общей конфигурации принтера (см. далее) и сохраняется в каталоге принтера на пути /dev.
- после этого создается каталог спулинга:

/var/spool/printers/имя_принтера.хост

- и наконец, в каталоге спулинга сохраняется файл со свойствами принтера.

В каталоге /etc/printers хранятся файлы общей конфигурации для наиболее часто используемых типов принтеров, в том числе перечисленные в табл. 14.1.

Таблица 14.1

Принтер	**Файл конфигурации**	**Фильтр Photon**
Canon	bjc.cfg	phs-to-bjc
Epson	epson.cfg	phs-to-escp2
Epson IJS	epijc.cfg	phs-to-ijs
Hewlett-Packard	pcl.cfg	phs-to-pcl
PostScript	ps.cfg	phs-to-ps

Существует также специальный фильтр phs-to-bmp, с помощью которого файл графического потока оболочки Photon преобразуется в формат BMP. В файлах конфигурации определяются возможные установки принтера и установки по умолчанию, а также, какой из фильтров подходит для данного принтера.

Когда производится печать из приложения оболочки Photon, то файл для печати отправляется в каталог спулинга /dev/printers/имя_принтера/spool. Приложение оболочки Photon может построить другой файл конфигурации для выбранного принтера в зависимости от введенной дополнительной информации.

Если уже имеется файл, формат которого правильно воспринимается принтером или для которого существует фильтр, то печать можно производить путем копирования файла в каталог спулинга для необработанных данных:

```
cp мой_файл /dev/printers/имя_принтера/raw
```

Когда программа spooler обнаруживает задание на печать в каталоге /dev/printers/имя_принтера/raw, файл задания копируется в каталог спулинга /var/spool/printers/имя_принтера.хост, после чего запускается соответствующий фильтр, с помощью которого происходит подготовка файла и отправка его на принтер.

Обычно файл спулера для печати сохраняется в каталоге на диске, затем эта информация о местонахождении файла передается фильтру. Если нужно сэкономить дисковую память, то при вызове программы spooler можно использовать ключ -F для отключения буферизации файлов для печати в каталоге спулинга. При наличии этого ключа программа spooler посылает фрагменты печатаемого файла непосредственно в буфер FIFO для печати по частям. Фильтр получает эти данные из буфера FIFO, и после обработки происходит печать фрагмента. После освобождения буфера spooler загружает в буфер следующий фрагмент файла и т. д. до окончания печати всего файла.

Если в приложении оболочки Photon вызывается команда предварительного просмотра печати (print preview), то в этом случае файл отправляется для обработки утилитой preview. Если необходимо просмотреть очередь печати или выполнить управление ею, то нужно запустить из командной строки команду prjobs или из меню запуска (**Launch**) выбрать пункт меню **Utilities→Print Manager**. Более подробные сведения приведены в руководстве "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01.

Печать с использованием USB принтера

При подключении USB принтера к компьютеру и запуске USB-стека и devu-prn, как описано в подразделе «USB-устройство» раздела «Подключение оборудования», необходимо запустить дополнительно spooler для управления (например, в /etc/rc.d/rc.local).

Примечание. ЗОСРВ «Нейтрино» не может обнаруживать (детектировать) USB принтеры.

Для установки USB принтера необходимо выполнить следующее:

- создать /usr/spool/output/device, где device – это устройство, созданное devu-prn для принтера (например, usbpar0).
- запустить спулер, заданный принтером. Например:

```
spooler -d /dev/usbpar0
```

Принтер должен появится в /dev/printers.

Удаленная печать по протоколу Qnet

Для печати через сеть Qnet необходимо отправить файл в каталог /net/имя_узла/dev/printers/имя_принтера/spool. Программа spooler для принтера должна быть запущена на узле с именем имя_узла.

Удаленная печать по протоколу TCP/IP

Для настройки печати с использованием программы spooler на удаленном принтере необходимо перенаправить задание на печать программе lpr. Такая возможность обеспечивается тем, что задание на печать пересылается на принтер фильтром. Поэтому необходимо просто указать имя удаленного принтера в командной строке фильтра в файле конфигурации, используемом программой spooler.

Для того чтобы освоить такую возможность, попробуйте сначала организовать работу удаленного принтера через программу lpr (см. разд. "Удаленная печать на TCP/IP-принтере с использованием утилиты lpr" ранее в этом разделе). После этого выполните следующее.

- скопируйте файл конфигурации принтера, который вы хотите использовать (в данном случае это принтер PostScript):

```
cp /etc/printers/ps.cfg /etc/printers/test.cfg
```

- в файле test.cfg найдите командную строку фильтра, которая выглядит приблизительно так:

```
Filter = phs:$d:phs-to-ps
Filter = raw:$d:cat
```

Командные строки фильтра имеют формат:

источник:приемник:фильтр

Командная строка фильтра phs указывает фильтру обработать файлы с расширением .phs посредством фильтра с именем phs-to-ps прежде, чем отправить их по адресу назначения, указанному программой spooler. Команда фильтра raw предназначена для утилит, которые на выходе дают уже корректно форматированный для печати результат.

- измените командную строку фильтра phs с такого вида:

```
Filter = phs:$d:phs-to-ps
```

на такой:

```
Filter = phs:ps:phs-to-ps
```

- добавьте строку такого вида, чтобы все PostScript-файлы после фильтра посылались на удаленный принтер rlpt2:

```
Filter   ps:$d:lpr -Prlpt2
```

Эти действия приведут к изменению задаваемого спулером адреса назначения на значение ps, так что после преобразования .phs-файла фильтром phs-to-ps в формат ps файл будет отправлен на ps-фильтр. Благодаря добавленной строке ps-фильтра, файлы отправляются для обработки в программу lpr, в качестве ключа которой указано выполнение печати на удаленном принтере (точно такая же ситуация была описана в разд. "Удаленная печать на TCP/IP-принтере с использованием утилиты lpr" ранее в этом разделе).

Возможно, вам интересно узнать, что произошло с адресом назначения ($d), передаваемым программой spooler. Этот адрес отброшен, потому что программа lpr (в отличие от фильтра phs-to-ps) не возвращает задание фильтру, а сама завершает все необходимые действия.

- и наконец, запустите новый экземпляр программы spooler, указав в качестве параметра путь к вашему новому файлу конфигурации (в нашем случае etc/printers/test.cfg) и имя необходимого принтера (в нашем случае rlpt2). Строка запуска будет выглядеть так:

```
spooler -d /dev/null -c /etc/printers/test.cfg -n rlpt2 &
```

С помощью ключа -n определяется имя принтера, которое появляется в диалоге печати приложения оболочки Photon.

- если вы хотите, чтобы программа spooler запускалась таким же образом при загрузке машины, то добавьте описанную выше команду в файл /etc/rc.d/rc.local. Более подробно об этом см. раздел 8.

Теперь можно печатать PostScript-файлы на удаленном TCP/IP-принтере либо из приложений оболочки Photon, либо из командной строки.

- удаленная печать из приложений оболочки Photon.

Выберите необходимый принтер (в примере это rlpt2) в диалоговом окне выбора принтера (Select Printer).

- удаленная печать из командной строки.

Скопируйте файл для печати в каталог, используемый спулером, с помощью такой команды:

```
cp /root/my_file.ps  /dev/printers/rlpt2/spool/
```

Примечание. Примеры файлов конфигурации для печати с использованием программ lpr, SAMBA и NCFTP даны в приложении.

14.4. Устранение неполадок

Сообщения об ошибках в программах семейства lpr

Приводимые далее описания сообщений об ошибках, генерируемые утилитами печати семейства lp*, помогут вам в разрешении возникающих в процессе печати проблем.

Сообщения об ошибках утилиты lpr.

- lpr: имя_файла: copyfile is too large

Отправленный на печать файл имеет больший размер, чем заданный максимальный размер файла для принтера. Этот размер определен параметром mx в соответствующей записи файла printcap.

- lpr: принтер: unknown printer

Неизвестный принтер. Принтер не найден в базе данных /etc/printcap, возможно, из-за пропущенной или неправильной записи.

- lpr: принтер: jobs queued, but cannot start daemon

Задание поставлено в очередь, но невозможно запустить сервис. Не удалось подключиться к сервису lpd на локальной машине, возможно, из-за зависания

сервера принтера или из-за отсутствия ответа от сервера. Суперпользователь может перезапустить сервис lpd с помощью команды:

```
/usr/bin/lpd
```

Проверить состояние сервиса главного принтера можно с помощью команды:

```
sin -P lpd
```

Если пользователь утилиты lpr не является root, а идентификатор группы не относится к категории daemon, то проверку можно выполнить с помощью команды:

```
ls -lg /usr/bin/lpr
```

- lpr: принтер: printer queue is disabled

Отключена работа очереди принтера. Данное сообщение означает, что работа очереди была отключена командой lprc disable (см. разд. "lprc — программа управления принтером" ранее в этом разделе), чтобы программа lpr не могла помещать файлы в очередь. Обычно такая операция выполняется перед тем, как принтер отключается на длительное время. Суперпользователь может снова включить принтер с помощью утилиты lprc.

Сообщения об ошибках утилиты lprq.

- waiting for printer to become ready (offline?)

Ожидание готовности принтера. Сервис не может открыть печатающее устройство. Это может быть вызвано несколькими причинами (например, принтер отключен, кончилась или замята бумага). Конкретная причина выясняется по значению кода ошибки, возвращаемого системным драйвером принтера. Некоторые принтеры не могут передать информацию, позволяющую определить точную причину проблемы (отключен принтер или произошел другой отказ). Это особенно свойственно принтерам, подключаемым через последовательный порт.

Другой возможной причиной появления такого сообщения является то, что устройство было открыто в эксклюзивном режиме каким-то другим процессом, например, выходным фильтром. Все что можно сделать в этом случае — аварийно завершить мешающую программу и перезапустить принтер с помощью утилиты lprc.

- printer is ready and printing

Принтер готов и печатает. Программа lprq проверяет, существует ли процесс сервиса для принтера, и выдает текущий статус файла, находящегося в каталоге спулинга. Если сервис не отвечает, то пользователь с учетной записью root может воспользоваться программой lprc для аварийного завершения текущего сервиса и для запуска его нового экземпляра.

- waiting for host to come up

Ожидание появления хоста. Это означает, что сервис пытается соединиться с удаленной машиной host для того, чтобы послать файлы из локальной очереди. Если удаленная машина работает, то, вероятнее всего, что зависла или заблокирована программа lpd на удаленной машине. В этом случае ее нужно перезапустить.

- sending to host

Отправка файлов хосту. Файлы в настоящий момент отправляются на удаленный хост. Если этого не происходит, то пользователь с учетной записью root должен воспользоваться программой lprc для аварийного завершения и перезапуска локального сервиса.

- Warning: printer is down

Предупреждение: принтер отключен. Данный принтер был помечен программой lprc как недоступный.

- Warning: no daemon present

Предупреждение: отсутствует сервис. Процесс lpd (сервис), заданный в файле блокировки данного каталога и следящий за очередью спулера, не существует. Обычно такое происходит при неожиданной блокировке сервиса. Для диагностики проблемы проверьте записи в журнальном файле с сообщениями об ошибках для данного принтера и в системном журнальном файле syslog. Для перезапуска сервиса lpd введите команду:

```
lprc restart printer
```

- no space on remote; waiting for queue to drain

Нет свободного места на удаленной машине; ожидание очистки очереди. Это сообщение означает, что недостаточно места на диске удаленной машины. Если файл имеет достаточно большой размер, то на удаленной машине всегда будет не хватать места (даже после освобождения очереди). Решением проблемы является перемещение очереди спулера или освобождение дискового пространства на удаленной машине.

Сообщение об ошибках утилиты lprrm.

- lprrm: принтер: cannot restart printer daemon

Невозможно перезапустить сервис принтера. Этот случай аналогичен ситуации, когда от программы lpr поступает сообщение о невозможности запуска сервиса.

Сообщения об ошибках утилиты lprc.

- couldn't start printer

Невозможно запустить принтер. Этот случай аналогичен ситуации, когда от программы lpr поступает сообщение о невозможности запуска сервиса.

- cannot examine spool directory

Невозможно проверить каталог спулинга. Ошибочные сообщения, начинающиеся со слова cannot (невозможно), обычно являются следствием неправильных прав доступа или режима защиты в отношении файла блокировки, каталога спулинга или программы lprc.

Сообщения об ошибках утилиты lpd.

При работе утилиты lpd может выдаваться много различных сообщений, которые поступают в системный журнал программы syslogd. Большинство сообщений касаются файлов, которые нельзя открыть. Обычно это означает, что есть некорректности в файле /etc/printcap или в режимах защиты других файлов. Файлы могут быть недоступны, если к ним обращаются иными средствами (а не с помощью программы lpr).

В дополнение к сообщениям, генерируемым программой lpd, любой из фильтров, порождаемых lpd, может посылать сообщения в системный журнальный

файл программы syslogd или в журнальный файл для сообщений об ошибках (файл, определенный в записи lf файла /etc/printcap). Если вы хотите диагностировать проблему, запустите программу syslogd.

Устранение неисправностей при удаленной печати

Если не работает печать отправленного файла, то вы можете получить сообщение об ошибке от одной из утилит семейства lp* (см. пункт "Сообщения об ошибках утилиты lpr"). Если же не появляется никаких сообщений об ошибках, то можно попробовать применить приводимые ниже действия.

- несмотря на то, что порождаемая программа lpd создает подкаталоги спулинга (что необходимо для сохранения задания для печати), главный каталог спулинга должен создаваться вами заранее. Проверьте еще раз, существует ли этот каталог (по умолчанию его путь /usr/spool/output/lpd);
- проверьте на каждом узле содержимое файла /etc/printcap;
- убедитесь в том, что в файле /etc/hosts.lpd на узле, где происходит печать, есть запись об узле, с которого отправляются файлы на печать;
- если lpd еще не запущен, и невозможно выполнить его запуск, необходимо проверить существует ли файл блокировки /usr/spool/output/lpd.lock. Если данный файл существует и не запущен lpd (например, после сбоя питания или падения системы), файл необходимо удалить;
- убедитесь в том, что работает программа сетевого администратора io-pkt* с соответствующими разделяемыми объектами (библиотеками);
- запустите программу syslogd и просмотрите системные сообщения в журнальном файле syslog.

15. Подключение оборудования

15.1. Общие сведения

При начальной загрузке настольной ЗОСРВ «Нейтрино» сперва запускается программа распознавания устройств (device enumerator) — управляющая программа, которая обнаруживает большинство из подключенных аппаратных устройств. Служба распознавания устройств загружает набор конфигурационных файлов из каталога /etc/system/enum/devices, в которых определено, что должна делать система (например, запускать конкретный драйвер), когда добавляется или удаляется оборудование.

Конфигурационные файлы службы распознавания устройств при необходимости можно редактировать. Более подробные сведения приведены в разделе 8 и в подразделе о enum-devices в руководстве "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01.

В системе на основе встраиваемой ЗОСРВ «Нейтрино» обычно входит вполне определенный набор оборудования, поэтому при загрузке системы, скорее всего, можно беспрепятственно запустить соответствующие драйверы.

Примечание. В настоящее время ЗОСРВ «Нейтрино» не поддерживает работу со средствами хранения на магнитных лентах.

Информация данного раздела потребуется вам в том случае, если служба распознавания устройств не сможет обнаружить какое-то системное устройство или вы хотите вручную выполнить конфигурирование статических устройств встраиваемой системы.

Примечание.

- для запуска любого драйвера нужно войти в систему под учетной записью root.
- до запуска ЗОСРВ «Нейтрино» проверьте, чтобы в BIOS была отключена функция PnP-aware OS.

15.2. Устройства PCI/AGP

Если вы не знаете тип используемого контроллера, то для его идентификации можно воспользоваться утилитой pci:

```
pci –vvv | less
```

В результате работы этой команды вы получите, например, следующий результат.

```
Class          = Mass Storage (IDE)
Vendor ID      = 8086h, Intel Corporation
Device ID      = 7111h, 82371AB/EB PIIX4 IDE Controller
PCI index      = 0h
Class Codes    = 010180h
Revision ID    = 1h
Bus number     = 0
Device number  = 4
Function num   = 1
Status Reg     = 280h
Command Reg    = 5h
        I/O space access enabled
        Memory space access disabled
        Bus Master enabled
        Special Cycle operations ignored
        Memory Write and Invalidate disabled
        Palette Snooping disabled
        Parity Checking disabled
        Data/Address stepping disabled
        SERR# driver disabled
        Fast back-to-back transactions to different agents disabled
Header type    = 0h Single-function
BIST           = 0h Build-in-self-test not supported
Latency Timer  = 20h
Cache Line Size= 0h
PCI IO Address = d800h length 16 enabled
Max Lat        = 0ns
Min Gnt        = 0ns
PCI Int Pin    = NC
Interrupt line = 0
```

```
Device Dependent Registers:
0x40: 07 c0 03 80 00 00 00 00 05 00 02 02 00 00 00 00
0x50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0xA0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0xB0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0xC0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0xD0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0xE0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0xF0: 00 00 00 00 00 00 00 00 30 0f 00 00 00 00 00 00
```

Найдите запись для необходимого устройства, и вы сможете выяснить конкретные данные относительно идентификаторов производителя/продавца и устройства. Для идентификации устройства можно воспользоваться поиском по ключевым словам (например, Audio).

Необходимую информацию можно найти на сайте производителя или воспользоваться идентификаторами Vendor ID и Device ID для установления перекрестных ссылок в файле /usr/include/hw/pci_devices.h. Можно также поискать нужную информацию на сайте **http://www.pcidatabase.com/**.

15.3. Устройства CD-ROM и DVD

Обычно CD и DVD-устройства подключаются к шине SCSI или EIDE (ATA). Выбор драйвера зависит от типа шины. Проверьте, чтобы оборудование было правильно установлено и чтобы оно правильно распознавалось в BIOS. Если CD-дисковод подключен к шине EIDE, то просто используйте драйвер devb-eide. Если дисковод работает с шиной SCSI, то нужно указать необходимый драйвер для интерфейса SCSI (см. подразд. "Жесткие диски" далее в этом разделе).

Разделяемый объект cam-cdrom.so загружается драйвером по умолчанию, он обеспечивает общий метод доступа для устройств CD-ROM. В зависимости от того, какой драйвер запущен, загружается один из следующих объектов:

- fs-cd.so — поддержка CD-ROM (файловая система ISO-9660);

- fs-udf.so — поддержка CD-ROM (файловая система ISO-9660) и DVD-ROM (файловая система Universal Disk Format).

Примечание. fs-cd.so исключает fs-udf.so.

Устройства CD-ROM и DVD-ROM указываются в каталоге /dev, как /dev/cdx, где x – количество приводов, начиная с 0.

Простое монтирование приводов с указанием типа файловой cd или udf. Например:

```
mount -t cd /dev/cd0 /fs/cdrom
mount -t udf /dev/cd0 /fs/dvdrom
```

При замене дисков нет необходимости в перемонтировании привода. Для более подробной информации о специальных опциях см. cam-cdrom.so, fs-cd.so и fs-udf.so в справке по утилитам.

Приводы DVD RAM могут быть интерпретированы, как жесткие диск. Они отображаются в каталоге /dev, как CD, но при этом их можно монтировать и использовать как жесткие диски – см. подраздел «Жесткие диски» далее.

15.4. Дисководы для гибких дисков

Драйвером дисковода для гибких дисков является devb-fdc. Для использования гибких дисков необходимо разрешить в BIOS работу контроллера гибких дисков, а также задать в BIOS конфигурацию правильного типа дисковода (например, **1.44MB/2.88MB**). По умолчанию драйвер использует следующие настройки:

- порт ввода/вывода 0x3f0;
- прерывание IRQ 6;
- канал прямого доступа в память DMA 2.

Если для контроллера используются другие адреса, то нужно внести соответствующие изменения в начальные установки драйвера.

Примечание. Размер кэша по умолчанию задается параметром модуля io-blk.so в размере 15% от объема системной памяти, что даже избыточно для

драйвера devb-fdc. Возможно, вы захотите уменьшить это значение до разумных пределов таким образом:

```
devb-fdc blk cache=128K &
```

По умолчанию распознавание устройств драйвером devb-fdc не запускается (см. подразд. «Распознавание устройств» раздела «Загрузка образа ЗОСРВ «Нейтрино»»). Если в системе есть привод для гибких дисков, драйвер можно запустить вручную или раскомментировать строку devb-fdc в файле /etc/system/enum/devices/block.

Драйвером создается запись /dev/fdx, где x — порядковый номер дисковода гибких дисков, начиная с 0. Если такая запись не появляется, то, возможно, сделаны неправильные установки в BIOS или есть какие-то проблемы с контроллером. Для прояснения вопроса проверьте информацию, выдаваемую утилитой sloginfo.

После появления записи в каталоге /dev нужно смонтировать дисковод для гибкого диска. Команда mount определяет тип используемой файловой системы (DOS, QNX 4), но его также можно явно указать в командной строке.

- для работы с дискетами формата DOS нужно ввести команду:

```
mount -tdos /dev/fd0 /fs/dos_floppy
```

Используйте команду mkdosfs для форматирования гибких и жестких дисков DOS. Данная утилита поддерживает FAT 12/16/32.

- для работы с дискетами формата QNX 4 нужно ввести команду:

```
mount -tqnx4 /dev/fd0 /fs/qnx_floppy
```

При замене дисководов гибких дисков выполнять команду перемонтирования устройства не требуется.

Примечание. Не вытаскивайте дискету в момент чтения или записи данных. Приводы гибких дисков работают медленнее жестких дисков, поэтому приходится ждать некоторое время окончания операции. Перед вытаскиванием дискеты убедитесь, что индикатор работы дисковода погас.

15.5. Жесткие диски

По умолчанию, ЗОСРВ «Нейтрино» детектирует контроллеры дисков, установленные в системе, и запускает соответствующие драйверы.

В такой системе драйверы блочных устройств ввода/вывода запускаются утилитой diskboot, включенной в образ ОС. Если вы хотите изменить порядок запуска драйверов, нужно изменить образ начального запуска и ключи запуска программы diskboot. Например:

```
diskboot -o devb-eide,blk cache=30m
```

Более подробные сведения приведены в разделе 8 и в руководстве "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01, в подразделе о diskboot.

Устройства EIDE

Для интерфейсов EIDE используется драйвер devb-eide, который по умолчанию автоматически обнаруживает интерфейс и подключенные к нему устройства. В драйвер, кроме поддержки стандартных режимов программируемого ввода/вывода (Programmed Input/Output, PIO), включена поддержка режимов UDMA (Ultra Direct Memory Access, ускоренный прямой доступ в память). В список поддерживаемого оборудования входят адаптеры и соответствующие функции для них (см. разд. "Общие сведения" ранее в этом разделе).

Драйвер devb-eide можно запускать без ключей и по умолчанию системный контроллер EIDE будет обнаружен автоматически:

```
devb-eide &
```

После запуска драйвера автоматически обнаруживаются все устройства, подключенные к контроллеру EIDE. Для каждого устройства создается запись в каталоге /dev (например, жесткий диск появляется как hdx, где x — порядковый номер устройства, начиная с 0).

Например, пусть в системе имеются два установленных жестких диска. В каталоге /dev драйвер создает следующие записи:

- /dev/hd0 — обычно это первичный ведущий диск (primary master);
- /dev/hd1 — обычно это первичный ведомый диск (primary slave) или следующий по порядку (вторичный ведущий) диск (secondary master).

Если в системе установлены один жесткий диск и CD-ROM, то записи будут выглядеть так:

- /dev/hd0 — первичный ведущий диск;
- /dev/cd0 — привод CD-ROM.

Примечание. При наличии ведомого диска в системе обязательно должен присутствовать ведущий диск.

При запуске драйвера на консоль выводится тип обнаруженного оборудования, а также другая отладочная информация, которая пересылается в системный журнал slogger. Для просмотра журнала служит утилита sloginfo.

Примечание. В информации, выводимой утилитой sloginfo, может встретиться некоторое число записей вида ASC_MEDIA_NOT_PRESENT. Драйвер выводит такие записи
в журнал в тех случаях, когда в приводе CD-ROM отсутствует компакт-диск. Такие записи можно игнорировать.

Устранение неисправностей драйвера devb-eide

Если драйвер не обнаруживает интерфейс или подключенный к нему дисковод, попробуйте применить приведенные далее действия.

- узнайте в службе технической поддержки, поддерживается ли данный интерфейс. Даже при отсутствии интерфейса в списке поддерживаемых контроллер EIDE может работать в стандартном режиме с программируемым вводом/выводом (PIO), который работает более медленно, но практически во всех случаях;
- проверьте правильность установок интерфейса в BIOS, а также правильность идентификации дисков в BIOS;
- проверьте правильность назначения дисков. Согласно стандарту ATAPI, для каждого ведомого диска обязательно должен присутствовать ведущий диск. Не может быть двух ведущих или двух ведомых дисков;

- убедитесь в правильной работе разъема электропитания;
- в конфигурационном файле драйвера должен быть указан идентификатор устройства и идентификатор продавца;
- в конфигурационном файле драйвера devb-eide должны быть указаны адрес порта ввода/вывода и номер прерывания IRQ.

Вы можете столкнуться и с другими проблемами. Ниже даны рекомендации, что при этом нужно делать.

- при зависании драйвера отключите режим управления шиной busmastering (например, командой devb-eide eide nobmstr);
- если по команде sloginfo вы видите записи: eide_transfer_downgrade: UDMA CRC error (downgrading to MDMA), уменьшите номер режима передачи и проверьте кабели;
- если по команде sloginfo вы видите записи: eide_timer: timeout path XX, device XX, проверьте, что драйвером используется правильное прерывание, уменьшите номер режима передачи и проверьте кабели;
- если при конфигурировании PCMCIA-диска использовались смежные адреса отображения ввода/вывода, т. е. 0x320 (а не 0x1f0, 0x170), задайте адрес управляющего блока (control block address) интерфейса. Адрес управляющего блока имеет смещение 12 относительно базы. Если интерфейс PCMCIA размещен на порте ввода/вывода с адресом 0x320 и использует прерывание IRQ 7, то задайте такую строку:
- devb-eide eide ioport=0x320:0x32c,irq=7,noslave
- если устройства поддерживают режим UDMA 4 или выше, но по команде sloginfo выдается, что драйвер использует режим с меньшим номером, убедитесь, что вы используете 80-контактный кабель;
- если у вас подключен 80-контактный кабель, устройства поддерживают режим UDMA 4 или выше, но по команде sloginfo выдается, что драйвер использует режим с меньшим номером, то, возможно, дело в устаревшей версии микропрограммного обеспечения (firmware) устройства.

При определении типа кабеля драйвер использует данные, предоставляемые микропрограммным обеспечением устройства. Необходимо убедиться в том, что производитель произвел обновление микропрограммного обеспечения. Для переопределения режима можно также использовать ключ udma=xxx в командной строке. Например:

```
devb-eide eide vid=0x8086,did=0x2411,pci=0,chnl=1,master=udma=4
```

Если дисководы успешно обнаружены, но работают медленно, попробуйте следующие действия.

- используйте команду sloginfo, чтобы проверить информацию, выдаваемую драйвером devb-* в системный журнал. Эта информация позволит установить текущую скорость работы драйвера (например, max udma 5, cur udma 3).

Примечание. В ЗОСРВ «Нейтрино» автоматически используется максимальное значение для режима UDMA, если в BIOS это максимальное значение не задано.

В табл. 15.1 указаны максимальные номера режимов и скоростей для каждого типа дискового интерфейса. Для режимов PIO, MDMA и режимов UDMA с меньшими номерами используется 40-контактный кабель. Для режимов UDMA с большими номерами требуется 80-контактный кабель.

Примечание. Под максимальной скоростью понимается максимальная теоретическая пропускная способность интерфейса при пиковых нагрузках (burst interface throughput). Пропускная способность для продолжительной нагрузки (sustained throughput) зависит от многих факторов, например от размера кэша дисковода, скорости вращения диска, от шины PCI и от файловой системы. Не следует ожидать от дисковода, работающего в режиме UDMA-6, пропускной способности для продолжительных сигналов, равной 100 Мбайт/с.

Таблица 15.1

Стандарт	PIO	MDMA	UDMA (40 контактов)	UDMA (80 контактов)	Макс. скорость, Мбайт/с
ATA	0	0	Не используется	Не используется	4
ATA 2	4	2	Не используется	Не используется	16
ATA 3	4	2	Не используется	Не используется	16
ATA 4	4	2	2	Не используется	33
ATA 5	4	2	2	4	66
ATA 6	4	2	2	5	100
ATA 7	4	2	2	6	133

- проверьте, действительно ли подсоединяемое устройство предназначено для работы в данном режиме UDMA.

- проверьте назначение первичного/вторичного и ведущего/ведомого устройств на интерфейсе. Например, установка двух жестких дисков на первичном и вторичном интерфейсах вместо ведущего/ведомого на первичном интерфейсе может привести к параллелизму в работе драйвера.

Устройства для шины SCSI

Интерфейс малых компьютерных систем SCSI (Small Computer Systems Interface), по сути, является еще одной шиной, к которой можно подключать несколько периферийных устройств. В ЗОСРВ «Нейтрино» поддерживаются многие типы SCSI-адаптеров разных производителей (см. описание devb-* в справочнике по утилитам).

Когда запускается драйвер шины SCSI, то происходит сканирование шины для обнаружения подключенных устройств. Когда драйвер обнаруживает поддерживаемое устройство, в каталоге /dev для него создается соответствующая запись (например, hdx для жесткого диска, где x — порядковый номер диска, начиная с 0).

Если драйвер не обнаруживает никаких устройств, то, возможно, причина в том, что адаптер имеет неизвестный идентификатор устройства (ID). Обычно

проблему можно разрешить, если драйверу передается необходимый идентификатор устройства или продавца. Для резидентной системы передача этих параметров происходит через процедуру diskboot (см. раздел 8).

В приводимом далее примере драйвер выполняет автоматическое сканирование подключенных SCSI-устройств и добавляет записи о найденных устройствах в каталог /dev. Например, если в системе присутствуют четыре жестких диска, то записи о них будут иметь вид:

- /dev/hd0 — первый диск с наименьшим идентификатором SCSI-устройства;
- /dev/hd1;
- /dev/hd2;
- /dev/hd3 — последний обнаруженный SCSI-диск.

При запуске драйвера в системный журнал посылается отладочная информация. Ее можно просмотреть с помощью утилиты sloginfo. Записи в этом журнале могут оказаться очень полезными, когда вы пытаетесь выяснить причину проблемы с SCSI-адаптером или SCSI-устройством.

Если драйвер неправильно обнаруживает устройства, то нужно выполнить перечисленные далее проверки.

- правильно ли установлен терминатор шины SCSI? Это часто является причиной того, что устройство показывается неправильно, появляется, а затем исчезает либо оказывается вовсе невидимым.
- поддерживается ли данный тип SCSI-адаптера? Даже если заявлено, что адаптер совместим с поддерживаемым типом, это не означает, что драйвер с этим адаптером будет работать правильно. Совместимость не означает идентичность. Для полной уверенности посмотрите, есть ли на нашем сайте идентификатор данного адаптера (см. также разд. "Общие сведения" ранее в этом разделе).
- проверьте, правильно ли распознаются SCSI-устройства в BIOS?

Если эти устройства там распознаются нормально, значит, их установки сделаны правильно, и у них нет конфликтующих идентификаторов. Это можно проверить, загрузив другую операционную систему. Если при ее загрузке все

устройства правильно обнаруживаются и не выводятся никакие сообщения об ошибках, значит, все устройства подключены правильно.

Помните о том, что при неправильной установке терминаторов SCSI-цепочки устройство может обнаруживаться в цепочке, но при его работе, скорее всего, будут возникать проблемы. Каждое устройство в SCSI-цепочке должно иметь уникальный идентификационный номер между 1 и максимальным значением, поддерживаемым адаптером (узнайте это значение из пользовательского руководства адаптера). Если два устройства имеют один и тот же идентификатор, то одно из них или оба будут работать неправильно либо они не будут распознаваться компьютером.

- нет ли проблем с PCI-мостом? Попробуйте переставить плату SCSI-адаптера в другое гнездо шины PCI. Иногда проблемы с PCI-мостом в ЗОСРВ «Нейтрино» могут мешать правильному подключению к адаптеру. Это связано с тем, что в ЗОСРВ «Нейтрино» не поддерживаются мосты с типом "other" ("другой").
- есть ли в BIOS установка для функции PnP-aware OS? ЗОСРВ «Нейтрино» не поддерживает эту функцию.
- нужен ли для правильной работы адаптера или цепочки внешний источник электропитания? В этом случае, даже если устройство имеет внешний источник, оно не сможет взаимодействовать с компьютером, если SCSI-адаптер не имеет своего питания.
- проверьте тип кабеля SCSI. Имеется несколько типов кабелей, и выбор конкретного типа кабеля зависит от типа подключаемого адаптера.

Убедитесь в том, что нет погнутых контактов в разъеме кабеля. Если вы используете специальный адаптер для согласования, например, шин SCSI 2 и SCSI 3, то проверьте, чтобы этот адаптер соответствовал рекомендованному для вашего оборудования типу. Не все адаптеры осуществляют корректно такое согласование.

Примечание. Максимальная скорость работы SCSI-устройства соответствует максимальной теоретической пропускной способности интерфейса для

пиковых нагрузок. Как указывалось ранее, пропускная способность для продолжительных нагрузок зависит от многих факторов.

Устройства типа SCSI RAID

В настоящее время в ЗОСРВ «Нейтрино» поддерживается оборудование типа RAID (Redundant Arrays of Independent Disks, избыточные массивы независимых дисков). Существует много типов сторонних производителей дисков SCSI RAID, которые можно использовать в ЗОСРВ «Нейтрино». Информацию о них можно найти в Интернете.

Устройства типа LS-120

Устройство LS-120 — это дисковод типа SuperDisk, в котором используется новая технология, позволяющая существенно улучшить позиционирование головок, что приводит к повышению емкости дискет до 120 Мбайт — значительно больше емкости обычных дискет формата 3,5". В ЗОСРВ «Нейтрино» устройство LS-120 воспринимается как диск стандарта EIDE.

Устройства типа ORB

В дисководе типа ORB используются для работы специальные сменные дискеты формата 3,5", работающие с высокой скоростью и имеющие большую емкость. Дисковод подключается к интерфейсу EIDE (ATA). Проверьте правильность подключения оборудования и правильное распознавание этого оборудования в BIOS. Дисководы типа ORB просто устанавливать, запись о них появляется в каталоге /dev как запись о жестком диске. Например:

- запись о первичном ведущем жестком диске имеет вид: /dev/hd0;
- запись о первичном ведомом диске типа ORB имеет вид: /dev/hd1.

Для монтирования дисковода типа ORB используется команда:

```
mount /dev/hd1 /fs/orb_drive
```

При смене дисков не требуется производить операцию перемонтирования.

Диски типов Zip и Jaz

Дисководы типов Zip и Jaz используют для работы специальные сменные диски, имеющие большую емкость. Диски используются для резервного копирования содержимого жестких дисков и для транспортировки файлов большого размера. Дисководы подключаются к интерфейсу EIDE (ATA). Перед началом использования дисководов проверьте правильность их подключения и правильное распознавание этого оборудования в BIOS. Дисководы типа ORB просто устанавливать, запись о них появляется в каталоге /dev как запись о жестком диске. Например:

- запись о первичном ведущем жестком диске имеет вид: /dev/hd0;
- запись о первичном ведомом диске типа Zip имеет вид: /dev/hd1.

Для монтирования дисковода типа Zip используется команда:

```
mount /dev/hd1 /fs/zip_drive
```

При смене дисков не требуется производить операцию перемонтирования.

Магнитооптические диски

Дисководы магнитооптических (MO) дисков обычно подключаются к шине SCSI или EIDE (ATA). Перед началом использования дисководов проверьте правильность их подключения и правильное распознавание этого оборудования в BIOS.

Тип драйвера зависит от того, подключается ли дисковод к интерфейсу SCSI или EIDE. Если используется интерфейс SCSI, то нужно определить для него соответствующий драйвер. Для интерфейса EIDE подходит драйвер devb-eide. Более подробно об этом сказано в разд. "Жесткие диски" ранее в этом разделе.

Драйверы для оптических дисков загружают разделяемый объект cam-optical.so, который обеспечивает метод прямого доступа для оптических дисков.

Запись о магнитооптическом диске появляется в каталоге /dev в виде /dev/mox, где x – порядковый номер диска, начиная с 0.

Для монтирования дисковода магнитооптических дисков используется команда:

```
mount /dev/mo0 /fs/mo_drive
```

При смене дисков не требуется производить операцию перемонтирования.

15.6. Виртуальные диски

Виртуальный диск – это область памяти, выглядящая как жесткий диск. Виртуальный диск в системе можно создать с использованием devb-ram, но это виртуальный диск с функционалом блочной файловой системы; по умолчанию, он инициализируется и форматируется в файловую систему QNX4 (если не указана опция ram nodinit).

Примечание. По умолчанию, io-blk.so выделяет 15% системной памяти под кеш. Драйвер devb-ram взаимодействует с io-blk.so также как любой другой драйвер диска, хотя в этом случае кеш не требуется. Чтобы установить минимальный объём кеш, можно указать опцию blk cache=512k.

Существует более удачный способ создания виртуального диска, для этого надо указать опцию blk ramdisk=... В результате будет создан внутренний псевдодиск, с которым io-blk.so работает без кеширования и будет использоваться секторы размером 4 кб.

При использовании одного из devb-* драйверов, можно скомбинировать псевдодиск с ним (указав, например, опцию blk ramdisk=10m).

Если действительно необходимо использовать devb-ram, чтобы он также предоставил внутренний псевдодиск, то применяется, например, следующая команда:

```
devb-ram disk name=ram ram capacity=0,nodinit blk
ramdisk=10m,cache=0,vnode=256
```

В этом случае, надо игнорировать устройство /dev/ram1 нулевого размера, которое создаёт devb-ram, и работать с устройством /dev/ram0, которое предоставляется io-blk.so. Необходимо проинициализировать псевдодиск перед первым использованием, например:

```
dinit /dev/ram0
mount -tqnx4 /dev/ram0
```

Такой подход увеличивает производительность поскольку исключает двойное копирование в памяти, не использует кеш и оперирует размер секторов 4 кб, что уменьшает накладные расходы.

15.6. Устройства ввода информации

Для управления вводом в графической оболочке Photon используется набор драйверов devi-*. В зависимости от присоединенного устройства ввода применяется соответствующий драйвер. В оболочке Photon может использоваться либо только мышь, либо мышь и клавиатура, либо одновременно несколько мышей (при условии, что для этого есть место).

Утилита inputtrap автоматически обнаруживает базовые устройства ввода (за исключением клавиатуры и мыши, подключенных через интерфейс USB). Эта утилита запускается в оболочке Photon после запуска графических адаптеров.

Вместо процедуры автоматического обнаружения устройств вы можете создать файл входного системного прерывания (input trap file), /etc/system/trap/input.имя_хоста (это место расположения файла по умолчанию, при необходимости его можно изменить). Каждая строка этого файла инициирует запуск драйвера:

- для драйвера devi-hirun в строке должны содержаться только аргументы, которые нужно передать. Например, такая строка соответствует запуску клавиатуры и мыши с разъемами типа PS/2:

```
kbd fd -d/dev/kbd ps2 mousedev
```

- для других драйверов входных устройств наряду с аргументами нужно указать и имя драйвера.

Мыши и клавиатуры

Для мыши и клавиатуры используется один и тот же драйвер devi-hirun. Тип подключенной к системе мыши определяет необходимый набор используемых параметров. Для последовательной мыши надо задать правильный протокол (например, протокол Microsoft Mouse).

Обнаружение клавиатуры происходит при ее подключении к таким интерфейсам:

- для клавиатуры с разъемом типа AT появляется запись /dev/kbddev;
- для клавиатуры с разъемом типа PS/2 появляется запись /dev/kbd.

Если утилита inputtrap обнаруживает наличие последовательной мыши типа Microsoft и клавиатуры, файловый дескриптор, полученный при открытии файла /dev/kbd, то запуск драйвера devi-hirun производится следующим образом:

```
devi-hirun kbd fd -d/dev/kbd msoft fd &
```

Если утилита inputtrap обнаруживает наличие мыши типа PS/2, подключенной к дополнительному порту контроллера клавиатуры (mousedev), а клавиатура подключена через первичный порт контроллера клавиатуры (kbddev), то запуск драйвера devi-hirun производится следующим образом:

```
devi-hirun kbddev ps2 mousedev &
```

После запуска мыши ее поведение можно изменить с помощью утилиты конфигурирования входных устройств (Input configuration utility) графической оболочки Photon. Эта утилита запускается либо из командной строки посредством команды input-cfg, либо на системной панели выбирается пункт **Mouse**, либо выбирается пункт меню **Launch→Configure→Mouse.**

Драйвер devc-con-hid поддерживает USB-клавиатуру в текстовом режиме (devc-con не поддерживает). Графическая оболочка Photon поддерживает USB мыши и клавиатуры; более подробную информацию см. в подразделе “Устройства USB” далее в этом разделе.

Сенсорные экраны

В ЗОСРВ «Нейтрино» осуществляется поддержка различных типов сенсорных экранов (touchscreen). Обратитесь к списку поддерживаемого оборудования на нашем сайте, чтобы узнать, какой драйвер нужно использовать. См. также описание devi-*в справочнике по утилитам. Определите, какие параметры нужно задать для установки устройства, а затем запустите нужный

драйвер. Например, драйвер для сенсорного экрана типа Dynapro SC4 запускается следующим образом:

```
devi-dyna dyna -4 fd -d/dev/ser1 &
```

По этой команде запускается драйвер devi-dyna с помощью протокола SC4 (-4) и файлового дескриптора, ассоциированного с последовательным портом 1 (fd -d/dev/ser1).

При первом запуске драйвер возвращает сообщение об ошибке, где сказано о невозможности прочитать калибрационный файл. Для калибровки сенсорного экрана после запуска оболочки Photon можно использовать утилиту calib.

15.7. Звуковые карты

Звуковая карта обнаруживается операционной системой по умолчанию. При запуске ОС служба распознавания устройств идентифицирует карту и для ее запуска используется драйвер io-audio. В ЗОСРВ «Нейтрино» драйверы звуковых карт инициализируются очень просто. При запуске драйвера io-audio для него можно использовать ключ -d:

```
io-audio -vv -d audiopc &
```

Ознакомиться с другими ключами запуска можно в руководстве "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01, в описании утилиты io-audio, а также в описании конкретной звуковой карты.

Если операционная система определяет звуковую карту неправильно, то драйвер можно запустить вручную. Для этого нужно вначале идентифицировать карту. Список поддерживаемого оборудования можно найти на нашем сайте (см. также подразд. "Общие сведения" ранее в этом разделе).

Карты для шины ISA

Карты для шины ISA могут как поддерживать, так и не поддерживать стандарт Plug and Play (PnP). Устройства, не поддерживающие этот стандарт, обычно устанавливаются вручную. Для идентификации устройства нужно иметь его описание или получить информацию у производителя (например, на его сайте).

В настоящее время в ЗОСРВ «Нейтрино» нет утилиты, которая выдает список всех установленных в системе устройств типа ISA.

Карты, не поддерживающие стандарт PnP

Для карт, не поддерживающих стандарт PnP, нужно вручную запускать драйвер, указывая адрес порта ввода/вывода (I/O), номер прерывания IRQ и номер канала DMA. Например, для запуска драйвера звуковой карты Sound Blaster используется такая команда:

```
io-audio -dsb ioport=порт,irq=прерывание,dma=канал,dma1=канал &
```

Для определения значений, которые нужно задать для порта ввода/вывода и номера прерывания, нужно визуально исследовать саму плату. Затем надо запустить драйвер, используя конфигурационные установки карты.

Проверьте, чтобы нужные вам номера порта ввода/вывода и прерывания в BIOS были зарезервированы для использования устройствами, не устанавливаемыми в шину PCI. При использовании звуковой карты типа Sound Blaster проверьте следующее:

- если драйвер не воспринимает карту, убедитесь в том, что заданный порт ввода/вывода не конфликтует с другим установленным оборудованием; попробуйте изменить значение адреса порта ввода/вывода;
- если звук вначале появляется, а затем исчезает, убедитесь в том, что используемый номер прерывания IRQ не конфликтует с другим установленным оборудованием и что он зарезервирован в BIOS; можно также попробовать изменить номер прерывания;
- если драйвер запускается корректно, но звук отсутствует, проверьте на карте установки для канала DMA, попробуйте, если возможно, изменить номер канала.

Карты, поддерживающие стандарт PnP

Конфигурация карт для шины ISA, поддерживающих стандарт PnP, должна производиться службой распознавания устройств при запуске ОС. Если этого не происходит, то, возможно, потребуется получить копию утилиты isapnp, которая

используется для инициализации карт данного вида. Данная утилита не входит в состав ЗОСРВ «Нейтрино», но ее можно найти в свободном доступе в Интернете.

Карты для шины PCI

Карты для шины PCI должны нормально запускаться службой распознавания устройств. Если звуковая карта все же не работает, попробуйте установить ее в другое гнездо. Иногда номер прерывания IRQ, установленный для данного гнезда, некорректно работает с выбранной звуковой картой.

Для получения дополнительной информации по звуковой карте используйте утилиту pci. Список поддерживаемого оборудования можно найти на нашем сайте (см. также подразд. "Общие сведения" ранее в этом разделе).

15.8. Карты для стандартов PCCARD и PCMCIA

В ЗОСРВ «Нейтрино» поддерживаются карты типов PCMCIA 1.0/2.0 и CardBUS. По умолчанию драйвер обнаруживает наличие контроллера шин ISA/PCI. Если адаптер не обнаруживается, зайдите на нашу страницу со списком поддерживаемого оборудования, чтобы проверить, поддерживается ли набор микросхем адаптера вашей карты. В настоящее время драйвер не позволяет задавать значение адреса порта ввода/вывода и номер прерывания IRQ, однако вы можете задать эти параметры непосредственно на карты.

Если драйвер не запускается, выполните следующее:

- убедитесь в том, что у сервера devp-pccard свободно окно памяти по адресу 0xD4000;
- в настройках BIOS компьютера или ноутбука проверьте, чтобы эта область памяти не кэшировалась или не использовалась другим устройством;
- проверьте, чтобы контроллер PC-карт в BIOS был установлен в режим CardBus/16bit, а не в режим PCIC.

Если набор микросхем настроен на работу в режиме, совместимом с PCIC, то работа происходит в режиме совместимости с контроллером PCMCIA для набора Intel 82365. В таком режиме набор микросхем невидим в пространстве PCI. Если же

набор микросхем настроен на работу в режиме CardBus/16bit, то он видим в пространстве PCI, и набор микросхем работает как адаптер PC-карт.

Для вывода информации о PC-карте можно воспользоваться утилитой pin. Результат вывода на экран может выглядеть следующим образом:

```
# pin
Sock     Func     Type           Flags            PID   Base    Size
IRQ
1                 Empty      -----MF------    None
1                 Empty      -----MF------    None
2        0        Network    C---I-+------    None 0x300  32      7
2                 Empty      ----MF---------  None
```

Для каждого гнезда существуют две записи, потому что драйвер (devp-pccard) поддерживает комбинированные карты, которые позволяют для каждого гнезда определить две функции. В приведенном выше примере категории означают следующее.

- Sock — гнездо, в которое установлена PC-карта. В примере сетевая карта установлена в гнездо 2;
- Func — параметр используется, если карта является многофункциональной PC-картой;
- Type — метка для функции PC-карты. Если карта является сетевой, то в этом столбце отображается значение Network;
- Flags — не установленные флаги обозначаются символом -. В табл. 15.2 приводится список обозначений для установленных флагов;

Таблица 15.2

Флаг	**Значение**
C	Карта присутствует в гнезде
B	Батарея разряжена
R	Запланировано для конфигурирования
N	Недостаточно ресурсов для конфигурирования карты
I или M	Карта ввода/вывода (I/O) или карта памяти
F	Не конфигурирована

Таблица 15.2

Флаг	**Значение**
+	Окно является частью предыдущей конфигурации
U	Данное окно невозможно блокировать
T	Окно является временным
B	Машина загрузилась с этого устройства
X или W	Заблокировано эксклюзивно/заблокировано для операций чтения/записи
R	Разрешен режим "только чтение"
L	Режим обработки прерывания IRQ при срабатывании по уровню сигнала, а не по фронту (level-mode IRQ*)
S	Разделяемое прерывание IRQ
A	Атрибутивная память (attribute memory**)
W	Расширенный (wide) доступ к памяти (16-битовый)

* Современные контроллеры прерываний имеют возможность переключения от срабатывания по перепаду входного сигнала (edge triggered interrupt mode), где входной сигнал подается прямо на вход цифровой микросхемы, к срабатыванию по уровню входного сигнала (level triggered interrupt mode), где сигнал с пологими фронтами вначале проходит через формирователь типа триггера Шмитта.

** Устройства типа PC-карт имеют два пространства памяти: атрибутивную (attribute memory) и общую (common memory).

- PID — идентификатор процесса, присоединенного к драйверу PC-карты (devp-pccard);
- Base — базовый адрес PC-карты. Эта информация полезна при запуске драйверов устройств;
- Size — число байтов в диапазоне адресов порта ввода/вывода;
- IRQ — номер прерывания IRQ PC-карты. Эта информация полезна при запуске драйверов вручную.

15.9. Устройства USB

Универсальная последовательная шина (USB) обеспечивает интерфейс с возможностью "горячей" замены для USB-устройств (например, сетевых адаптеров, входных устройств, устройств символьного ввода, звуковых устройств, хабов и

т. п.). Более подробные сведения об особенностях USB-устройств, спецификациях и др. можно найти на сайте **www.usb.org**.

Если вы не знаете, какой тип USB-устройства вы используете, то для идентификации можно воспользоваться утилитой usb:

```
usb -vvv | less
```

Результат работы команды выглядит примерно так:

```
Device Address             : 1
Vendor                     : 0x05c7 (QTRONIX)
Product                    : 0x2011 (USB Keyboard and Mouse)
Device Release             : r1.12
USB Spec Release           : v1.00
Serial Number              : N/A
Class                      : 0x00 (Independent per interface)
Max PacketSize0            : 8
Languages                  : 0x0409 (English)
Current Frame              : 511 (1024 bytes)
Configurations             : 1
  Configuration            : 1
    Attributes             : 0xa0 (Bus-powered, Remote-wakeup)
    Max Power              : 50 mA
    Interfaces             : 2
      Interface            : 0 / 0
        Class              : 0x03 (HID)
        Subclass           : 0x01 (Boot interface)
        Protocol           : 0x01 (Keyboard)
        Endpoints          : Control + 1
          Endpoint         : 0
            Attributes     : Control
            Max Packet Size: 8
          Endpoint         : 1
            Attributes     : Interrupt/IN
            Max Packet Size: 8
            Interval       : 20 ms
      Interface            : 1 / 0
        Class              : 0x03 (HID)
        Subclass           : 0x01 (Boot interface)
        Protocol           : 0x02 (Mouse)
        Endpoints          : Control + 1
```

```
Endpoint         : 0
  Attributes     : Control
  Max Packet Size: 8
```

В полях поставщика (Vendor) и продукта (Product) указываются тип устройства и, возможно, используемый в нем набор микросхем.

Наиболее часто используются USB-контроллеры следующих типов:

- UHCI (Universal Host Controller Interface, универсальный интерфейс контроллера хоста);
- EHCI (Enhanced Host Controller Interface, улучшенный интерфейс контроллера хоста);
- OHCI (Open Host Controller Interface, открытый интерфейс контроллера хоста — от сторонних производителей).

Примечание. Контроллер EHCI поддерживает только высокоскоростную передачу сигналов. Для поддержки низкоскоростных или полноскоростных устройств должен присутствовать один из контроллеров OHCI или UHCI. Если в системе отсутствует контроллер EHCI, то устройство будет работать на низкой скорости.

В операционной системе должен быть запущен стек протоколов, для того чтобы взаимодействовать с устройствами и контроллерами USB. Для этого необходимо выполнить следующие действия.

- идентифицировать контроллер.

В документации по оборудованию должен быть описан тип контроллера (OHCI, UHCI или EHCI). Если вы не знаете, какой тип контроллера используется, воспользуйтесь командой:

```
pci -vvv
```

Среди выведенной информации отыщите запись для контроллера USB, чтобы определить идентификаторы производителя/продавца и устройства. Эту информацию можно найти либо на сайте производителей (**www.usb.org**), либо, используя идентификаторы продавца и устройства, на сайте **http://www.pcidatabase.com/**.

Коды классов, выводимые командой pci –vvv, приведены в табл. 15.3.

Возможно, в системе присутствует несколько наборов микросхем, поэтому требуется загрузить несколько драйверов.

Можно также попробовать запустить просто один из стеков USB. Если это не поможет, попробуйте загрузить другой стек.

Таблица 15.3

Код класса	**Тип контроллера**
0c0300	UHCI
0c0310	OHCI
0c0320	EHCI

- войдите в систему с учетной записью root и запустите стек io-usb с соответствующим модулем:
  - для контроллера OHCI: devu-ohci.so;
  - для контроллера UHCI: devu-uhci.so;
  - для контроллера EHCI: devu-ehci.so.

При этом в каталоге /dev должна быть создана запись /dev/io-usb/io-usb.

Примечание. Если вы запускаете стек USB и драйвер в сценарии начального запуска, проверьте, чтобы использовалась команда waitfor, благодаря которой запись /dev/io-usb/io-usb появится прежде, чем будет запущен драйвер. Например:

```
io-usb -dohci
waitfor /dev/io-usb/io-usb
devu-prn
```

- после запуска стека запустите драйверы устройств так, как описано далее.

Примечание. Для USB-хабов драйверы не нужны. Эти устройства поддерживаются самим стеком.

Принтеры

Для USB-принтера запустите стек USB, а затем драйвер devu-prn. Например:

```
io-usb -dohci
waitfor /dev/io-usb/io-usb
devu-prn
```

Мыши и клавиатуры

Дравер devc-con-hid поддерживает USB клавиатуру в текстовом режиме (devc-con не поддерживает). Графическая оболочка Photon поддерживает USB Human-Interface Devices (HID) такие как клавиатура и мышь.

Для подключения к устройствам, работающим по стандарту USB HID, необходимо выполнить следующие действия.

- запустить стек USB, как описано ранее;
- запустить утилиту io-hid для загрузки модуля devh-usb.so:

```
io-hid –dusb
```

Если в вашей системе также используются последовательные входные устройства или устройства с интерфейсом PS/2, то дополнительно нужно загрузить модуль devh-ps2ser.so.

- после запуска оболочки Photon запустите драйвер devi-hid для устройств стандарта USB HID следующим образом:

```
devi-hid kbd [-u номер_устройства_USB] mouse
```

Утилита io-hid может быть запущена через файл rc.local, но этот способ нельзя применять для драйвера devi-hid, потому что во время запуска системой файла rc.local оболочка Photon еще не работает. Поэтому команду запуска драйвера devi-hid нужно включить во входной файл системного прерывания, см. описание утилиты inputtrap в руководстве "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01.

В оболочке Photon после запуска устройств для конфигурирования мыши можно использовать утилиту input-cfg. Для этого на системной панели выберите пункт меню Launch→Configure→Mouse. Для получения информации по устройствам с интерфейсом стандарта HID можно воспользоваться утилитой hidview.

Сенсорные экраны

Для сенсорных экранов, подключаемых через интерфейс USB, необходимо вначале запустить стек USB, затем утилиту io-hid, загружающую драйвер devh-usb.so. После этого запускается драйвер devi-microtouch:

```
io-hid -dusb
devi-microtouch microtouch touchusb
```

Запоминающие устройства большой емкости

Драйвер devb-umass осуществляет поддержку устройств, которые соответствуют стандарту Mass Storage Class Specification для запоминающих устройств большой емкости. Определить соответствие устройства этому стандарту можно на основе результата, выдаваемого на консоль командой usb -vv:

```
Mass Storage Class  08h
SubClass Code    Command Block Specification
     01h            Reduced Block Command (RBC)
     02h            SFF-8020i, MMC-2 (ATAPI)
     04h            UFI
     05h            SFF-8070i
     06h            SCSI transparent

Protocol Code    Protocol Implementation
     00h            Control/Bulk/Interrupt
                      (with command completion interrupt)
     01h            Control/Bulk/Interrupt
                      (with no command completion interrupt)
     50h            Bulk-Only Transport
```

Для использования в ЗОСРВ «Нейтрино» устройств памяти большой емкости, подключаемых через интерфейс USB, нужно запустить утилиту io-usb, как описано ранее, а затем запустить драйвер devb-umass. По умолчанию этот драйвер создает в каталоге /dev запись для устройств дискового типа в виде /dev/hdn, где n — порядковый номер диска. После запуска драйвера с устройством можно работать как с диском.

Например, для такого устройства, использующего контроллер UHCI, нужно выполнить команду:

```
io-usb -d uhci
devb-umass cam pnp
```

Диагностика неисправностей

- в каталоге /dev не создано устройство.

Возможно, устройство не соответствует стандарту Mass Storage Class Specification. Проверьте результат вывода по команде usb -vv.

- в каталоге /dev не создано устройство fdn для флоппи-дисковода.

Именем по умолчанию является /dev/hdn. Можно использовать ключ name в команде cam-disk.so для переопределения префикса.

15.10. Символьные устройства

Обычные последовательные адаптеры

По умолчанию драйвер последовательного порта автоматически обнаруживает порт ввода/вывода и номер соответствующего прерывания IRQ. Для стандартной PC-системы используется драйвер devc-ser8250. В документации по пакету BSP указывается драйвер, необходимый для вашей конкретной целевой системы.

Если драйвер не обнаруживает все последовательные порты, проверьте, включены ли порты в BIOS. Если порты включены, то попробуйте при запуске драйвера указать конкретные значения для адреса порта и IRQ. Для разделения адреса и номера прерывания используется запятая; для разделения пар (адрес порта — номер IRQ) вводится пробел. Например:

```
devc-ser8250 3f8,4 2f8,3
```

Примечание. Если запускается драйвер последовательного устройства для адаптера UART или модема при уже запущенном другом последовательном драйвере, то в строке запуска нужно использовать ключ -u, чтобы присвоить

номер новому драйверу. Тогда этот номер будет присоединен к имени устройства и не возникнет конфликта с существующей записью /dev/ser.

Стандартный драйвер devc-ser8250 может работать только с протоколом RS-232. В комплекте разработчика драйверов символьных устройств DDK (Character Driver Development Kit) имеется исходный текст для драйвера devc-ser8250. Его можно использовать как основу для реализации драйвера, работающего с любым другим протоколом или реализующим другие функции.

Драйвер последовательного порта может работать как в режиме программного, так и аппаратного управления потоком данных.

- для включения программного управления потоком запустите драйвер последовательного порта с ключом -s или после запуска драйвера введите команду stty:

```
stty +osflow +isflow < /dev/ser1
```

- для отключения программного управления потоком запустите драйвер с ключом -S или введите команду:

```
stty -osflow -isflow < /dev/ser1
```

- для включения аппаратного управления потоком запустите драйвер с ключом -f или введите команду:

```
stty +ohflow +ihflow < /dev/ser1
```

- для отключения аппаратного управления потоком запустите драйвер с ключом -F или введите команду:

```
stty -ohflow -ihflow < /dev/ser1
```

Примечание. В режиме редактирования (-e) управление потоком отключено. Не включайте одновременно работу с аппаратным и программным управлением потоком.

В некоторых системах интенсивное использование последовательного порта может быть накладным с точки зрения производительности, поскольку по умолчанию для каждого вводимого или передаваемого символа в

последовательном адаптере генерируется прерывание. Для уменьшения числа генерируемых прерываний можно использовать приводимые далее ключи:

- -T число — включить буфер FIFO на передачу и установить число передаваемых символов равным 1, 4, 8 или 14 для каждого сгенерированного прерывания на передачу. Значение по умолчанию равно 0 (буфер FIFO отключен);
- -t число — включить буфер FIFO на прием и установить для него порог равным 1, 4, 8 или 14 символов. Значение по умолчанию равно 0 (генерация прерывания отключена).

Наличие тайм-аута при приеме данных гарантирует, что символы не будут оставаться в буфере слишком долго. Например, устройство принимает сообщение:

Это сообщение принимается через последовательный порт.

По умолчанию для приема всего сообщения система должна обработать 54 запроса на прерывание. Если вы установите уровень срабатывания буфера приема равным 14 символам, то число обрабатываемых прерываний снизится до четырех. Это в целом повысит производительность системы, но платой за это будет надежность: чем выше порог срабатывания буфера (-t), тем больше вероятность потери данных.

Многопортовые последовательные адаптеры

Для многопортовых последовательных адаптеров может потребоваться вручную задавать в драйвере для каждого порта значения адреса порта и номера прерывания IRQ (см. примеры в предыдущем разделе). По умолчанию драйвер должен автоматически определять адреса портов и номера прерываний, но для некоторых многопортовых адаптеров служба распознавания устройств не всегда корректно идентифицирует порты.

Вы можете откорректировать работу службы распознавания устройств, чтобы она лучше работала с вашими многопортовыми картами и правильно делала бы установки для каждого порта. Для этого нужно просто отредактировать конфигурационный файл /etc/system/enum/devices/overrides. Подробнее см.

описание утилиты enum-devices в разделе 8 и в руководстве "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01.

Параллельные порты

В стандартном IBM-совместимом компьютере и в некоторых системах на базе семейства x86 для параллельных портов используется драйвер devc-par (см. документацию по пакету BSP относительно драйвера), необходимый для работы вашей целевой системы.

По умолчанию параллельный порт обнаруживается драйвером автоматически. При необходимости можно с помощью ключа -p указать адрес параллельного порта.

Если драйвер не обнаруживает параллельный порт, посмотрите, включен ли порт в BIOS. Если и это не помогает, попробуйте задать порт ввода/вывода в строке запуска драйвера.

Терминалы

В стандартном IBM-совместимом компьютере и в некоторых системах на базе семейства x86 драйвер devc-con или devc-con-hid управляет непосредственно физической консолью, в состав которой входят адаптер дисплея, экран и системная клавиатура. По умолчанию конфигурация драйвера дает возможность работы не более чем с четырьмя виртуальными консолями: /dev/con1, ..., /dev/con4. Драйвер devc-con также является драйвером для клавиатуры в текстовом режиме, если она подключается не через интерфейс USB. Драйвер запускается командой:

```
devc-con &
```

Управляющая программа devc-con-hid похожа на devc-con, но работает с привязкой к io-hid и поддерживает PS2, USB и все остальные устройства интерфейса пользователя.

Более подробное описание утилиты devc-con см. в руководстве "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01.

Атрибуты ввода/вывода

Для установки или отображения атрибутов ввода/вывода для символьного устройства (tty) используется утилита stty. Более подробно о процедуре начальной установки терминала см. в подразд. "Поддержка терминалов" раздела 4.

15.11. Сетевые адаптеры

Для установки сетевого адаптера выполняются следующие основные шаги:

- идентификация сетевой интерфейсной карты (NIC, Network Interface Card).
- запуск драйвера.
- проверка взаимодействия драйвера и оборудования.

Идентификация сетевого адаптера

В документации на адаптер должен быть указан тип используемого набора микросхем.

Если вы не знаете тип используемого в адаптере набора микросхем, его можно идентифицировать с помощью команды pci -vvv.

Среди выведенных данных найдите запись, относящуюся к сетевому контроллеру, где указаны идентификаторы производителя/продавца и устройства. Дополнительную информацию можно найти на сайте производителя или с помощью идентификатора продавца и устройства на сайте **http://www.pcidatabase.com/**.

Запуск драйвера

Запустить найденный драйвер для сетевого адаптера можно с помощью утилиты io-pkt*. Можно либо указать драйвер в качестве параметра утилиты io-pkt*, либо присоединить драйвер к уже работающей копии утилиты io-pkt*. Например, чтобы запустить io-pkt-v4-hc для модуля devn-e1900.so (адаптер 3Com 905), введите команду:

```
io-pkt-v4-hc -d el900 -t tcpip &
```

Для присоединения драйвера к работающей копии введите набор команд:

```
io-pkt-v4-hc -t tcpip &
mount -T io-pkt devn-el900.so
```

Примечание. Драйвер автоматически определяет похожие сетевые адаптеры, установленные для организации подсетей. Для монтирования различных адаптеров используется утилита mount.

Проверка правильности взаимодействия драйвера и оборудования

Для проверки работы адаптера по приему и отправке пакетов используется утилита nicinfo. Если не происходит прием пакетов на высокоскоростной сети, это может означать, что нет взаимодействия между драйвером и оборудованием. Далее приведен типовой набор данных, выдаваемых командой на экран:

```
Physical Node ID ..... 000102 C510D4
Current Physical Node ID ..... 000102 C510D4
Current Operation Rate ..... 100.00 Mb/s full-duplex
Active Interface Type ..... MII
Active PHY Address ..... 3
Power Management State ..... Active
Maximum Transmittable data Unit ..... 1514
Maximum Receivable data Unit ..... 1514
Receive Checksumming Enabled ..... TCPv6
Transmit Checksumming Enabled ..... TCPv6
Hardware Interrupt ..... 0x5
DMA Channel ..... 0
I/O Aperture ..... 0xd400 – 0xd47f
ROM Aperture ..... 0
Memory Aperture ..... 0xe6000000 – 0xe6000FFF
Promiscuous Mode ..... Off
Multicast Support ..... Enabled

Packets Transmitted OK ..... 104
Bytes Transmitted OK ..... 10067
Broadcast Packets Transmitted OK ..... 6
Multicast Packets Transmitted OK ..... 1
Memory Allocation Failures on Transmit ..... 0
```

```
Packets Received OK ..................... 1443
Bytes Received OK ....................... 168393
Broadcast Packets Received OK ............ 427970
Multicast Packets Received OK ............ 37596
Memory Allocation Failures on Receive ..... 0

Single Collisions on Transmit ............ 0
Multiple Collisions on Transmit .......... 0
Deferred Transmits ...................... 0
Late Collision on Transmit errors ........ 0
Transmits aborted (excessive collisions) ... 0
Transmits aborted (excessive deferrals) .... 0
Transmit Underruns ...................... 0
No Carrier on Transmit .................. 0
Jabber detected ......................... 0
Receive Alignment errors ................ 0
Received packets with CRC errors ........ 0
Packets Dropped on receive .............. 0
Ethernet Headers out of range ........... 0
Oversized Packets received .............. 0
Frames with Dribble Bits ................ 0
Total Frames experiencing Collision(s) ..... 0
```

Примечание. Результаты вывода по команде nicinfo зависят от того, какие параметры контролируются драйвером. Не все драйверы обеспечивают вывод всех приведенных полей. Тем не менее, всегда представляется информация о принятых и отправленных байтах и пакетах.

Далее описаны категории параметров, представленные выше в примере. Если возникают какие-либо сетевые проблемы, то прежде всего обратите внимание на следующие категории параметров:

- Physical Node ID (Идентификатор физического узла);
- Hardware Interrupt (Аппаратное прерывание);
- I/O Aperture (Диапазон адресов ввода/вывода);
- Packets Transmitted OK (Правильно переданные пакеты);
- Total Packets Transmitted Bad (Общее число ошибочно переданных пакетов);
- Packets Received OK (Правильно принятые пакеты);

- Received packets with CRC errors (Ошибки CRC при приеме пакетов).

Physical Node ID (Идентификатор физического узла)

Идентификатор физического узла известен как MAC-адрес карты (Media Access Control, управление доступом к среде передачи). Это значение уникально для каждой сетевой карты, хотя для некоторых моделей допускается назначать свои собственные адреса. Правда, такое встречается редко и, как правило, во встраиваемых системах.

Если этот адрес представлен значением FFFFFF FFFFFF или 000000 000000, то, скорее всего, что-то не так с аппаратными установками, либо требуется отдельное назначение MAC-адреса для карты. Обратитесь к руководству пользователя для адаптера, чтобы уточнить этот вопрос.

Примечание. Если оборудование установлено неправильно, то MAC-адрес может принимать иные значения.

Первые шесть символов MAC-адреса являются идентификатором продавца. Проверьте правильность этого идентификатора, обратившись к странице **http://www.cavebear.com/CaveBear/Ethernet/vendor.html** со списком зарегистрированных продавцов. Последние шесть символов соответствуют идентификатору карты. Это число должно быть похоже на случайное. Например, значение вроде 444444, скорее всего, некорректно.

Current Physical Node ID (Идентификатор текущего физического узла)

Идентификатор текущего физического узла показывается в случае, если карта настроена для имитации ("spoof") идентификатора другой карты. Обычно этот параметр передается драйверу и означает, что узел представлен именно таким фактическим идентификатором. В зависимости от типа карты некоторые драйверы принимают такую подмену. На более высоком (программном) уровне эта подмена (спуфинг) означает фильтрацию пакетов, предназначенных для данного узла. Этот метод работает значительно медленнее, чем если бы фильтрация производилась на аппаратном уровне. Поскольку в этом случае карта должна быть установлена в

смешанный (promiscuous) режим работы, то она должна принимать все входящие пакеты и использовать для их сортировки программный режим.

Аналогию описанной ситуации может найти в работе почтовой системы. Если мы хотим "притвориться" кем-нибудь (т. е. читать чужую почту), то нужно получать с почтового отделения всю корреспонденцию. Однако затем придется всю полученную корреспонденцию сортировать. Это займет гораздо больше времени, чем если бы в почтовом отделении проводилась предварительная сортировка, а вы получали бы только корреспонденцию, адресованную непосредственно вам. Подробнее об этом см. в подразд. "Promiscuous Mode (Смешанный режим работы)" далее в этом разделе.

Current Operation Rate (Текущая скорость работы)

Скорость передачи означает скорость, с которой сетевая карта передает данные. Для большинства карт это значение равно 10 или 100 Мбит/с. Рядом с этим значением указывается также тип связи, используемый картой. В большинстве случаев используется дуплексный (full-duplex) или полудуплексный (half-duplex) режим передачи:

- при дуплексном режиме данные могут передаваться одновременно в обоих направлениях;
- при полудуплексном режиме данные могут передаваться в обоих направлениях, но не одновременно.

Простой иллюстрацией этих режимов может быть сравнение с дорогой. По двухполосной дороге (дуплексный режим) машины могут одновременно двигаться в обоих направлениях, не залезая на полосу другого направления. По однополосной дороге (полудуплексный режим) в каждый момент времени машина может двигаться только в одном направлении.

Когда исследуется скорость передачи, нужно проверять собственно скорость работы, учитывать режим дуплексной или полудуплексной работы, а также режим работы, поддерживаемый хабом. Не все хабы поддерживают дуплексный режим работы.

Active Interface Type (Тип активного интерфейса)

Для данного параметра указывается тип физического интерфейса, используемого адаптером Ethernet. Обычно используются интерфейсы типов UTP (unshielded twisted pair, неэкранированная витая пара), STP (shielded twisted pair, экранированная витая пара), оптоволокно, AUI (attachment unit interface, интерфейс подключаемых устройств), MII, или BNC (коаксиальный кабель).

Active PHY Address (Активный физический адрес)

Идентификатор PHY указывает, какая часть физического адреса используется для работы в сети. Число принимает значения от 0 до 31 и изменяется в зависимости от того, задан ли параметр PHY конкретным значением или он выбирается драйвером по умолчанию (в зависимости от типа карты).

Power Management State (Состояние управления питанием)

Этот параметр определяет текущий статус подачи питания на сетевой адаптер: Off (отключено), Standby (в резерве), Idle (режим ожидания) или Active (активный). Если пакеты не передаются или не принимаются, убедитесь, что для карты установлен режим Active. Если это не так, то в вашей системе, возможно, есть неполадки в системе управления питанием.

Maximum Transmittable data Unit (Максимальный размер передаваемого блока данных)

Максимальный размер передаваемого блока данных (MTU) определяет максимальный размер кадра, который может быть послан по физической среде передачи. При отладке этот параметр обычно не нужен, но он может быть полезен при оптимизации работы сети. Значение 0 является недопустимым и служит признаком того, что карта установлена неправильно. Значение по умолчанию равно 1514.

Maximum Receivable data Unit (Максимальный размер принимаемого блока данных)

Этот параметр, аналогичный MTU, только он относится к принимаемому кадру. Значение по умолчанию равно 1514.

Receive Checksumming Enabled (Включено вычисление контрольной суммы при приеме), Transmit Checksumming Enabled (Включено вычисление контрольной суммы при передаче)

Эта возможность поддерживается не всеми типами карт. Если адаптер поддерживает такую возможность, то будет указано, какой метод вычисления контрольной суммы нужно использовать: IPv4, TCPv4, UDPv4, TCPv6 или UDPv6.

Hardware Interrupt (Аппаратное прерывание)

Аппаратное прерывание — это линия запроса на прерывания (IRQ) вашей сетевой карты. Назначение конкретного номера IRQ зависит от типа интерфейса карты: PCI или ISA. В случае интерфейса PCI значение IRQ назначается утилитой pci-bios. Для карт ISA значение IRQ жестко задано перемычками на самой плате.

Примечание. Карты с интерфейсом ISA не могут совместно использовать один и тот же номер IRQ. Для карт с интерфейсом PCI такая возможность существует.

DMA Channel (Канал DMA)

Параметр указывает на используемый картой номер канала прямого доступа в память (DMA). Значение определяется картой и тем каналом, который на ней используется.

I/O Aperture (Диапазон адресов ввода/вывода)

Это шестнадцатеричное значение показывает занимаемое картой пространство адресов для портов ввода/вывода. Диапазон адресов ввода/вывода используется для локализации портов и для их отображения на данное адресное пространство. Диапазон зависит от платформы.

Memory Aperture (Диапазон адресов ОЗУ)

Это шестнадцатеричное значение показывает занимаемое картой пространство адресов ОЗУ. Диапазон адресов ОЗУ используется для локализации

ОЗУ и для его отображения на данное адресное пространство. Диапазон зависит от платформы.

ROM Aperture (Диапазон адресов ПЗУ)

Это шестнадцатеричное значение показывает занимаемое картой пространство адресов ПЗУ. Диапазон адресов ПЗУ используется для локализации ПЗУ и для его отображения на данное адресное пространство. Диапазон зависит от платформы.

Promiscuous Mode (Смешанный режим работы)

Когда карта установлена в смешанный (promiscuous) режим работы, то она должна принимать из сети любой пакет Ethernet. Это весьма накладно при нормальной работе системы, но является обычной практикой при работе в режиме отладки.

Кроме того, при задании смешанного режима работы сетевой MAC-адрес карты может имитироваться (spoof), т. е. карта будет воспринимать все пакеты, как адресованные, так и не адресованные к ней. Тогда на более высоком (программном) уровне можно воспринимать пакеты, адресованные кому угодно. По умолчанию смешанный режим работы отключен.

Multicast Support (Поддержка многоадресных пакетов)

При включении многоадресного режима передачи пакет можно пометить специальным адресом назначения так, что его смогут принять многие узлы сети. Допускаются также многоадресные пакеты.

Packets Transmitted OK (Правильно переданные пакеты)

Прежде чем смотреть значение этого параметра, попробуйте вначале воспользоваться некоторыми видами передачи данных по сети (ping, telnet, передача файлов). Если карта установлена неправильно, то число указанных в этом параметре пакетов имеет очень маленькое значение или нулевое. Если через карту не отправляются пакеты, то дело, скорее всего, в драйвере. Проверьте все ключи и

параметры, передаваемые при запуске драйвера. Возможно, в каком-то из них вы обнаружите ошибку.

Bytes Transmitted OK (Правильно переданные байты)

Этот параметр указывает число переданных в сеть байтов данных. Число возрастает по мере роста числа переданных пакетов.

Total Packets Transmitted Bad(Общее число ошибочно переданных пакетов)

Эту статистику можно использовать для определения наличия сбоев в работе оборудования. Если все отправляемые пакеты оказываются ошибочными, то, вероятно, это аппаратная проблема, хотя возможная причина может быть в использовании неправильного драйвера. Проверьте совместимость оборудования. Если все указывает на аппаратные проблемы, попробуйте поменять плату, возможно проблема исчезнет.

Broadcast Packets Transmitted OK (Правильно переданные широковещательные пакеты)

В этом параметре указывается число переданных с сетевой карты широковещательных пакетов.

Multicast Packets Transmitted OK (Правильно переданные многоадресные пакеты)

В этом параметре указывается число переданных с сетевой карты многоадресных пакетов.

Memory Allocation Failures on Transmit (Сбои при выделении памяти во время передачи)

Перед началом передачи драйвером резервируется системная память для буфера, в котором сохраняются данные для передачи. При готовности карты в нее пересылается содержимое буфера.

Если возникает ошибка выделения памяти, то, скорее всего, причиной является недостаток памяти в системе. Проверьте, имеется ли в системе

достаточное количество памяти. Если ошибка выделения памяти возникает регулярно, то рассмотрите возможность наращивания памяти в системе. Другой причиной может быть наличие утечек памяти в системе, из-за этого системная память медленно "съедается" до полного истощения ресурса.

Packets Received OK (Правильно принятые пакеты)

Значение этого параметра соответствует числу успешно принятых от карты пакетов. Если возникают проблемы с приемом данных, то следует проверить соединение кабелей и хаба. Возможно, что проблемы при приеме связаны с драйвером. Драйвер может быть правильно установлен, и он обеспечивает нормальную передачу данных, но не обеспечивает их приема. Если данные принимаются, но не отправляются, то, как правило, причиной этого является некорректная работа драйвера. Проверьте правильность параметров драйвера. Обратитесь к системному журналу, используя утилиту sloginfo, может быть, там вы найдете необходимую подсказку.

Bytes Received OK (Правильно принятые байты)

Этот параметр соответствует числу принятых из сети байтов данных. Значение параметра возрастает по мере роста числа принятых пакетов.

Single Collisions on Transmit (Одиночные конфликты при передаче)

Данный параметр соответствует числу конфликтов (collisions), которые возникали при попытке передачи кадров.

В сетевой карте осуществляется контроль несущей, когда оказывается, что сеть какое-то время не используется. После этого начинается передача кадра данных. Проблема возникает, когда две карты одновременно начинают передачу, осуществляя при этом контроль несущей. Такого рода ошибка обычно возникает на сетях с большой занятостью.

Когда на сетевых картах обнаруживается конфликт такого рода, то они обе прекращают передачу и переходят в режим ожидания на случайно выбираемый период времени. Для обеих сетевых карт выбираемые периоды времени оказываются различными. Поэтому теоретически после окончания времени

ожидания первой сетевой карты другая сетевая карта либо уже начнет передачу, либо будет по-прежнему находиться в состоянии ожидания. Так удается избегать конфликтных ситуаций.

Проблемы такого рода в меньшей степени свойственны сетям с дуплексным режимом работы.

Multiple Collisions on Transmit (Многократные конфликты при передаче)

Данная ошибка возникает в случае, когда при попытке передачи подряд возникает несколько конфликтных ситуаций, даже несмотря на ожидание в течение нескольких случайно заданных периодов времени. Такая проблема обычно характерна для занятых сетей, работающих в полудуплексном режиме. При наличии большого числа ошибок такого рода попробуйте перейти на работу в дуплексном режиме. Если сеть работает по протоколу TCP/IP, то вместо хабов можно попробовать установить коммутаторы (switch).

Deferred Transmits (Отложенные передачи)

Ненулевое значение этого параметра обычно встречается в сетях, работающих в полудуплексном режиме, причем это не означает, что имеются какие-либо проблемы. Просто с сетевой карты делается попытка отправить данные в сетевой кабель, но сеть оказывается занятой передачей по кабелю других данных. Поэтому нужно просто подождать некоторый случайный промежуток времени. Значение этого параметра может быть большим при сильной занятости сети.

Late Collision on Transmit errors (Ошибки поздней занятости при передаче)

Ошибки поздней занятости встречаются в случае, когда карта передала большую часть кадра, поэтому в сети должно быть известно, что сеть занята, но, тем не менее, другая система в сети все равно пытается начать передачу кадра по той же линии. Эта ситуация похожа на обычные ошибки конфликта доступа, они просто были обнаружены слишком поздно.

В зависимости от протокола этот тип ошибок может вредно сказываться на общей пропускной способности протокола. Например, потеря 1% пакетов для

протокола NFS при использовании применяемых по умолчанию таймеров повторной передачи оказывается достаточной для уменьшения скорости передачи приблизительно на 90%. Если в сети наблюдается низкая пропускная способность, проверьте, нет ли избыточного количества ошибок описанного типа. Обычно адаптеры Ethernet не делают повторной передачи кадров, потерянных в результате появления ошибок поздней занятости.

Появление таких ошибок является признаком того, что время распространения сигнала по сети больше, чем время, необходимое для выдачи сетевой картой в сеть всего пакета. Поэтому мешающая работе система не знает, что сеть постоянно используется, следовательно, она пытается разместить в сети свой новый кадр.

Узлы, которые пытаются одновременно занять сеть, обнаруживают ошибку после первого временного сегмента в 64 байта. Это означает, что сетевая карта обнаруживает конфликты поздней занятости только после передачи кадра длиннее 64 байт. Проблема состоит в том, что при передаче кадра менее 64 байт сетевая карта не способна обнаружить ошибку. Обычно если возникают ошибки поздней занятости в сети с кадрами большого размера, то, скорее всего, эти же ошибки и возникают при малом размере кадра.

Появление таких ошибок обычно связано с кабелями Ethernet, длина которых оказывается более значения, указываемого стандартом IEEE 802.3, или же превышено максимальное число используемых кабелей конкретного типа, или же между двумя узлами сети использовано избыточное число повторителей.

Другой причиной возникновения такого вида ошибок является то, что в сети имеется неправильно работающее оборудование, которое посылает в сеть поврежденные кадры, которые воспринимаются как конфликтные фрагменты. Эти поврежденные фрагменты могут иногда появляться в сетевой карте после возникновения ошибок поздней занятости.

Transmits aborted (excessive collisions) (Прекращение передачи — слишком большое число конфликтов)

Данная ошибка встречается при возникновении в сети слишком большого числа конфликтов. Сетевой карте не удается передать кадр в сеть после возникновения 16 конфликтов. Это означает, что сеть "заклинило" или сеть перегружена.

Примечание. Иногда маршрутизаторы не могут передать кадр, если в них возникают ситуации с избыточными конфликтами. Однако вместо того, чтобы послать уведомление на соответствующий передающий узел, маршрутизатор просто отбрасывает кадр.

При частом возникновении ошибок подобного рода нужно пересмотреть топологию сети: уменьшить размер сети, вставить коммутатор в стратегически важные места, что позволит уменьшить число посылаемых в общую сеть пакетов. Проблема может быть разрешена переходом в режим дуплексной работы.

Transmits aborted (excessive deferrals) (Прекращение передачи — слишком большое число отложенных передач)

Прекращение передачи из-за слишком большого числа отложенных передач означает, что сетевой карте не удается передать кадр из-за чрезвычайной занятости сети. Решением проблемы может оказаться переход в режим дуплексной передачи.

Transmit Underruns (Передачи с недозагрузкой)

Этот тип ошибки может встречаться в микросхемах, в которых предусмотрено использование механизма прямого доступа в память (DMA). При использовании механизма DMA пакеты копируются в буфер FIFO, из которого данные передаются в линию. При оборудовании низкой производительности система DMA оказывается неспособной заполнять буфер FIFO с той скоростью, как они передаются в линию, поэтому возникает ошибка недозагрузки, и передача прекращается.

No Carrier on Transmit (Нет несущей при передаче)

Когда карта памяти собирается начать передачу кадра, то вначале выполняется контроль несущей (это похоже на то, что, подняв трубку телефона перед набором номера, вы проверяете наличие тонального сигнала готовности

линии от станции). При передаче кадра сетевая карта "прослушивает" линию с точки зрения появления возможных конфликтов или ошибок. Ошибки возникают в случае, когда сетевая карта передает кадр в сеть, и вдруг выясняется, что не удается обнаружить ее собственную несущую (по аналогии с телефоном: при наборе номера вы слышите ответную реакцию на каждую нажимаемую цифровую клавишу).

Ошибки такого рода возникают при отключении и подключении сетевых кабелей или в результате малой оптической мощности оптоволоконного трансивера (Fiber Optic Transceiver, FOT).

Jabber detected (Обнаружение сбойного пакета)

Обычно такая ошибка возникает в сетях со скоростью передачи 10 Мбит. Появление ошибки указывает на то, что сетевая карта продолжает передачу после того, как пакет уже был послан. Для более скоростных сетей эта ошибка не встречается, поскольку в этих сетях допускается больший размер кадра.

Receive Alignment errors (Ошибки выравнивания при приеме)

Появление ошибки выравнивания при приеме означает, что карта приняла из сети поврежденный пакет. Появление этой ошибки сопровождается также появлением ошибки FCS (Frame Check Sequence, контрольная последовательность кадра). Эти ошибки встречаются в случае, когда размер принимаемого кадра не кратен восьми битам (одному байту).

Обычно эти ошибки являются следствием некачественного подключения кабеля, использования кабелей не в соответствии со стандартом IEEE 802.3, неисправности сетевой карты, а возможно, и неисправности хаба или коммутатора. Для локализации места возникновения проблемы используйте отключение фрагментов сети по методу бинарного дерева.

Received packets with CRC errors (Ошибки CRC при приеме пакетов)

Запись в этом поле указывает на то, сколько раз (на аппаратном уровне) сетевая карта приняла искаженные данные. Наличие неправильно принятых данных может быть вызвано неисправностью кабеля, хаба или сетевой карты.

Наилучшим способом выявления источника появления ошибок CRC является отключение фрагментов сети по методу бинарного дерева. Начав такую процедуру можно заменять оборудование одно за другим. Поскольку эта ошибка является ошибкой на стороне приема, то весьма трудно определить, была ли ошибка CRC при отправке пакета на передающей стороне.

Packets Dropped on receive (Отбрасывание пакетов при приеме)

Обычно это означает, что возникло переполнение при приеме пакета. Это имеет отношение к работе системы DMA и буфера FIFO (аналогично ошибке Transmit Underruns — передача с недозагрузкой), только на этот раз система DMA не успевает копировать пакеты в память с той скоростью, как они поступают из сети. В результате часть пакетов отбрасывается. Как и в случае передачи с недозагрузкой, обычно причина ошибок кроется в низкой производительности оборудования.

Ethernet Headers out of range (Выход за допустимые пределы заголовков Ethernet)

В этой записи указывается число пакетов, для которых поле типа/длины протокола Ethernet оказывается недействительным.

Oversized Packets received (Прием пакетов слишком большого размера)

Пакеты слишком большого размера обычно обнаруживаются на стороне приема. Принимаемый пакет оказывается больше заданного размера приемного буфера драйвера.

Frames with Dribble Bits (Кадры с избыточными битами)

Избыточные биты данных принимаются после приема значения CRC протокола Ethernet. Наличие таких битов обычно вызывается неисправным оборудованием или проблемами кабельного подсоединения сети Ethernet, которые не соответствуют стандарту IEEE 802.3.

Total Frames experiencing Collision(s) (Общее число кадров с конфликтами)

Значение этого параметра соответствует общему количеству кадров, где возникали конфликтные ситуации при попытке их передачи в сеть. Это значение иногда может быть большим, все зависит от степени загруженности сети. При загруженной сети этот тип ошибок встречается чаще, чем в сети с небольшим трафиком.

15.12. Модемы

Модемы могут быть следующих типов:

- внутренние (на основе интерфейса ISA с поддержкой Plug and Play или без нее);
- на основе PCI;
- внешние;
- кабельные.

Внутренние модемы

Для внутренних модемов может использоваться интерфейс ISA, причем модемы могут поддерживать или не поддерживать стандарт Plug and Play (PnP). Устройства, не поддерживающие этот стандарт, должны устанавливаться вручную.

Примечание. Для идентификации устройства необходимо иметь документацию на него или связаться с производителем устройства. В настоящее время в составе ЗОСРВ «Нейтрино» нет утилиты, с помощью которой можно было бы получить список установленных в системе ISA-устройств.

Модемы с интерфейсом ISA без поддержки PnP

Для конфигурирования модема необходимо использовать адрес порта ввода/вывода и номер прерывания IRQ, которые не вызывают конфликта в системе.

Драйвер devc-ser8250 должен произвести автоматическое определение модема, и в каталоге /dev должна появиться запись о нем в виде serx, где x — целое число.

Примечание. Такое имя может иметь более одной записи. Имейте в виду, что первые две записи представляют порты comm. Любая дополнительная запись, скорее всего, относится к модему. При сомнениях можно проверить все записи ser с помощью утилиты qtalk. Более подробно этот вопрос рассмотрен в разд. "Тестирование модемов" далее в этом разделе.

Записи обычно имеют такой вид:

```
Comm1 is enabled in the BIOS
Comm1 is disabled
Modem is configured to Comm2's ioport and IRQ
```

В каталоге /dev при этом можно будет увидеть следующее:

- ser1 — Comm1;
- ser2 — модем.

Модемы с интерфейсом ISA и поддержкой PnP

Если модем ISA поддерживает стандарт PnP и с помощью перемычек на его плате можно вручную установить значения для порта ввода/вывода и прерывания IRQ, то предпочтительнее использовать все же ручной метод, а не Plug and Play.

Драйвер devc-ser8250 должен произвести автоматическое определение модема, и в каталоге /dev должна появиться запись о нем в виде serx, где x — целое число.

Примечание. С именем ser в каталоге /dev может существовать более одной записи. Имейте в виду, что первые две записи представляют порты comm. Любая дополнительная запись, скорее всего, относится к модему. Если по этому поводу есть сомнения, то можно проверить все записи ser с помощью утилиты qtalk. Более подробно этот вопрос рассмотрен в разд. "Тестирование модемов" далее в этом разделе.

Если модем не обнаруживается, то для определения номера порта ввода/вывода и прерывания IRQ воспользуйтесь утилитой isapnp, после чего используйте эти значения для запуска драйвера devc-ser8250.

Примечание. Если вы запускаете драйвер последовательного канала для UART или модема, при этом уже был запущен другой драйвер последовательного канала, то необходимо использовать при запуске ключ -u, чтобы присвоить номер новому драйверу. Этот номер присоединяется к имени устройства, так что не возникнет конфликта с ранее существовавшими в каталоге /dev/ser записями.

Модемы на картах PCI

Драйвер devc-ser8250 должен произвести автоматическое определение модема, и в каталоге /dev должна появиться запись о нем в виде serx, где x — целое число.

Если не создано никакой записи, запустите утилиту pci -vvv. Она выведет на экран информацию, где можно увидеть назначенные для модема адрес порта ввода/вывода и номер прерывания IRQ. Используйте эти данные для запуска драйвера devc-ser8250. Если вы используете правильные значения для порта и прерывания, то в каталоге /dev будет создана корректная запись.

Внешние модемы

Внешние модемы устанавливаются очень легко. В каталоге /dev можно найти последовательный порт, к которому подключен модем. Модем подключается к разъему на задней панели системы. Если модем подключен к последовательному порту 1, то в каталоге /dev для него будет присутствовать запись с именем ser1.

Кабельные модемы / ISDN

Будем предполагать, что кабельный модем подключается к вашей системе через сетевую карту, и драйвер карты уже был запущен и нормально работает. Если это не так, то см. подразд. "Сетевые адаптеры" ранее в этом разделе.

Для задания конфигурации воспользуйтесь инструментами конфигурирования протокола TCP/IP. Это можно сделать путем запуска из командной строки утилиты phlip, выбором пункта Network на системной панели оболочки Photon или выбором пункта меню **Launch→Configure→Network**. После этого выполните описанные далее действия.

Перейдите на вкладку **Devices** и задайте следующие установки:

- выберите en0;
- выберите для соединения службу DHCP;
- в поле Server введите идентификатор машины, предоставленный вам оператором кабельной сети.

Перейдите на вкладку **Network** и задайте следующие установки:

- используйте идентификатор машины в качестве имени хоста;
- установите имя домена, которое вам предоставил оператор кабельной сети;
- установите IP-адрес шлюза, предоставленный вам оператором кабельной сети;
- маска по умолчанию (0.0.0.0) заполняется автоматически;
- в поле **Name Servers** добавьте любой известный вам IP-адрес DNS-сервера (можно больше, чем один); первым должен стоять адрес, предоставленный вам оператором кабельной сети.

Перезагрузите компьютер. Служба DHCP будет запущена автоматически.

Тестирование модемов

Для тестирования модема можно использовать утилиту qtalk.

- убедитесь в том, что модем подключен к телефонной линии;

- введите команду stty для установки скорости работы модема в бодах. Например, для установки скорости 57 760 (такую скорость используют модемы 56К) для модема, установленного в каталог /dev/ser1, нужно ввести:

```
stty baud=57760 < /dev/ser1
```

- введите команду qtalk -myстройство, где устройство — имя последовательного порта (например, /dev/ser1);
- введите команду at. От модема должен быть получен ответ OK.

Устранение неисправностей модемов

Если вы следовали вышеописанным инструкциям, но от модема не получен ответ OK, нужно попробовать выполнить следующие действия.

- проверьте правильность установки скорости работы модема.
- проверьте, подключен ли модем.
- проверьте, не относится ли модем к программному типу (software modem).

В ЗОСРВ «Нейтрино» не поддерживаются Win-модемы и HSP-модемы (Host Signal Processor, модем на основе сигнального процессора), которые известны как soft-модемы. ЗОСРВ «Нейтрино» может работать с PnP-модемами, но тогда в BIOS нужно отключить функцию определения ОС с автоматическим распознаванием PnP-устройств.

- нет ли конфликта модема с другим устройством, имеющим то же значение адреса порта ввода/вывода или номера прерывания IRQ? Если модем является внутренним с интерфейсом ISA, то, возможно, потребуется зарезервировать в BIOS диапазон адресов ввода/вывода и значение прерывания IRQ, чтобы эти значения не использовались PCI-устройствами.
- отключили ли вы в BIOS работу COM-порта, если вы используете для модема адреса ввода/вывода и номер прерывания этого COM-порта? Это требование применимо только к внутренним модемам.

15.13. Видеокарты

На нашем сайте можно найти список видеокарт, поддерживаемых в ЗОСРВ «Нейтрино» (см. подразд. "Общие сведения" ранее в этом разделе).

Изменение видеорежимов в оболочке Photon

Для изменения видеорежимов графического адаптера в оболочке Photon можно использовать инструмент Display Configuration (Конфигурация дисплея). Это можно сделать либо через команду phgrafx в командной строке, либо выбрав пункт **Graphics** в системной панели, либо выбрав пункт меню **Launch→Configure→Display.**

Для изменения видеорежима необходимо выполнить следующие действия.

- выбрать нужную конфигурацию.
- щелкнуть по кнопке **Apply** (Применить).

Вначале появится черный экран, после чего должен произойти переход в новый режим. Если выбранный новый режим работает некорректно, нужно подождать 15 секунд, затем дисплей автоматически вернется к предыдущим установкам. Можно просто нажать клавишу <Esc> или <Enter>.

- если режим изменился и работает нормально, щелкните по кнопке **Accept** (Принять).

С помощью данной утилиты можно также поменять драйвер, разрешение, частоту обновления экрана, палитру (только для режима с 8-битовой палитрой) и отключить аппаратный курсор.

Дополнительную информацию о изменении командной строки для запуска адаптера дисплея см. в разделе о phgrafx.

Ручная настройка видеокарты

Итак, ручная настройка видеокарты выполняется так:

- идентифицируйте тип видеоадаптера. В документации должен быть описан набор используемых в нем микросхем.
- отредактируйте файл конфигурации /etc/system/config/display.conf в соответствии с особенностями оборудования. Более подробную информацию о драйвере графических устройств см. в подразделе «Конфигурационный файл io-display».

Примечание. В этом случае, можно просто перезапустить систему, чтобы новая конфигурация была распознана. Следующие шаги описывают, как вручную остановить работу и перезагрузить драйвер.

- в текстовом режиме завершить работу io-display, и убедиться, что процесс завершен.

Примечание. При выполнении данных действий Photon может быть не запущен.

- запустите io-display -dvid=0xVVVV,did=0xDDDD где VVVV идентификатор производителя и DDDD идентификатор устройства видео-карты.
- перезапустить photon, запустив /usr/bin/ph.

Информацию о данных опциях см. описание графических драйверов devg-* в справке по утилитам, а также в описании io-display.

Установка нескольких дисплеев

Для установки двух или нескольких дисплеев можно использовать конфигурационный файл io-display. Можно выполнить конфигурирование двух или более видеокарт для работы с несколькими дисплеями либо использовать одну видеокарту для поддержки нескольких дисплеев.

GF

В целях поддержки множественных дисплеев в GF, необходимо вручную создать конфигурационный файл display.conf перед запуском io-display.

Следующий пример показывает конфигурационный файл, устанавливающий возможность работы с двумя дисплеями, с использованием графического драйвера poulsbo:

```
device {
drivername=poulsbo
vid=0x8086
did=0x8108
modeopts=/etc/system/config/poulsbo.conf
deviceindex=0
display {
```

```
xres=800
yres=480
refresh=60
pixel_format=argb8888
}
display {
xres=800
yres=480
refresh=60
pixel_format=argb8888
}
}
```

Photon

Следующие примеры описывают, как сконфигурировать множественные дисплеи, используя Photon.

Конфигурация множественных дисплеев с множественными видео-картами

При существовании множественных видео-устройств, конфигурационный файл, содержащий записи для каждого устройства, автоматически конфигурируется при загрузке. По умолчанию, используется видео-карта со значением дисплея index 0. Для установки драйвера устройства для разрешения использования множественных дисплеев, необходимо запустить phgrafx, и удостовериться, что установлено Enable. Также для конфигурации части параметров и установки режима для каждого устройства можно использовать phgrafx.

Следующий пример конфигурационного файла отображает использование видео-карт Radeon и ATI-rage128 с выходом на два дисплея:

```
device {
photon {
driver {
drivername=svga
modeopts=
```

```
}
driver {
drivername=ati_rage128
modeopts=
}
driver {
drivername=vesabios
modeopts=
}
}
drivername=ati_rage128
modeopts=
vid=0x1002
did=0x5050
deviceindex=0x0
display {
xres=640
yres=480
refresh=60
pixel_format=argb1555
photon {
enabled=1
xoffset=1024
yoffset=0
cursor=hardware
input_group=1
}
}
}
device {
photon {
driver {
drivername=svga
modeopts=
}
driver {
```

```
drivername=radeon
modeopts=
}
driver {
drivername=vesabios
modeopts=
}
}
drivername=radeon
modeopts=
vid=0x1002
did=0x4966
deviceindex=0x0
display {
xres=1024
yres=768
refresh=60
pixel_format=argb8888
photon {
enabled=1
xoffset=0
yoffset=0
cursor=hardware
input_group=1
}
}
}
```

Конфигурация множественных дисплеев с использованием одной видео-карты

Одна видео-карта может осуществлять поддержку множественных дисплеев (при этом в видео-карте должны быть множественные выходы). Для этого необходимо вручную создать в файле конфигурации дополнительный раздел для display. Необходимо помнить, что для распознавания новой конфигурации необходима перезагрузка компьютера.

Следующий пример отображает файл конфигурации для видео-карты Matroxg и двух дисплеев.

```
device {
photon {
driver {
drivername=svga
modeopts=
}
driver {
drivername=matroxg
modeopts=
}
driver {
drivername=vesabios
modeopts=
}
}
drivername=matroxg
modeopts=
vid=0x102b
did=0x525
deviceindex=0x0
display {
xres=1024
yres=768
refresh=60
pixel_format=rgb565
photon {
enabled=1
xoffset=0
yoffset=0
cursor=hardware
input_group=1
}
}
display {
```

```
xres=640
yres=480
refresh=60
pixel_format=rgb565
photon {
enabled=1
xoffset=0
yoffset=0
cursor=hardware
input_group=1
}
}
}
```

Примечание. По умолчанию, в автоматически-созданном конфигурационном файле видео-карта с множественными выходами посылает одинаковый видео-сигнал на все дисплеи, т. к. автоматически-созданный по умолчанию файл конфигурации содержит только один раздел для display..

Для каждого дисплея можно изменить свойства графики прямо в файле конфигурации, или использовать phgrafx для конфигурации каждого дисплея.

Более подробная информация о формате файла конфигурации io-display и возможных опциях содержится в описании io-display в справке по утилитам.

16. Настройка встраиваемого Web-сервера

ЗОСРВ «Нейтрино» поставляется вместе с небольшим Web-сервером Slinger, оптимизированным для работы со встраиваемыми приложениями. Поскольку этот Web-сервер поддерживает общий шлюзовый интерфейс (Common Gateway Interface, CGI) версии 1.1, стандарт HTTP 1.1 и динамический HTML (через команды SSI), то во встраиваемые приложения можно просто добавлять встраиваемые услуги HTTP и динамическое содержание.

Например, можно написать приложение, которое управляет принтером и использует сервер Slinger для обновления удаленного клиента, где отображаются данные о текущем состоянии принтера (рис. 16.1).

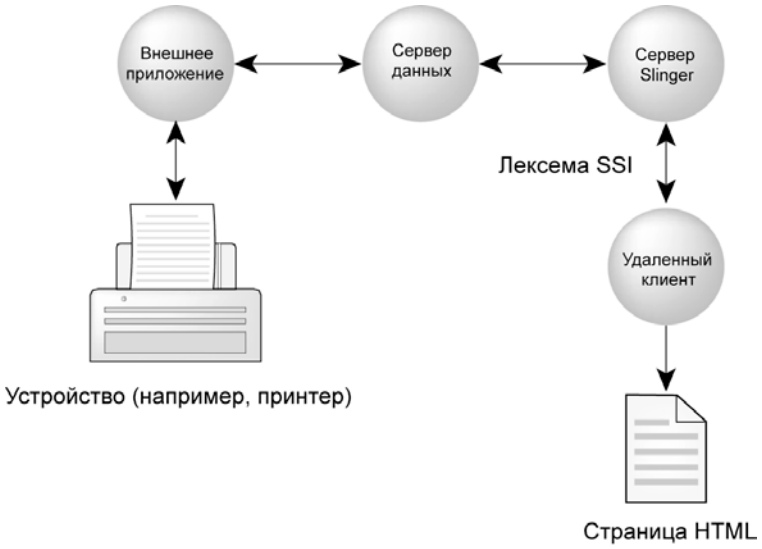


Рис. 16.1. Обновление удаленного клиента

16.1. Где следует размещать файлы?

Перед запуском Web-сервера Slinger и началом создания Web-страниц необходимо определить подходящую структуру каталогов и места для размещения файлов.

Предупреждение. Обращайте особое внимание на место размещения файлов. Каталог для них (или каталоги) не должен быть доступен извне, иначе ваша система может подвергаться необоснованному риску. Например, не размещайте CGI-сценарии в том же каталоге, где находятся системные бинарные

файлы, потому что тогда другие пользователи будут иметь возможность запустить любую команду на машине, которая поддерживает ваш Web-сервер.

Для конфигурирования Web-сервера Slinger используйте следующие переменные окружения:

HTTPD_ROOT_DIR — имя каталога, где предполагается размещение файлов данных для Web-сервера Slinger. Имя каталога по умолчанию: /usr/local/httpd;

HTTPD_ROOT_DOC — имя основного (корневого) HTML-документа. Когда Web-клиент запрашивает основной документ, то значение **HTTPD_ROOT_DOC** присоединяется к значению **HTTPD_ROOT_DIR**, в результате чего формируется полный путь к основному документу. По умолчанию основному документу назначается имя index.html.

Например, если в качестве значения **HTTPD_ROOT_DOC** определено имя index.html, а в качестве значения **HTTPD_ROOT_DIR** определен каталог /usr/www, то в качестве полного пути в Web-сервере Slinger будет использоваться значение /usr/www/index.html.

После принятия решения о структуре каталога до запуска сервера Slinger нужно экспортировать указанные выше переменные окружения:

```
export HTTPD_ROOT_DIR=/usr/local/httpd
export HTTPD_ROOT_DOC=index.html
```

Подробности об установке переменных окружения см. в разделе 9.

16.2. Запуск Web-сервера Slinger

Для запуска Web-сервера Slinger нужно просто ввести команду:

```
slinger &
```

Примечание. Web-сервер Slinger работает поверх сокетов TCP. Поэтому должны быть запущены программы поддержки работы с сокетами, что означает

необходимость запуска стека протокола TCP/IP. Подробности об этом см. в разделе 13.

Сервер Slinger осуществляет прослушивание порта 80 протокола TCP. Поскольку номер этого порта меньше значения 1024, то запуск сервера Slinger должен осуществляться от имени пользователя с учетной записью root. Сразу после присоединения к порту 80 сервер изменяет свой идентификатор пользователя на –2, используя для этого вызов (setuid (-2)).

Во многих встраиваемых серверах при добавлении страниц пользователю приходится повторно соединяться с сервером, что является угрозой надежности, потому что программные коды поставщика и пользователя оказываются в разделяемом пространстве памяти. Несмотря на свой размер, сервер Slinger предоставляет достаточно функциональных возможностей для поддержки доступа к динамически сгенерированным HTML-страницам через сценарии CGI или механизм SSI.

16.3. Динамический HTML

Встраиваемый Web-сервер предоставляет несколько путей создания динамических HTML-страниц:

- через сценарии CGI;
- через механизм SSI;
- с использованием сервера данных.

Метод CGI

Встраиваемый Web-сервер поддерживает интерфейс CGI (Common Gateway Interface) версии 1.1, который представляет собой удобное средство для управления динамическими данными. Недостатком CGI является то, что он ресурсоемок из-за использования фрагментов на языке интерпретируемого типа.

Если вы используете интерфейс CGI, то нужно решить, где располагать каталог cgi-bin, в котором сохраняются CGI-сценарии.

Для настройки Web-сервера на использование интерфейса CGI необходимо применять переменную окружения **HTTPD_SCRIPTALIAS**, в которой указывается место расположения исполняемых файлов и CGI-сценариев, например:

```
export HTTPD_SCRIPTALIAS=/usr/www/cgi-bin
```

После определения переменной **HTTPD_SCRIPTALIAS** станет возможно запускать сценарии или процессы, которые размещаются в этом каталоге вашей машины. Поэтому убедитесь в том, что вы создали отдельный каталог для размещения сценариев. Если не будет определена переменная окружения **HTTPD_SCRIPTALIAS**, все функциональные возможности по работе со сценариями CGI будут отключены, а все запросы на исполнения этих сценариев будут приводить к ошибке.

> **Предупреждение**. Не используйте для каталога CGI-сценариев имена /bin или /usr/bin. Не размещайте в каталоге cgi-bin никаких значимых файлов, потому что они не будут защищены от воздействия со стороны того, кто использует Web-сервер.

Проверьте, чтобы файлы в каталоге cgi-bin могли бы исполняться любым пользователем, но их изменение должно быть доступно только пользователям с учетной записью root. Поэтому для файлов этого каталога должен быть установлен код прав доступа 755 (нужно выполнить команду chmod 755).

Например, пусть для каталога в переменной окружения **HTTPD_SCRIPTALIAS** определено имя /usr/www/cgi-bin. Если сервер Slinger получит запрос на ресурс **www.site.ru/cgi-bin/get_data.cgi/foo**, то будет выполнен найденный в каталоге /usr/www/cgi-bin сценарий get_data.cgi, а имя foo будет передано сценарию get_data.cgi в качестве пути. Имя каталога foo сохраняется в переменной окружения **PATH_INFO**, и оно используется для передачи дополнительной информации о пути.

У сервера Slinger существует несколько переменных окружения, которые используются в CGI-сценариях. Более подробно об этом см. в описании программы

slinger в руководстве "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01.

Метод SSI

SSI (Server Side Includes, вставки на стороне сервера) представляет собой разновидность командного языка, элементы которого могут встраиваться в HTML-файлы. С помощью SSI можно добавлять к HTML-странице динамическое содержание. Для передачи информации SSI-команде exec в сервере Slinger используются переменные окружения **PATH** и **CMD_INT**. Просматривая динамические страницы HTML, клиенты могут использовать на своих Web-страницах в реальном времени интерактивные функции.

Клиенты могут создавать динамические HTML-страницы, помещая на них внутри кода HTML лексемы языка SSI. Лексема языка SSI включает в себя команду, управление которой осуществляет сервер Slinger. При передаче HTML-кода сервер Slinger заменяет лексему HTML-данными, основанными на теге, содержащемся в лексеме SSI.

Например, встраиваемый сервер может:

- исполнять утилиты в указываемой пользователем точке HTML-документа (в качестве дополнительной возможности выходные результаты этих утилит могут быть включены в состав документа);
- вставлять в указанную пользователем точку содержимое других HTML-файлов;
- исполнять условные операторы (например, if, break, goto), чтобы определить, какие части HTML-документа должны передаваться.

Для того чтобы лексемы языка SSI обрабатывались сервером Slinger, HTML-файл должен иметь расширение .shtml.

Теги языка SSI можно использовать для взаимодействия с сервером данных.

Синтаксис команд SSI

Далее приводится несколько примеров команд SSI, которые вы можете использовать в своих сценариях.

Отображение времени и даты:

<!-- #echo var="DATE_LOCAL" -->

Отображение времени и даты по Гринвичу:

<!-- #echo var="DATE_GMT" -->

Отображение IP-адреса посетителя:

<!-- #echo var="REMOTE_ADDR" -->

Отображение информации о браузере посетителя:

<!-- #echo var="HTTP_USER_AGENT" -->

Отображение даты последнего изменения страницы:

<!-- #config timefmt = "%A %B %d, %y" --> Последнее изменение этого файла <!-- #echo vars="LAST_MODIFIED" -->

Включить в данном месте HTML-документа содержимое файла myfile.shtml:

<!-- #include virtual = "myfile.shtml" -->

Выполнить CGI-сценарий counter.pl и поместить выходные результаты его работы на Web-страницу:

<!-- #exec cgi = "counter.pl" -->

Отобразить на Web-странице содержимое каталога /tmp:

<!-- #config cmdecho = "on" --><!-- #exec cmd = "cd /tmp; ls" -->

Метод с использованием сервера данных

Управлять динамическими HTML-страницами можно также с использованием сервера данных (процесс ds). Сервер данных позволяет использовать многопоточные разделяемые данные, независимо от границ процессов. Поскольку встраиваемый Web-сервер поддерживает язык SSI, мы решили расширить его возможности, добавив взаимодействие с сервером данных.

Теперь можно запустить процесс, обновляющий сервер данных, касающихся состояния аппаратного устройства, в то время как встраиваемый Web-сервер получает независимый, но надежный доступ к информации об этом состоянии.

Более подробно о процессе сервера данных см. в описании программы ds в руководстве "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01. Там же приведен пример приложения, осуществляющего мониторинг устройства.

16.4. Меры обеспечения безопасности

Когда вы выбираете каталог для хранения файлов данных, мы советуем вам соблюдать следующие рекомендации:

- не размещайте в каталоге для документов значимые для работы файлы;
- изолируйте каталог с вашими файлами данных от каталога с системными файлами. Например, использовать каталог /usr/www гораздо безопаснее, чем корневой каталог /. Корневой каталог / открывает для обслуживания сервером Slinger всю вашу систему.

При конфигурировании сервера Slinger для поддержки работы с CGI-сценариями придерживайтесь таких рекомендаций:

- размещайте CGI-сценарии в каталоге, изолированном от каталога с обычными системными исполняемыми файлами. Не используйте для этих целей каталоги /bin или /usr/bin;
- если владельцем файла является привилегированный пользователь (например, имеющий учетную запись root), то не допускайте установку для CGI-сценария флага SUID;
- храните файлы CGI-сценариев и файлы документов в разных каталогах. Это предотвратит от получения несанкционированного доступа к файлам сценариев.

Не подвергайте необоснованному риску вашу машину. Проверьте, чтобы:

- все файлы и каталоги имели атрибут "только чтение";

- не было файлов, владельцем которого был бы пользователь с ID (-2), потому что под этим идентификатором запускается сервер Slinger, а следовательно, он станет владельцем таких файлов.

Данные рекомендации помогут избежать ситуации несанкционированной замены файла с паролем или Web-страниц.

Более подробно об этом см. в разделе 19.

16.5. Примеры

Конфигурация

Мы порекомендовали размещать файлы ваших документов и сценариев в разных каталогах. В приводимом далее примере для документов отводится каталог /usr/local/httpd, файл index.html размещается в корневом каталоге, а для CGI-сценариев предназначается каталог /usr/www/cgi-bin.

```
export HTTPD_ROOT_DIR=/usr/local/httpd
export HTTPD_ROOT_DOC=index.html
export HTTPD_SCRIPTALIAS=/usr/www/cgi-bin
slinger &
```

В следующем примере показан неправильный способ задания конфигурации сервера Slinger. При такой конфигурации возможна загрузка файлов со сценариями сторонним пользователем, поскольку документы и сценарии оказываются в одном и том же каталоге:

```
export HTTPD_ROOT_DIR=/usr/www
export HTTPD_ROOT_DOC=index.html
export HTTPD_SCRIPTALIAS=/usr/www
slinger &
```

Для изменения конфигурации сервера Slinger таким образом, чтобы была возможность работать с командами SSI и проводить отладку, используйте следующие команды:

```
export HTTPD_ROOT_DIR=/usr/local/httpd
```

```
export HTTPD_ROOT_DOC=index.shtml
export HTTPD_SCRIPTALIAS=/usr/www/cgi-bin
slinger -des&
```

Сценарий

Далее приводятся два примера простого CGI-сценария, с помощью которого на Web-страницу выводится случайно выбираемое изображение. Один и тот же сценарий представлен на языках C и Perl, поэтому можно увидеть, как реализовать такую процедуру на любом из этих языков.

Исполняемую программу на C (rand_images.cgi) и сценарий на языке Perl (rand_images.pl) нужно поместить в каталог /usr/www/cgi-bin. Используйте команду chmod, чтобы проверить установку кода прав доступа 755 для обоих файлов.

Выбираемые изображения размещаются в каталоге /usr/local/httpd/images. Доступ к изображению со стороны Web-страницы происходит из локального каталога, в CGI-сценарии просто вычисляется, какое изображение нужно загрузить.

Для запуска сценария из Web-страницы используется следующий HTML-код с командами SSI:

```
<H2>Случайное изображение</H2>
<P>
Perl Script: <!--#exec cgi="rand_images.pl" --><BR>
C Program: <!--#exec cgi="rand_images.cgi" --><BR>
```

Программа rand_images.c

Для компиляции этого приложения нужно выполнить команду:

```
cc -o rand_images.cgi rand_images.c
```

Листинг программы:

```
/*  Данная программа генерирует случайное число и затем выбирает
    изображение в зависимости от этого числа. В результате
    при каждой загрузке Web-страницы изображение меняется.
*/

    #include <stdio.h>
    #include <stdlib.h>
```

```
#include <time.h>

/*  установка переменных */

char *dir = "/images/";

char *files[] ={"file1.jpg", "file2.jpg",
                "file3.jpg", "file4.jpg",
                "file5.jpg"};

int num;
int size;

int main()

{
  size = sizeof (files) / sizeof (files[0]);
  srand( (int)time(NULL) );
  num = ( rand() % 4 );

  /*  Печать заголовка со случайным именем файла и
      базовым каталогом */

  printf("<img src=\"%s%s\" alt=%s border=1 >\n<BR>",
         dir, files[num], files[num]);
  printf("Location: %s%s\n\n<BR>",dir, files[num]);
  return (0);
}
```

Программа rand_images.pi

```
#!/usr/bin/perl

# Данный сценарий генерирует случайное число, а
# затем выбирает изображение в зависимости от этого числа.
# В результате при каждой загрузке Web-страницы
# изображение меняется.

# установка переменных
$dir = "/images/";
@files = ("file1.jpg", "file2.jpg", "file3.jpg",
          "file4.jpg", "file5.jpg");
```

```
srand(time ^ $$);
$num = rand(@files); # Pick a Random Number

# Печать заголовка со случайным именем файла и
# базовым каталогом

print "<img src=\"$dir$files[$num]\"
alt=$files[$num] border=1 >\n<BR>";
print "Location: $dir$files[$num]\n\n<BR>";
```

17. Использование CVS

17.1. Сверхкраткий курс по CVS

Система управления версиями CVS (Concurrent Versions System) является инструментом с открытым исходным кодом, предназначенным для управления версиями файлов. Под контроль системы CVS можно поставить любые типы файлов, но в данном разделе основное внимание будет сконцентрировано на файлах исходных текстов программ и других текстовых файлах.

Под управлением версиями (version control) понимается возможность отслеживания изменений в файлах. Каждый раз, когда файл изменяется, в системе регистрируется дата изменения, описание изменения и пользователь, который изменил файл. Это позволяет отслеживать когда, кем и почему был изменен файл. Система CVS позволяет также координировать изменения, выполняемые для одного и того же файла многими пользователями.

Использование системы CVS для контроля за версиями файлов исходных текстов дает возможность выделять, какие изменения в файлах станут частью очередной версии программного обеспечения, а какие — нет. Это означает, что можно выпустить текущую версию проекта, продолжая, в то же время, работу над будущей версией с новыми функциями. Именно возможность такого параллелизма при контроле версий программного обеспечения сделали систему CVS столь популярной.

Мы будем знакомиться с основами использования системы CVS с начальной настройки ее параметров для управления файлами с исходными текстами. Позже будут затронуты более глубокие вопросы, такие как параллельная разработка и удаленный доступ.

Более подробную информацию о системе CVS, включая полное руководство пользователя CVS ("CVS User's Guide"), можно найти на сайте **http://www.cvshome.org**.

17.2. Основы CVS

В системе CVS файлы сохраняются в едином месте, называемом репозиторием (repository). Репозиторий размещается на диске вашей локальной машины или на удаленном сервере. В данном разделе предполагается локальное сохранение версий.

Версии

Каждый раз, когда в хранимый в CVS файл вносятся изменения, создается новая версия (revision) файла. В версию файла включается дата изменения, имя пользователя, сделавшего изменение, и журнальное сообщение, описывающее суть изменения. В любое время можно извлечь для изучения произвольные версии файла. Отдельным версиям можно присваивать символические имена, метки (tags), для упрощения ссылок на них.

Версия обозначается как последовательность чисел и точек. Этот метод аналогичен стандартной схеме нумерации версий программного обозначения. Например, в файле foo.c за последние несколько дней было сделано три изменения. Первая редакция файла получает номер 1.1, вторая — 1.2, а третья — 1.3. Номера в системе CVS назначаются автоматически и используются только внутри системы. Вам придется использовать эти номера в разных ситуациях.

Изменения в файле foo.c носят кумулятивный характер, поэтому в версии 1.3 содержатся как все изменения, внесенные между версиями 1.1 и 1.2, а также все изменения, внесенные между версиями 1.2 и 1.3.

Основные операции

Как в системе CVS становится известно о внесенных в файл изменениях? Создается ли новая версия каждый раз, когда файл сохраняется?

Фактически вы не манипулируете файлами непосредственно в репозитории. Вместо этого вы создаете у себя на диске копию репозитория. Изменения вносятся в копии файлов, а когда вы будете удовлетворены внесенными изменениями, то системе CVS дается команда на перенос измененного файла в репозиторий и о

создании новой версии. Этот процесс называется обновлением версии (checking in). В момент обновления вы вводите комментарий о сделанных изменениях.

Как создать у себя локальную копию репозитория? Эта операция называется выгрузкой версии (checking out) и является обратной операции обновления. При выполнении операции выгрузки создается копия репозитория с информацией о ее текущем состоянии. Обычно требуется получить копию текущего состояния репозитория, но иногда необходим более детальный контроль над версиями файлов. Поэтому для этой операции предусмотрено множество ключей, в том числе возможность использования символических имен и явных дат.

Репозиторий

Для обновления и выгрузки файлов вначале нужно создать репозиторий. Для вновь открываемых проектов вы создаете новые файлы и добавляете их в репозиторий по мере появления. Для существующих проектов, когда отсутствовал контроль изменения версий, можно импортировать в репозиторий одной командой весь проект сразу.

Для всех операций в первую очередь нужно знать место нахождения репозитория. Для него нет местоположения по умолчанию. Репозиторий — это просто каталог, имя которого можно указать через командную строку или задать значение для некоторой переменной окружения.

Редакторы и система CVS

В системе CVS часто выдается запрос на получение некоторой информации, при этом запускается редактор с определенным шаблоном. Можно управлять этим процессом, указав, какую именно программу редактора нужно запускать. Это делается путем установки значения переменной окружения **EDITOR**. Например, для использования редактора оболочки Photon ped нужно использовать такую команду для помещения этой информации в профиль .profile:

```
export EDITOR=ped
```

Сведения о доступных редакторах приведены в разделе 7. Информация о профиле .profile приведена в разделе 9.

Создание репозитория

Вначале нужно решить, где будет размещаться репозиторий. Например, для него выбран каталог $HOME/cvs.

Для создания пустого репозитория введите следующую команду:

```
cvs -d$HOME/cvs init
```

Если заглянуть в каталог $HOME/cvs, то там можно увидеть созданный каталог CVSROOT. В нем находятся внутренние административные файлы, необходимые для работы системы CVS.

Ключ -d в команде cvs используется в системе CVS для указания на место расположения репозитория. Исполняемая операция init осуществляет создание нового репозитория. Ключ -d является глобальным, поскольку он появляется перед исполняемой операцией init. В общем случае вызов команды cvs имеет следующий формат:

```
cvs [глобальные_ключи] команда [ключи_команды] имена_файлов
```

После того как был создан репозиторий, в каталоге CVSROOT нужно отредактировать указанные далее файлы:

- readers – список пользователей, которые могут читать файлы из репозитория;
- writers – список пользователей, которые могут читать файлы из репозитория и записывать их туда.

Один и тот же пользователь не может присутствовать в обоих файлах.

Загрузка файлов в репозиторий и выгрузка из него

Есть два способа помещения в репозиторий исходных файлов: добавление новых файлов или импортирование существующего дерева каталога. Сначала рассмотрим вариант с созданием новых файлов.

Поскольку работа начинается с новым репозиторием, то нужно создать локальную рабочую копию. Но в каталоге пока ничего нет.

Сначала можно выполнить выгрузку упомянутого выше каталога CVSROOT — кроме него ничего другого в репозитории нет — как это можно сделать с любым каталогом. Необходимо также создать каталог для помещения локальной копии, назовем его "песочницей" (sandbox) и разместим его в своем домашнем (home) каталоге:

```
cd $HOME
mkdir sandbox
cd sandbox
```

Теперь необходимо поместить в этот каталог рабочую копию из репозитория:

```
cvs -d$HOME/cvs checkout .
```

или

```
cvs -d$HOME/cvs get .
```

Использование точки (.) в качестве имени файла интерпретируется как "получить содержимое всего репозитория".

В каждом выгруженном каталоге вы обнаружите подкаталог с именем CVS. Этот каталог используется системой CVS для сохранения информации о том, в каком месте репозитория размещены файлы, каковы версии этих файлов и т. д. Не вносите никакие изменения в файлы этого каталога.

Предупреждение.

Если вы создаете новый проект, копируя каталоги из одной части "песочницы" в другую, не копируйте подкаталог CVS. Если вы сделаете это, то файлы могут сохраниться не в том месте репозитория, в котором вы ожидаете.

Теперь, когда у нас имеется рабочая копия, можно создать другие каталоги и файлы, чтобы продемонстрировать выполнение операций обновления и выгрузки. Начнем с написания стандартной программы "Hello, world" на языке C. Традиционно рекомендуется сохранять все проекты в отдельной структуре каталогов, поэтому мы также создадим для проекта новый каталог.

Для этого нужно выполнить команду:

```
mkdir myproj
```

Теперь добавим этот каталог в репозиторий системы CVS:

```
cvs -d$HOME/cvs add myproj
```

Наконец, пришло время создать тестовый файл, но вначале нужно оказаться в каталоге проекта:

```
cd myproj
```

Теперь вызывайте любой текстовый редактор, чтобы создать файл foo.c со следующим содержимым:

```
#include <stdio.h>

int main (int argc, char *argv[]) {
  printf ("Hello, world.\n");

}
```

Добавление файла в репозиторий производится так же, как и добавление каталога:

```
cvs add foo.c
```

Обратите внимание на то, что мы не используем в команде cvs ключ -d. Это сделано специально. Когда вы выгружаете каталог, система CVS создает свой специальный каталог для хранения административных файлов и файлов статуса, поэтому в системе известно, с каким каталогом репозитория ведется работа. Все последующие операции подразумевают работу именно с этой частью репозитория.

Данная команда системы CVS означает, что вы хотите добавить файл. Реально он еще не добавлен. Просто системе явно сообщается, что вы закончили внесение изменений в локальную копию репозитория. Таким образом вы можете изменять или добавлять несколько файлов и каталогов в любое время, а затем, после завершения работы, сообщить системе CVS принять все эти изменения. Для этого используется операция commit.

Перенос изменений в репозиторий

Операции commit или ci (от англ. check in — обновление) приводят содержимое репозитория в соответствие с содержимым вашей локальной копии. Если с общим репозиторием работает несколько человек, все происходит немного по-другому, но в данном случае мы предполагаем, что репозиторием пользуется только один человек:

```
cvs commit foo.c
```

или

```
cvs ci foo.c
```

После ввода команды система CVS запускает текстовый редактор, чтобы вы ввели описание файла. Введите какую-нибудь полезную информацию, например: "Файл для проверки системы CVS". Текст может вводиться в свободной форме, так что можете добавить любое сообщение. После завершения ввода текста сохраните его и выйдите из редактора. После этого от системы CVS поступит сообщение о том, что операция совершена.

Импорт существующего дерева файлов с исходными текстами

Вероятно, вы уже обратили внимание на то, что добавление в систему CVS файлов из существующего дерева каталогов с использованием описанных выше команд add и commit становится трудоемким, если файлов более двух. В этом случае можно воспользоваться операцией import. В данном разделе мы коснемся наиболее общих принципов использования этой операции. Позднее будут рассмотрены некоторые дополнительные возможности.

При использовании операции import предполагается, что где-то на диске имеется дерево каталогов. Эти каталоги не должны находиться ни внутри репозитория, ни внутри локальной копии репозитория. Для добавления в CVS-репозиторий всего дерева каталогов используются команды следующего формата:

cd добавляемый_источник

cvs -dпуть_к_репозиторию import путь_в_репозитории метка_ветки_поставщика метка_версии

После этого в открывшемся окне редактора можно добавить комментарий.

На первый взгляд такой формат команды может показаться несколько странным, но операция import выполняет не только простое импортирование. Эта команда всегда импортирует содержимое текущего рабочего каталога. Параметр путь_в_репозитории означает, что система CVS должна создать этот путь внутри репозитория и поместить туда содержимое текущего каталога.

Последние два аргумента — метка_ветки_поставщика (vendor) и метка_версии (init) — служат для создания так называемой "ветки" (branch) и метки для импортируемых файлов (см. подразд. "Параллельная разработка: ветвление и слияние" далее в этом разделе). Эти параметры не нужны при импортировании собственного программного обеспечения, но в соответствии с форматом их все равно требуется указать для системы CVS.

Получение информации о файлах

Текущий статус файла можно получить с помощью команд status или stat:

```
cvs status foo.c
```

Результат работы команды будет выглядеть примерно так:

```
===================================================================
File: foo.c             Status: Up-to-date

   Working revision:    1.1     Tue Jun 3 17:14:55 2003
   Repository revision: 1.1     /home/fred/cvs/myproj/foo.c,v
   Sticky Tag:          (none)
   Sticky Date:         (none)
   Sticky Options:      (none)
```

Изменение файлов

При создании файла foo.c мы не вставили в него ни одной строки комментария! Это нужно исправить. С помощью текстового редактора добавьте в самом начале файла foo.c строку:

/* Это файл для проверки системы CVS */

Теперь давайте посмотрим на статус файла:

```
===================================================================
File: foo.c             Status: Locally Modified

   Working revision:    1.1     Tue Jun 3 17:14:55 2003
   Repository revision: 1.1     /home/fred/cvs/myproj/foo.c,v
   Sticky Tag:          (none)
   Sticky Date:         (none)
   Sticky Options:      (none)
```

Статус файла изменился на Locally Modified (локально изменен). Это сигнал о том, что вы изменили файл, но должны еще сообщить об этом системе CVS. Это можно сделать так:

```
cvs commit foo.c
```

Как и раньше, после запуска команды открывается окно редактора для ввода сообщения (комментария). Здесь можно написать о причине изменений или кратко описать их характер. Опять же текст может быть в свободной форме, поэтому напишите любое сообщение. Мы, например, напишем: Добавили комментарий для ясности. Сохраните файл и закройте редактор.

Опросим статус файл. Теперь мы получим такой результат:

```
===================================================================
File: foo.c             Status: Up-to-date

   Working revision:    1.2     Tue Jun 3 17:30:49 2003
   Repository revision: 1.2     /home/fred/cvs/myproj/foo.c,v
   Sticky Tag:          (none)
   Sticky Date:         (none)
   Sticky Options:      (none)
```

Дополнительная информация о файлах: что изменилось и почему?

В показанном выше результате о статусе файла номер его версии (revision) изменился с 1.1 на 1.2. У нас теперь есть две версии файла foo.c, поэтому можно посмотреть, что изменилось и почему были сделаны изменения. Для получения ответа на вопрос "почему" нужно просмотреть сообщения журнала, куда заносятся записи каждый раз при совершении операции по внесению изменений. Содержимое журнала выводится по команде:

```
cvs log foo.c
```

Результат вывода будет таким:

```
RCS file: /home/fred/cvs/myproj/foo.c,v
Working file: foo.c
head: 1.2
branch:
locks: strict
access list:
keyword substitution: kv
total revisions: 2;     selected revisions: 2
description:
----------------------------
revision 1.2
date: 2003/06/03 17:35:43;  author: fred; state: Exp; lines: +2 -
0
Added comments for clarity.
----------------------------
revision 1.1
date: 2003/06/03 17:19:34; author: fred; state: Exp;
A file to test the basic functionality of CVS
=============================================================
==
```

Для того чтобы увидеть различия между двумя версиями, используем команду diff:

```
cvs diff -r1.1 foo.c
```

Результат вывода будет таким:

```
Index: foo.c
```

```
    ===================================================================
==
    RCS file: /home/fred/cvs/myproj/foo.c,v
    retrieving revision 1.1
    retrieving revision 1.2
    diff -r1.1 -r1.2
    0a1,2
    >/* Это файл для проверки системы CVS */
    >
```

Последние строки, начиная с diff -r1.1 -r1.2z, показывают фактические различия, используя стандартный формат команды diff (см. руководство "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01).

Возможно, вы уже обратили внимание на то, что для команды diff с помощью ключа -r был указан номер только одной версии. В системе CVS подразумевается, что второй версии соответствует файл foo.c, находящийся в вашей "песочнице". Из последнего результата вывода статуса по команде status видно, что номер рабочей версии был 1.2, следовательно, это вторая версия. Мы могли бы явно указать номер версии, используя второй ключ -r:

```
    cvs diff -r1.1 -r1.2 foo.c
    Index: foo.c
    ===================================================================
==
    RCS file: /home/fred/cvs/myproj/foo.c,v
    retrieving revision 1.1
    retrieving revision 1.2
    diff -r1.1 -r1.2
    0a1,2
    > /* Это файл для проверки системы CVS */
    >
```

Как видно, получены аналогичные результаты.

17.1. Система CVS и деревья каталогов

В системе CVS дерево каталогов просматривается автоматически, начиная с текущего рабочего каталога (если не задано конкретное имя файла или имя каталога). Например, после выполнения команды:

```
cvs stat
```

выдается статус всех файлов текущего каталога и всех нижележащих подкаталогов.

Такая возможность весьма удобна для внесения изменений в различные части дерева по прошествии некоторого времени после изменения отдельных файлов. Для регистрации всего множества изменений сразу нужно просто перейти в корневой каталог и ввести команду:

```
cvs commit
```

Вы получите запрос на ввод только одного сообщения для журнала. Это сообщение будет распространяться на все изменения, выполненные этой командой.

17.2. Параллельная разработка: ветвление и слияние

Иногда возникает необходимость работы над более чем одной версией файла. Например, вам может потребоваться исправление ошибки в текущей версии программы, в то время как ведется работа над новыми функциональными возможностями будущей версии. В системе CVS это делается просто путем ветвления (branch) файлов.

Ветвление

Когда вы создаете ветку, в системе CVS создается другая копия файла или файлов, причем редактировать можно любую версию. В системе CVS отслеживается, к какой версии относятся соответствующие изменения.

Основная ветка (branch) разработки называется головной (head). Можно принять решение о разработке новых функциональных возможностей в головной ветке, а можно создать отдельные ветки для других версий программного обеспечения (рис. 17.1).

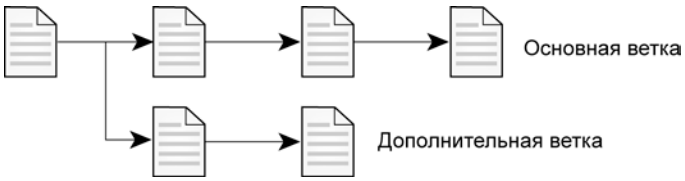


Рис. 17.1. Ветвление файла в системе CVS

Допустим, выпускается версия 1.0 нового продукта, названного Stella, и в состав этого продукта входит файл foo.c. Можно создать ветку для этой версии следующим образом:

```
cvs tag -b "Stella_1.0" foo.c
```

Stella_1.0 является меткой (sticky tag). Любые изменения, которые вы делаете в вашей "песочнице", связываются с веткой Stella_1.0, а не с основной веткой. Если вы хотите работать с основной веткой, то нужно обновить "песочницу", указав ключ -A, который отключит работу с метками:

```
cvs update -A
```

или

```
cvs up -A
```

Что нужно сделать, если вы хотите выгрузить обе версии? Можно обновлять содержимое "песочницы", используя отдельную команду для головной ветки (как показано ранее) и отдельную для рабочей ветки (cvs update -r Stella_1.0), однако в этом случае трудно отслеживать, над какой именно версией вы сейчас работаете. Вместо этого вы можете выгрузить ветку в другой каталог:

```
cd ~/cvs
mkdir version1.0
cd version1.0
cvs checkout -r Stella_1.0 путь_к_файлам
```

Слияние

Итак, вы сделали изменения в одной ветке, и нужно сделать изменения в другой ветке. Можно отредактировать файлы дважды, но это не очень удобно. Вместо этого в системе CVS можно выполнить слияние одной ветки с другой.

Примечание. Обычно проще выполнить слияние дополнительной ветки с основной.

Для слияния изменений в файле foo.c ветки Stella_1.0 с версией головной ветки перейдите в каталог, откуда была выгружена основная ветка в вашу "песочницу", и введите команду:

```
cvs update -j Stella_1.0 foo.c
```

Примечание. Рекомендуется проверить файл на корректность выполнения операции слияния. Никогда не полагайтесь полностью на действия машины.

Иногда изменения, сделанные в одной ветке, конфликтуют с изменениями, сделанными в другой ветке. Если такое случается, то при слиянии версий перед именем файла выводится символ C. В системе CVS конфликтующие строки остаются без изменения, но помечаются знаками "больше чем", "равно", "меньше чем". Вам нужно отредактировать файлы для устранения несоответствий, а затем повторно обновить версии в системе CVS.

17.3. Удаление и восстановление файлов

Когда файл удаляется из репозитория, система CVS перемещает его на "чердак" (attic). У каждого каталога в репозитории имеется подкаталог Attic. Этот каталог нельзя выгрузить в вашу "песочницу", но содержимое каталога можно исследовать через Web-интерфейс в системе CVS.

Для удаления файла (например, phoenix.c) необходимо выполнить такую последовательность действий.

- удалите файл из "песочницы":

```
rm phoenix.c
```

- удалите файл из репозитория:

```
cvs remove phoenix.c
```

или

```
cvs rm phoenix.c
```

- выполните команду применения изменений (commit).

Если нужно восстановить файл, то выполните следующие действия.

- определите номер последней версии файла:

```
cvs log phoenix.c | less
```

- перейдите в базовый каталог вашей "песочницы" (например, ~/cvs) и выгрузите файл с номером версии на единицу меньшим последней версии. Например, если удаленная версия файла phoenix.c была 1.4, то вам нужна версия 1.3. Для этого введите команду:

```
cvs checkout -r 1.3 my_project/phoenix.c
```

или

```
cvs get -r 1.3 my_project/phoenix.c
```

Ключ -r устанавливает метку.

- возвратитесь в каталог, в который вы хотите поместить файл, переименуйте его или переместите в другое место, после чего снимите метку:

```
mv phoenix.c save_phoenix.c
cvs update -A
```

- переименуйте или переместите файл обратно, а затем добавьте его в репозиторий:

```
mv save_phoenix.c phoenix.c
cvs add phoenix.c
```

- выполните команду применения изменений (commit).

17.4. Настройка сервера CVS

Настройка сервера CVS похожа на настройку локального репозитория (см. подразд. "Создание репозитория" ранее в этом разделе). Однако нужно выполнить три дополнительные операции.

- проверьте, чтобы в файле /etc/services присутствовала строка вида:

```
pserver 2401/tcp
```

- проверьте, чтобы в файле /etc/inetd.conf присутствовала запись вида (все в одной строке):

`pserver stream tcp nowait root /usr/bin/cvs cvs -b /usr/local/bin –f--allow-root=`корневой_каталог `pserver`

Здесь корневой_каталог – это путь, который вы хотите использовать в качестве корневого каталога системы CVS. Общепринято, чтобы путь заканчивался именем CVSRoot, хотя это не обязательное правило.

Можно иметь более одного корневого каталога. Нужно просто добавить несколько экземпляров параметра --allow-root=корневой_каталог.

- выполните команду init:

`cvs -d` корневой_каталог `init`

После этого будет создан каталог с именем корневой_каталог, и в него будут занесены все необходимые для работы репозитория файлы.

Более подробная информация содержится на сайте **http://www.cvshome.org**.

18. Резервное копирование и восстановление данных

18.1. Общие сведения

Даже если у вас имеется надежное оборудование и источник электропитания, даже если вы уверены в том, что никогда случайно не сотрете свои рабочие файлы, все равно имеет смысл выполнять резервное копирование файлов. Стратегии резервного копирования различаются в зависимости от простоты их использования, скорости работы, устойчивости к внешним воздействиям и стоимости.

Далее будут отдельно рассматриваться разные типы архивов, тем не менее, в табл. 18.1 мы приведем краткие сведения по расширениям файлов архивов и утилитам, которые могут использоваться для работы с ними.

Таблица 18.1

Расширение файла	**Утилита**
.tar	pax или tar
.cpio	pax или cpio
.gz	gzip или gunzip
.tar.gz или .tgz	tar -z
.z или .F	melt

Не важно, насколько устойчивой к внешним воздействиям является файловая система, в реальной жизни всегда возникают ситуации, когда искажаются данные на диске. Время от времени отказывает оборудование, возникают перебои в работе источника питания и т. д.

Файловая система QNX 4 была построена с учетом возможности возникновения таких катастрофических событий. Основным принципом, положенным в основу структуры файловой системы, является то, что при любых обстоятельствах должна сохраняться целостность файловой системы. В то время как большая часть данных проходит через буферную кэш-память и записывается с небольшой задержкой, запись критических данных файловой системы происходит немедленно. Обновления блоков каталога, блоков индексных дескрипторов (inode blocks), блоков экстентов (extent blocks) и блоков битовой карты (bitmap) сразу же

записываются на диск, чтобы гарантировать сохранность структуры файловой системы (т. е. данные на диске никогда не должны быть внутренне противоречивыми).

Примечание. Файловая система Power-Safe сконструирована таким образом, что ее невозможно испортить; у пользователя всегда есть доступ к текущей версии данных. Дополнительную информацию см. в главе «Файловые системы» руководства "Описание применения. Часть 1. Системная архитектура" КПДА.10964-01 31 01. Некоторые сведения о восстановлении данных действительны только для файловых систем QNX 4.

При возникновении аварийной ситуации можно использовать различные утилиты: fdisk, dinit, chkfsys и spatch. Они помогут обнаружить и восстановить поврежденные файлы, которые были открыты на запись в момент возникновения аварии. Во многих случаях удается полностью восстановить файловую систему.

Иногда характер повреждений может быть более серьезным. Например, вполне возможно, что поврежденный сектор (bad block) окажется в середине файла, записанного на жесткий диск, либо (что еще хуже) в середине каталога или какого-нибудь критического блока данных.

Опять же, поставляемые утилиты помогут определить степень повреждения. Часто можно перестроить файловую систему таким образом, чтобы избежать использования поврежденных областей диска. В таком случае некоторые данные будут потеряны, но путем некоторых усилий можно будет восстановить большую часть поврежденных данных.

18.2. Стратегии резервного копирования

Стратегия резервного копирования состоит в проведении одной или нескольких операций копирования данных. Резервное копирование можно проводить периодически или по отдельной команде запуска процедуры. Для каждой включенной в вашу стратегию процедуры резервного копирования необходимо выбрать:

- среду хранения и место размещения копируемых данных;
- способ архивации и (дополнительно) способ компрессии данных;
- содержимое данных для копирования, частоту или условия запуска процедуры резервного копирования;
- автоматический или ручной способ резервного копирования;
- локальный или удаленный способ управления процедурой копирования.

Очень часто комплексная стратегия резервного копирования включает в себя проведения нескольких сеансов копирования на локальной стороне (т. е. с управлением и размещением архивов на той же машине, где расположены сами данные), а для ряда сеансов копируемые данные сохраняются на удаленной машине. Например, можно ежедневно автоматически копировать данные разработчика на другой раздел жесткого диска, а один раз в неделю автоматически сохранять резервную копию тех же данных на централизованный удаленный сервер.

Выбор носителя и места размещения копируемых данных

На начальном этапе определения стратегии резервного копирования нужно, прежде всего, выбрать место сохранения копируемых данных и носитель, где эти данные будут сохраняться. Эти факторы, в первую очередь, влияют на стоимость оборудования и среды сохранения, связанные с вашей системой.

Для выбора наилучшего варианта определитесь с тем, что нужно копировать и как часто. Эти параметры помогут определить емкость носителя для хранения копируемых данных, скорость передачи по каналу данных и то, как несколько пользователей смогут совместно использовать этот ресурс.

Выбор необходимого носителя для хранения данных зависит от того, создаются ли резервные копии данных на локальной машине или же данные передаются по сети на удаленную машину:

- преимуществом локального размещения копируемых данных является скорость и потенциально более полный контроль за процессом со стороны конечного пользователя, а ограничения касаются технологий резервного

копирования и типов сред, непосредственно поддерживаемых в ЗОСРВ «Нейтрино»;

- резервное копирование на удаленную машину, рассматриваемое в масштабах компании, часто предоставляет больше возможностей по оборудованию и дополнительные возможности по объему и средам хранения данных; недостатки метода обычно связаны с необходимостью передачи данных по сети и с тем, что оборудование совместно используется несколькими пользователями, а это накладывает ограничения на доступ к данным и на процесс их извлечения.

В табл. 18.2 приведен перечень некоторых сред для хранения резервных данных, а также их доступность при локальной или удаленной операции копирования.

Таблица 18.2

Среда	Локально/Нейтрино	Удаленно
Гибкий диск (флоппи)	Да	Да
Устройство LS-120	Да	Да
Магнитная лента	Нет	Да
CD-диски	Да	Да
DVD-диски	Нет	Да
Жесткий диск	Да	Да
Флэш-память	Да	Да
Память большой емкости с интерфейсом USB	Да	Да

Выбор формата резервного копирования

При резервном копировании данных нужно решить, будет ли отдельно копироваться каждый файл и каталог или будет создаваться архив с набором файлов. Также нужно решить, будет ли проводиться операция компрессии данных для уменьшения требований к объему хранилища для резервных копий.

Потеря времени на выполнение компрессии и декомпрессии данных может компенсироваться уменьшением времени при чтении/записи сжатых данных на среду хранения или на передачу по сети. Для уменьшения издержек в процессе выполнения компрессии данных можно задачу компрессии резервной копии

запустить в фоновом режиме. Возможно, эта процедура компрессии будет запускаться через несколько дней или недель для компрессии более поздних резервных копий (это уменьшит занимаемое ими место), в то время как последние резервные копии останутся максимально доступными.

Управление резервным копированием

Операцию резервного копирования нужно производить настолько часто, чтобы восстанавливаемые данные были бы достаточно актуальными или чтобы актуальности можно было бы достичь путем затраты минимальных усилий. Для группы разработчиков программного обеспечения периодичность копирования может составлять от одного дня до недели. Каждый день задержки в копировании, приводящий к необходимости использования устарелой копии, будет стоить лишнего дня на восстановление данных. Если необходимо резервное копирование финансовых данных или данных по продажам, то эта операция обычно выполняется ежедневно или дважды в день. Хорошей практикой является использование хранилища данных на стороне.

18.3. Архивация данных

Резервные копии можно создавать отдельно для каждого файла, а можно сохранять резервные копии нескольких файлов в архиве. Файлы, помещенные в архив, проще идентифицировать по принадлежности к определенному времени создания или компьютеру (например, можно присваивать архиву соответствующее имя). Архив проще передавать целиком в другие системы (передается единственный файл архива), а иногда архив оказывается быстрее сжать по сравнению с компрессией разрозненных файлов.

В ЗОСРВ «Нейтрино» для архива можно использовать несколько форматов, включая pax и tar. В ЗОСРВ «Нейтрино» поддерживается также формат cpio (*.cpio), но мы рекомендуем его выбирать лишь в том случае, когда архив предназначен для других систем, использующих именно формат cpio.

Создание архива

Самым простым способом создания резервной копии в системе является индивидуальное дублирование файлов с помощью команд cp или pax. Например, дублирование единственного файла может быть сделано так:

```
cp -t мой_файл каталог_для_копии
```

или

```
echo мой_файл | pax -rw каталог_для_копии
```

Для резервного копирования целого каталога нужно ввести:

```
cp -Rt мой_каталог каталог_для_копии
```

или

```
find мой_каталог -print | pax -rw каталог_для_копии
```

Для копирования только определенных файлов, соответствующих некоторому критерию, нужно использовать утилиту find или другое средство для идентификации копируемых файлов и перенаправить вывод утилите pax –rw, например, так:

```
find мой_каталог -name '*.[ch]' | pax -rw каталог_для_копии
```

Для объединения отдельных файлов в единый архив можно использовать программы tar или pax. Эти утилиты берут все передаваемые им файлы и помещают их в один большой непрерывный файл. Эти же утилиты могут использоваться для извлечения отдельных файлов из архива.

Примечание. В файловой системе не поддерживаются архивы (или отдельные файлы) размером больше 2 Гбайт.

Когда в качестве архиватора используется утилита pax (режим pax -w), то создаваемые ей архивы имеют формат TAR. Какую из команд использовать, определяется тем синтаксисом командной строки, который вам больше нравится, поскольку создаваемые архивы будут идентичны. Утилита pax была создана как часть стандарта POSIX для обеспечения совместимого механизма для обмена архивами (pax расшифровывается как Portable Archive eXchange — обмен

переносимыми архивами). Это позволило избежать конфликтов, возникающих при работе с различными вариантами программы tar, при которых наблюдалось их разное поведение.

Далее приведены возможные варианты структуры архива.

- одиночные файлы (хотя нет большого смысла создавать такой архив с помощью команд tar и pax). Например:

```
pax -wf my_archive.tar code.c
```

В этой команде берется файл code.c, и из него создается архив (который иногда называется "tarball" — "смоляной шарик") с именем my_archive.tar. Ключ -wf требует от команды pax записи файла.

- несколько файлов. Для архивации нескольких файлов их нужно перечислить в конце командной строки. Например:

```
pax -wf my_archive.tar code.c header.h readme.txt
```

Программа pax создает из всех них один архив с именем my_archive.tar.

- каталоги. Для архивации каталога нужно просто указать его имя в командной строке:

```
pax -wf my_archive.tar workspace
```

По этой команде создается архив с именем my_archive.tar, куда включается все содержимое каталога workspace.

- разделы диска (partitions). Необходимо указать имя каталога раздела в командной строке:

```
pax -wf my_archive.tar /fs/hd0-t79
```

По команде все содержимое раздела t79 архивируется в один очень большой архив с именем my_archive.tar.

Архив можно хранить на своей локальной системе, но мы рекомендуем обязательно делать копию на удаленной системе. Если вдруг произойдет физическое разрушение локальной системы или окажется поврежденным жесткий диск, то локальный архив будет потерян.

Извлечение файлов из архива

Для извлечения файлов из архива можно использовать утилиту pax с ключом -r:

```
pax -rf my_archive.tar
```

Или можно использовать команду tar с ключами -x (от англ. extract — извлечь), -v (от англ. verbose — вывод информации о ходе работы программы) и -f (от англ. filename — имя файла):

```
tar -xvf my_archive.tar
```

Примечание. Чтобы просмотреть содержимое архива без извлечения из него файлов, можно в программе tar вместо ключа -x использовать ключ -t.

Компрессия архива

Архив может иметь достаточно большой размер, особенно если архивируется целый раздел диска. Для уменьшения занимаемого архивом места можно сжать архив, хотя для компрессии при сохранении и для декомпрессии при извлечении файлов требуется некоторое время.

В ЗОСРВ «Нейтрино» имеются следующие программы компрессии/декомпрессии:

- bzip2 и bunzip2;
- freeze и melt;
- gzip и gunzip.

Обычно лучше всего использовать программу gzip, потому что ее поддержка осуществляется во многих операционных системах, тогда как программа freeze используется, главным образом, для обеспечения совместимости с системами на базе ОС QNX 4. Существует также много других программ для компрессии от сторонних производителей.

Например, для сжатия архива my_archive.tar с получением результата в файле my_archive.tar.gz нужно ввести:

```
gzip my_archive.tar
```

Размер сжатого файла будет гораздо меньше исходного, поэтому его легче хранить. Для некоторых утилит, включая gzip, имеется множество ключей, которые позволяют управлять процессом и качеством компрессии. Обычно, чем лучше сжимается файл, тем больше времени требуется на эту процедуру.

Примечание. По умолчанию сжатый файл получает расширение .tar.gz, но может встретиться и расширение .tgz. Для задания этого суффикса нужно при вызове программы gzip использовать ключ -S.

Декомпрессия архива

Для декомпрессии архива используется соответствующая утилита. В случае файлов с расширениями .gz или .tgz необходимо использовать программу gunzip таким образом:

```
gunzip my_archive.tar.gz
```

или

```
gunzip my_archive.tgz
```

В обоих случаях происходит декомпрессия файла в результирующий файл my_archive.tar. Для извлечения файла из архива без предварительной декомпрессии можно также запустить программу tar с ключом -z:

```
tar -xzf my_archive.tgz
```

18.4. Выбор средства для хранения

Сменные носители

В ЗОСРВ «Нейтрино» поддерживаются устройства типа LS-120, магнитооптические дисководы, встроенные ZIP-дисководы и устройства для хранения массивов данных с интерфейсом USB. У каждого из этих устройств есть свои достоинства и недостатки. Поэтому принятие окончательного решения по

использованию для резервного копирования конкретного устройства остается за вами. Инструкции по установке этого оборудования приведены в разделе 15.

Резервное копирование физических жестких дисков

Идентичные образы жестких дисков можно делать с помощью простых утилит, входящих в состав ЗОСРВ «Нейтрино». Эта процедура выполняет полное копирование "сырых данных" (raw copy) с диска.

Если у вас имеются идентичные жесткие диски (одинаковые производители, размеры, номер модели), то можно просто присоединить этот диск к системе. Убедитесь, что вы знаете установки для диска (например, первичный ведомый диск для интерфейса EIDE).

После присоединения диска загрузите систему ЗОСРВ «Нейтрино». После загрузки должен быть автоматически обнаружен жесткий диск, и для него будет создана запись в каталоге /dev. Если в системе имеются всего два диска, то новая запись появится в виде /dev/hd1. Если же имеется более двух дисков (дисководов), то, соответственно, вновь установленный диск может получить имя hd1, hd2 и т. д. В этом случае для идентификации конкретного диска придется использовать программу fdisk. Вновь установленный диск не должен иметь никаких разделов и быть чистым.

Предупреждение.

Перед началом выполнения последующих действий вы должны быть абсолютно уверены в правильной идентификации дисков. В противном случае можно скопировать содержимое пустого диска на диск, содержащий исходные данные. Затертые таким способом данные восстановить невозможно.

После идентификации дисков выполните команду:

```
cp -V /dev/hd0 /dev/hd1
```

где hd0 — исходный жесткий диск с данными, а hd1 — новый чистый диск, на который копируются данные.

По этой команде содержимое первого диска полностью копируется на второй диск, включая таблицы разделов, начальные загрузчики и т. д. Чтобы убедиться в идентичности дисков, нужно просто отключить исходный диск и поставить на его место диск-копию. После этого перезагрузить систему. Должна произойти загрузка ЗОСРВ «Нейтрино», и все должно выглядеть и работать, как на исходном диске. Сохраните резервную копию в безопасном
месте.

Образы типа Ghost

Некоторые пользователи ЗОСРВ «Нейтрино» выбирают в качестве резервных копий образы типа Ghost. Мы не рекомендуем делать это. С таких копий может неправильно восстанавливаться информация о разделах, что приведет к неправильной загрузке файловой системы. Если вы снова запустите на таком диске программу fdisk, то появится сообщение о некорректной информации, и программа fdisk запишет на диск неправильные данные.

18.5. Резервное копирование на удаленную систему

Создание резервной копии на удаленной системе является более безопасным способом хранения данных, чем хранение в локальной системе. Это связано с тем, что удаленный сервер обычно является более надежной системой.

В зависимости от ситуации, возможно, имеет смысл приобрести хорошую систему с оборудованием серверного класса, а затем купить обычные системы для ведения разработок. Не забывайте регулярно создавать резервные копии сервера.

Система CVS

ЗОСРВ «Нейтрино» поставляется с копией клиентской утилиты CVS (Concurrent Versions System, система управления версиями). Для использования системы CVS необходимо иметь CVS-сервер (желательно, чтобы администрирование сервера осуществлялось вашей компанией). С помощью системы CVS можно безопасно и удаленно управлять вашими архивами. Более подобно об этом см. раздел 17.

Удаленные файловые системы

Сохранение второй резервной копии на удаленной системе часто является простым, но эффективным способом предотвращения потери данных. Например, если базовый архив ваших программных кодов хранится в отдельном каталоге на локальной системе, и вдруг по непредвиденной причине отказывает жесткий диск, то вы потеряете и локальную, и резервную копию архива. Размещая же копию на удаленной файловой системе, вы эффективным образом снижаете шансы потери данных. Поэтому мы настоятельно рекомендуем использовать этот метод.

Примечание. Если вы размещаете файл в файловой системе, не относящейся к типу «Нейтрино», то могут быть утеряны существующие права доступа к файлу. Файлы в среде ЗОСРВ (как и во всех системах типа UNIX) имеют специальные права доступа (см. раздел 6), которые теряются, если отдельные файлы сохраняются в файловых системах типа Windows. Если создается архив (см. разд. "Архивация данных" ранее в этом разделе, то права доступа к файлам, записанным внутри архива, сохраняются.

Другие способы создания удаленных резервных копий

Существуют другие версии систем для удаленного резервного копирования (похожие на CVS), которые доступны в ЗОСРВ «Нейтрино» как продукты сторонних производителей. Многие из них являются бесплатными. Попробуйте поискать в Интернете инструменты, которые подойдут для использования в вашей компании и ее проектах.

18.6. Дисковая структура файловой системы QNX 4

Если вы когда-нибудь сталкивались с проблемами при работе в файловой системе QNX 4, то вам нужно представлять, как в этой файловой системе данные сохраняются на диске. Такое знание поможет распознать и, возможно, предотвратить повреждение данных, если вы захотите перестроить файловую систему. В заголовочном файле <sys/fs_qnx4.h> содержатся определения, касающиеся структуры данных, о чем пойдет речь в данном разделе.

С полным описанием файловой системы QNX 4 можно ознакомиться в разделе 11.

Компоненты раздела

Файловая система QNX 4 может занимать целый диск (например, в случае флоппи-дисков) или один из разделов жесткого диска. В пределах раздела жесткого диска в состав файловой системы QNX 4 входят следующие компоненты (рис. 18.1):

Рис. 18.1. Компоненты файловой системы QNX 4 в разделе диска

- блок загрузчика;
- корневой блок;
- блоки битовой карты;
- корневой каталог;
- другие каталоги, файлы, свободные блоки и т. д.

Указанные структуры создаются при инициализации файловой системы утилитой dinit.

Блок загрузчика

Первый физический блок раздела диска является блоком загрузчика. В нем содержится начальный код, который загружается программой BIOS и выполняет дальнейшую загрузку ОС с данного раздела. Если на диске отсутствуют разделы

(например, это флоппи-диск), то блоком загрузчика является первый физический блок диска.

Корневой блок

Корневым является второй блок раздела системы QNX 4. Его структура соответствует структуре стандартного каталога, и в ней содержится поле метки и информация по индексным дескрипторам (inode) для таких специальных файлов, как:

- корневой каталог файловой системы (обычно /)
- /.inodes
- /.boot
- /.altboot

В файлах /.boot и /.altboot содержатся образы операционной системы, которые могут быть загружены с помощью начального загрузчика.

Обычно для ЗОСРВ «Нейтрино» загружается образ, хранящийся в файле /.boot. Однако если имеется непустой файл /.altboot, то можно загрузить хранящийся в этом файле образ. Подробнее об этом см. раздел 8.

Блоки битовой карты

За корневым блоком последовательно располагается несколько блоков. Эти битовые блоки формируют битовую карту (матрицу) раздела системы QNX 4. Каждому физическому блоку раздела соответствует один бит карты. Таким образом, один блок битовой карты используется для каждых 4096 физических блоков диска (это соответствует дисковому пространству 2 Мбайт).

Если значение бита равно нулю, то соответствующий блок диска не используется. Неиспользуемые биты в конце последнего блока битовой карты (для которых нет соответствующих физических блоков на диске) установлены во включенное состояние.

Назначение битов происходит, начиная с самого младшего бита нулевого байта первого блока битовой карты – это в системе QNX 4 соответствует блоку с номером 1.

Корневой каталог

Корневой каталог располагается после блоков битовой карты. Корневой каталог – это обычный каталог (см. подразд. "Каталоги" далее в этом разделе) с двумя исключениями:

- каталоги с именами "точка" (.) и "две точки" (..) ссылаются на одну и ту же индексную информацию, а именно на индексный узел корневого каталога в корневом блоке;
- в корневом каталоге всегда имеются записи для файлов /.bitmap, /.inodes, /.boot и /.altboot; эти записи организованы таким образом, чтобы программы, которые передают информацию файловой системы, могли бы работать с записями как с нормальными файлами.

Корневой каталог изначально создается с помощью утилиты dinit размером в 4 блока, что обеспечивает достаточно места для записей на 32 каталога.

В корневом каталоге (/) имеются записи для нескольких специальных файлов (рис. 18.2, табл. 18.3), которые всегда присутствуют в файловой системе QNX 4. Эти файлы также создаются с помощью утилиты dinit во время первой инициализации файловой системы.

Таблица 18.3

Файл	**Описание**
/.	Ссылка на корневой каталог /
/..	Ссылка на корневой каталог /
/.bitmap	Файл с атрибутом "только чтение", в котором хранится битовая карта для всех блоков диска. Каждый бит указывает, используется ли соответствующий блок
/.inodes	Обычный файл, состоящий, по крайней мере, из одного блока для флоппи- или RAM-диска и из 16 блоков для других дисков. В файле /.inodes содержится набор записей индексных дескрипторов (inode). Первая запись является зарезервированной, это сигнатурная/информационная

Таблица 18.3

Файл	**Описание**
	область. Первые байты файла .inode содержат текст IamTHE.inodeFILE
/.longfilenames	Дополнительный файл, в котором сохраняется информация о файлах, чьи имена имеют длину более 48 символов. См. подразд. "Файловая система QNX 4" раздела 11
/.boot	Это файл образа ОС, который будет загружен в память во время процесса стандартной загрузки. Если файл загрузки отсутствует, то этот файл будет иметь нулевую длину
/.altboot	Это файл образа ОС, который будет загружен в память во время процесса альтернативной загрузки. Если файл загрузки отсутствует, то этот файл будет иметь нулевую длину

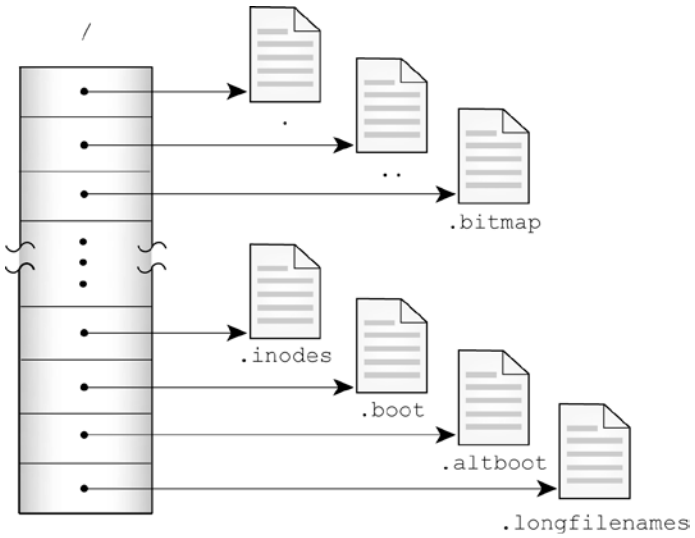


Рис. 18.2. Содержимое корневого каталога /

Каталоги

Каталог представляет собой файл, имеющий в файловой системе специальное назначение. В нем содержится коллекция элементов каталога (рис. 18.3).

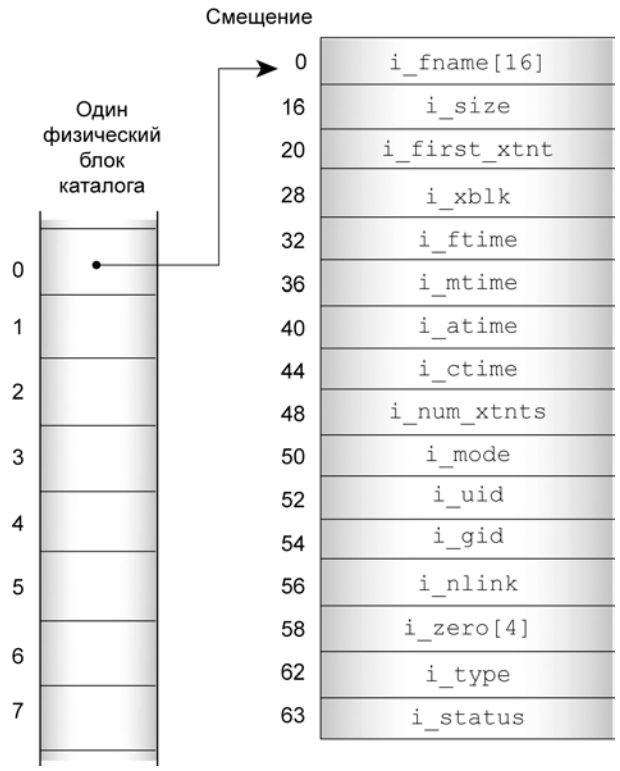


Рис. 18.3. Элемент каталога

Биты в поле i_status указывают тип элемента каталога (табл. 18.4).

Таблица 18.4

QNX4FS_FILE_LINK	QNX4FS_FILE_USED	Тип элемента
0	0	Неиспользуемый элемент каталога
0	1	Обычный, используемый элемент каталога
1	0	Ссылка на соответствующую запись в файле /.inodes (которую следует использовать)
1	1	Некорректный тип элемента

Первый элемент каталога всегда используется для ссылки с именем . ("точка") и включает сигнатуру каталога. Этот элемент ссылается на сам каталог, указывая на элемент внутри родительского каталога, который описывает данный каталог.

Второй элемент каталога всегда имеет запись о связи с именем .. ("две точки"). Этот элемент ссылается на родительский каталог, указывая на первый блок родительского каталога.

Каждый элемент каталога либо определяет файл, либо указывает на запись внутри файла /.inodes. Индексные записи inode используются в случае, когда имя файла оказывается длиннее 16 символов или когда одно или более имен связаны в один файл. Если включена поддержка длинных имен, то в корневом каталоге файловой системы присутствует также файл .longfilenames, в котором сохраняется информация о файлах, чьи имена имеют длину более 48 символов.

Первый экстент файла (если он имеется) описывается в элементе каталога или в записи индексного дескриптора. Дополнительные экстенты файла требуют использования связанного списка блоков экстентов, начало которого находится также в элементе каталога в записи индексного дескриптора. В каждом блоке экстентов может содержаться информация, но не более чем 60 экстентов.

Ссылки

Файлы с длиной имени более 16 символов и файлы, которые являются ссылками (link) на другие файлы, имеют специальную форму записи в элементе каталога. Для этих записей в поле i_status устанавливается бит QNX4FS_FILE_LINK (0x08).

Для таких файлов часть элемента каталога перемещается в файл /.inodes (рис. 18.4).

Если имя файла оказывается длиннее 48 символов, то:

- поле l_fnames элемента каталога содержит усеченную до 48 символов версию имени;
- поле l_lfn_block указывает на запись в файле .longfilenames.

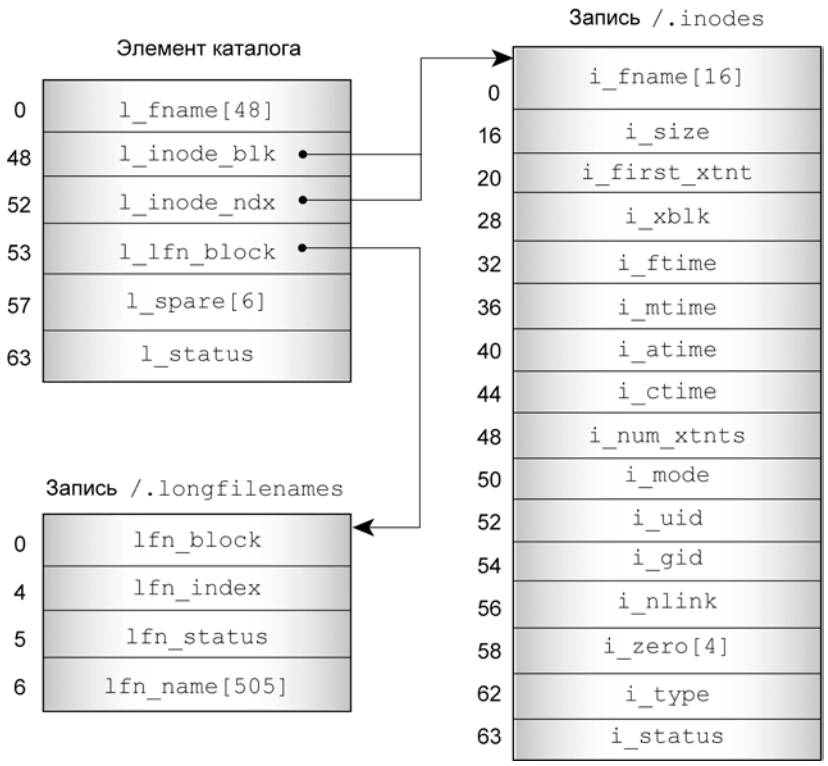


Рис 18.4. Запись индексного дескриптора

Блоки экстентов

Блоки экстентов используются для любого файла, у которого есть больше чем один экстент. Поле i_xblk в элементе каталога указывает на один из таких блоков экстентов, в котором, в свою очередь, определяется, где могут быть найдены второй и последующие экстенты.

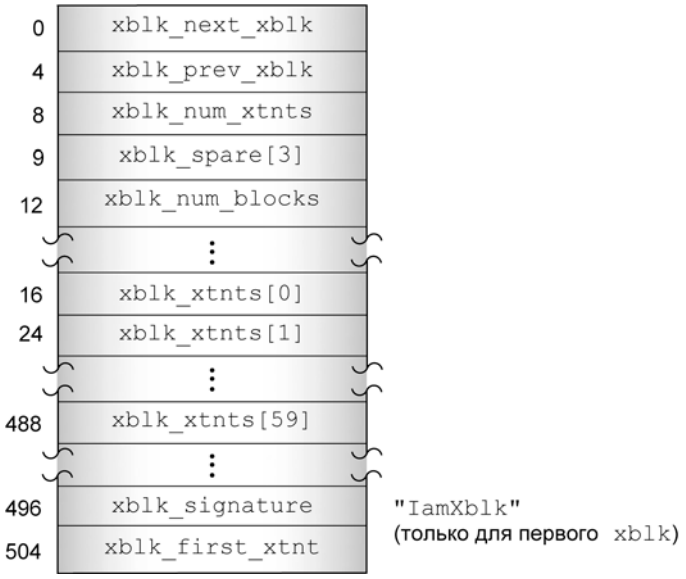


Рис. 18.5. Блок экстентов

Блок экстентов занимает в точности один сектор диска размером 512 байтов и имеет формат, представленный на рис. 18.5.

Каждый блок экстентов содержит:

- указатели на предыдущий/следующий блок;
- счетчик экстентов;
- счетчик всех блоков во всех экстентах, определяемых данным блоком экстентов;
- указатели и счетчики для каждого экстента;
- сигнатуру (IamXblk).

В первом блоке экстентов содержится также избыточный указатель на первый экстент файла (он присутствует также в элементе каталога или в записи индексного дескриптора). Это позволяет восстановить все данные в файле, обнаружив только один этот блок.

Файлы

Файлы или экстенты файлов представляют собой группы блоков, указанные в элементах каталога или в записях inode. Для них в файловой системе QNX 4 нет какой-либо специальной структуры.

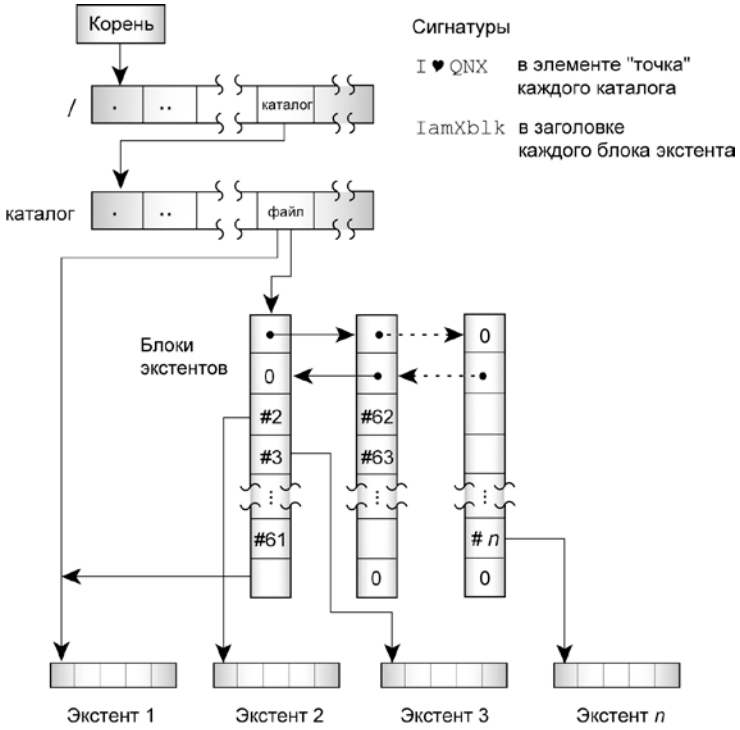


Рис. 18.6. Файловая структура в QNX 4

18.7. Утилиты для обслуживания файлов

Если случаются какие-либо аварийные отказы, то для восстановления системы и файлов можно использовать следующие утилиты:

- fdisk;
- dinit;
- chkfsys;
- dcheck;
- zap;
- spatch.

В этом разделе дается краткое описание перечисленных утилит. Более подробно о них см. в руководстве «Описание программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

fdisk

С помощью утилиты fdisk можно создавать и обслуживать блок раздела на жестком диске. Этот блок совместим с другими операционными системами, и работа с ним может осуществляться с помощью программ fdisk от других ОС (однако преимуществом нашей версии программы является то, что она распознает специфическую для ЗОСРВ «Нейтрино» информацию). Если загрузчик раздела отсутствует или поврежден, то утилита fdisk может создать его.

Примечание. Мы рекомендуем сохранять в виде печатной копии информацию о таблице разделов каждого из дисков в сети.

dinit

С помощью утилиты dinit создаются следующие компоненты файловой системы QNX 4:

- блок загрузчика;
- корневой блок;
- блоки битовой карты;
- корневой каталог;

- файл /.inodes;
- файл /.longfilenames.

Если первые несколько блоков вашей файловой системы оказываются поврежденными, то можно попытаться их восстановить с помощью сначала утилиты dinit с ключом -r, а затем утилиты chkfsys. Более подробно об утилите dinit см. в руководстве "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01.

chkfsys

Утилита chkfsys является главным инструментом, необходимым для обслуживания файловой системы.

Примечание. Утилита chkfsys запрашивается при повреждении файловой системы устойчивой к сбоям питания; для этого типа файловой системы необходимо использовать chkqnx6fs.

С ее помощью можно выполнить следующее:

- проверить структуру каталогов для всего раздела диска, получить сообщения о наличии противоречивых данных и, если возможно, устранить обнаруженные противоречия;
- проверить распределение блоков для всего диска;
- записать новый файл /.bitmap (при подтверждении с вашей стороны).

При запуске утилиты chkfsys предполагается, что корневой блок имеет нормальную рабочую структуру. Если структура корневого блока нарушена, то выдается сообщение об этом, и утилита chkfsys прекращает работу. Восстановление структуры корневого блока производится с помощью утилиты dinit.

dcheck

С помощью утилиты dcheck происходит поиск на диске поврежденных секторов, предпринимающий попытку прочесть каждый сектор. Если при запуске утилиты указывается ключ -m, то поврежденный сектор изымается из

использования с помощью соответствующей отметки в битовой карте (файл /.bitmap).

Если во время работы утилиты обнаруживается файл /.bad_blks, то происходит обновление битовой карты и воссоздание файла /.bad_blks. Утилиту dcheck можно запускать несколько раз для увеличения надежности распознавания поврежденных секторов и их добавления в файл /.bad_blks.

zap

С помощью утилиты zap пользователь с учетной записью root может удалять из файловой системы файлы или каталоги без возвращения использованных блоков в список освободившихся. Это может потребоваться, например, в случае повреждения элемента каталога или при возникновении ситуации, когда два файла ошибочно занимают на диске одно и то же место.

Восстановление ошибочно стертого файла

Если вы по ошибке стерли файл, то иногда стертый файл можно восстановить с помощью утилиты zap, если использовать ключ -u и запустить утилиту немедленно после удаления. Восстановление удаленного файла с помощью утилиты zap возможно при выполнении следующих условий:

- элемент каталога для только что удаленного файла еще не был использован;
- блоки диска, ранее использовавшиеся для удаленного файла, не были назначены и закреплены за другим файлом.

spatch

Иногда может оказаться, что из-за повреждения диска полностью потеряны файлы или каталоги. Если выяснится, что после работы утилиты chkfsys не были восстановлены некоторые ключевые файлы и каталоги, то можно попробовать запустить утилиту spatch для восстановления некоторых или даже всех данных.

Утилита spatch дает возможность просканировать весь диск напрямую и устранить мелкие проблемы. Иногда удается устранить временно возникающие

проблемы на диске путем чтения и перезаписи сбоящего блока с помощью утилиты spatch.

Примечание. Перед тем как использовать утилиту spatch, убедитесь в том, что вы понимаете тонкости структуры файловой системы QNX 4. Обратитесь еще раз к подразд. "Дисковая структура файловой системы QNX 4" ранее в этом разделе.

18.8. Восстановление дисков и файлов

Использование утилиты chkfsys

Утилита chkfsys является вашим основным инструментом для проверки и восстановления потенциально поврежденной файловой системы. С ее помощью можно идентифицировать и откорректировать массу небольших проблем, а также проверить целостность всей дисковой системы.

Обычно для нормальной работы утилиты chkfsys необходимо, чтобы файловая система была в неиспользуемом состоянии и на устройстве не были открыты никакие файлы. Нужно закрыть все процессы, которые открывают или могут открыть файлы во время работы chkfsys.

Для запуска утилиты chkfsys в точке монтирования введите команду:

chkfsys точка_монтирования

Утилита выполнит сканирование всего раздела диска, начиная с корневой области и далее вниз по структуре каталогов, выполняя построение внутренней копии битовой карты и проверяя целостность всех найденных в процессе сканирования файлов и каталогов.

После окончания обработки всех файлов производится сравнение внутренней копии битовой карты с содержимым существующей битовой карты на диске. Если данные согласуются, работа программы chkfsys успешно завершается. Если найдены несоответствия, то после получения согласия со стороны пользователя произойдет перезапись битовой карты в соответствии с найденными и верифицированными файлами.

В дополнение к верификации выделения блоков (битовой карты) при работе утилиты chkfsys делается попытка исправить любые возникающие в процессе сканирования проблемы. Например, с помощью утилиты chkfsys можно:

- снять метку занятости с файлов, для которых в момент возникновения сбоя производилась операция записи;
- откорректировать размер файла в элементе каталога, чтобы он соответствовал реальному значению.

Когда нужно запускать утилиту chkfsys

Весьма полезно запускать утилиту chkfsys при проведении регулярно выполняемых процедур технического обслуживания — это позволит убедиться в том, что данные на диске не повреждены. Например, хорошо бы запускать утилиту chkfsys на сетевых серверах при каждой их загрузке. Автоматическая проверка файловой системы в момент загрузки гарантирует, что во время сканирования будет сделана попытка устранить все обнаруженные проблемы. Для автоматизации этого процесса добавьте chkfsys в файл re.local на сервере (см. раздел 8).

Утилиту chkfsys особенно важно запускать после системного сбоя, аварийного отключения источника питания, после неожиданного перезапуска системы. Это позволит проверить, не были ли повреждены какие-нибудь файлы. Утилита проверяет на диске состояние флага "целостности" файловой системы ("clean" flag), чтобы определить, находится ли в настоящее время система в целостном, непротиворечивом состоянии.

Флаг "целостности" хранится на диске, его установка или сброс производится системой. Сброс этого флага происходит в момент окончания монтирования файловой системы, а установка — при размонтировании файловой системы. При установленном флаге "целостности" подразумевается, что файловая система не повреждена. Если при запуске утилиты chkfsys флаг "целостности" сброшен, то делается попытка устранить проблему.

Для утилиты chkfsys можно использовать ключ -u, при котором установленный флаг "целостности" игнорируется, и утилита запускается безусловно. Такая необходимость может возникнуть, когда:

- при работе утилиты dcheck обнаруживаются поврежденные секторы;
- были намеренно удалены или затерты (zapped) некоторые файлы;
- нужно принудительно запустить общую проверку диска.

Использование утилиты chkfsys на работающей системе

Для проведения всесторонней верификации диска с помощью утилиты chkfsys обычно требуется эксклюзивное использование файловой системы.

Предупреждение.

Запуск утилиты chkfsys на работающей системе сопряжен с определенным риском, потому что со стороны утилиты и со стороны файловой системы может производиться чтение и, возможно, даже запись одного и того же блока диска.

Если вы все же запустите утилиту chkfsys при работающей файловой системе и утилита выполнит запись чего-нибудь, то файловой системе будет отправлено сообщение о необходимости приостановки ее работы. В ответ на это сообщение файловая система перемонтирует себя и обратится к диску для повторного считывания всех данных. В этом случае все открытые файлы будут помечены как устаревшие, при попытке чтения или записи будет выдаваться ошибка типа EIO, пока файлы не будут закрыты и не будут открыты снова. Подобная ситуация может относиться, например, к файлам типа системных журналов.

Статические изменения в файлах и каталогах, которые в текущий момент не открыты файловой системой, вероятно, не приведут к таким проблемам.

Если запущено приложение, для которого невозможна приостановка работы, или невозможно запустить утилиту chkfsys, потому что файлы были открыты для обновления, то попробуйте при запуске утилиты chkfsys использовать ключ -f:

```
chkfsys -f /dev/hd0t79
```

При таком варианте запуска для утилиты устанавливается специальный режим "только чтение", при котором возможна проверка всей файловой системы с открытыми файлами (без проведения каких-либо корректирующих действий). Результаты такой проверки могут дать подсказку о том, что делать дальше.

Восстановление при наличии поврежденного блока в середине файла

На жестких дисках со временем появляются поврежденные секторы. В большинстве случаев удается восстановить большинство данных или даже все данные файла, содержащего поврежденные секторы.

Некоторые поврежденные секторы возникают из-за отказов или сбоев питания или из-за плохой среды носителя жесткого диска. В этих случаях иногда простые операции чтения и перезаписи "восстанавливают" сектор на некоторое время. За это время можно успеть скопировать весь файл в другое место, пока сектор не откажет снова. От такой процедуры нет никакого вреда, и ее часто стоит попробовать.

Для проверки блоков внутри файла используется утилита spatch. Когда дело доходит до поврежденного сектора, утилита spatch должна выдавать сообщение об ошибке, но при этом из сектора может быть фактически считана порция "хороших" байтов. Запись назад того же самого сектора часто заканчивается успешно.

В то же самое время при перезаписи восстанавливается правильное значение CRC (Cyclic Redundancy Check, контроль при помощи циклического избыточного кода), что может снова сделать блок неповрежденным (хотя, возможно, и с ошибочными данными).

После этого можно скопировать файл в другое место и применить к ранее поврежденному файлу утилиту zap. Для завершения процедуры ненадежный блок помечается как поврежденный (путем добавления его в файл /.bad_blks), после чего запускается утилита chkfsys для восстановления остальных неповрежденных блоков.

Если такая процедура не поможет, то можно воспользоваться утилитой spatch для копирования в другой файл как можно большего числа блоков поврежденного файла. После этого нужно применить к поврежденному файлу утилиту zap, а затем запустить утилиту chkfsys.

18.9. Что делать, если система больше не загружается?

Если ранее работавшая система на основе ЗОСРВ «Нейтрино» неожиданно перестает работать и не загружается, то это может быть следствием следующего:

- произошел отказ оборудования или были повреждены данные на жестком диске;
- кто-то либо изменил/перезаписал файл загрузки, либо изменил системный файл инициализации – это два наиболее вероятных сценария.

Далее приводятся шаги, которые помогут вам идентифицировать проблему. Если это возможно, предлагаются корректирующие действия.

Попробуйте загрузиться с CD-диска или по сети.

- если имеется сеть, через которую можно загрузиться, то попробуйте это сделать. После того как компьютер загрузится, войдите в систему с учетной записью root.
- если сеть отсутствует, загрузитесь с инсталляционного компакт-диска. В этом случае файловая система уже будет запущена, и вы будете подключены к системе под учетной записью root.

Запустите драйвер жесткого диска. Например, для запуска драйвера SCSI-адаптера типа Adaptec 4 введите команду:

```
devb-aha4 ключи &
```

Если используется другой тип адаптера, введите его имя, например:

```
devb-eide ключи qnx4 ключи &
```

В этом случае будет создан блочно-ориентированный файл с именем /dev/hd0, который представляет весь жесткий диск.

Запустите утилиту fdisk.

Запуск этой утилиты сразу же даст вам полезную информацию о состоянии жесткого диска. Утилита fdisk может выдать информацию, касающуюся одной из нескольких типов проблем (табл. 18.5).

Таблица 18.5

Проблема	**Вероятная причина**	**Рекомендации по устранению**
Ошибка при чтении блока 1	Произошел отказ контроллера диска или самого диска	Если жесткий диск исправен, то замена контроллера, возможно, позволит продолжить работу с тем же диском. В противном случае придется заменить жесткий диск, переустановить ЗОСРВ «Нейтрино» и восстановить файлы из резервной копии
Неправильные параметры диска	Возможно, потеряна информация о данном жестком диске. Одна из вероятных причин — разряд батареи питания CMOS-памяти	Обычно перезапуск процедуры настройки оборудования (или (для PS/2) программируемой процедуры выбора настройки) дает положительный результат. Конечно же, замена батареи даст более надежное решение проблемы
Неверная информация о разделах	Если утилита fdisk показывает правильный размер диска, но неверно отображается информация о разделах, это значит, что почему-то были разрушены данные в блоке 1 физического диска	Используйте утилиту fdisk для восстановления правильной информации о разделах. Очень полезно записать заранее или распечатать правильные данные о разделах в блоке 1 жесткого диска. Это значительно облегчит задачу восстановления информации при реализации данного шага

Произведите монтирование раздела и файловой системы.

К этому моменту было установлено, что оборудование работает (по крайней мере, это касается блока 1) и что для ЗОСРВ «Нейтрино» определено правильное разделение на разделы. Теперь нужно создать блочно-ориентированный файл для самого раздела QNX 4 и смонтировать его в качестве файловой системы QNX 4:

```
mount -e /dev/hd0
```

```
mount /dev/hd0t79 /hd
```

После этого будет создан том с именем /dev/hd0t79. В зависимости от состояния раздела QNX 4 монтирование может быть успешным или неудачным. Если информация о разделе правильна, то проблем не будет. Поскольку корневой каталог (/) уже существует (на компакт-диске или на удаленном диске сети), то мы смонтировали раздел локального жесткого диска как файловую систему с именем /hd.

Теперь основная задача состоит в запуске на диске утилиты chkfsys для проверки файловой системы, а возможно, и устранения ее неисправностей.

Примечание. Если загрузка происходила с компакт-диска, и не ожидается, что есть какие-либо неисправности в файловой системе на жестком диске (например, система не могла загрузиться из-за простой ошибки в загрузочном файле или в системном файле инициализации), то можно просмотреть символическую связь с разделом вашего жесткого диска в префиксном дереве в памяти администратора процессов:

```
ln -sP /hd /
```

Если вы запустили эту команду, то остальную часть текста в этом разделе можно пропустить.

Если утилита mount не работает

Если не удается выполнить команду монтирования, то, скорее всего, поврежден раздел QNX 4 (поскольку драйвер откажется монтировать нечто, что рассматривается как некорректная файловая система).

В этом случае можно воспользоваться утилитой dinit, чтобы восстановить на диске достаточное количество хороших данных, с которыми сможет работать драйвер:

```
dinit -hr /dev/hd0t79
```

Использование ключа -r является указанием на перезапись следующей информации:

- корневого блока;
- битовой карты (с информацией обо всех выделенных блоках);
- постоянной части корневого каталога.

Теперь нужно попробовать снова ввести команду mount и попытаться создать для файловой системы QNX 4 точку монтирования с именем /hd.

После успешного монтирования необходимо перестроить битовую карту с помощью утилиты chkfsys, даже для хорошего раздела диска.

Хотя бы часть файловой системы QNX 4 теперь должна быть доступной. Можно запустить утилиту chkfsys, для того чтобы проверить файловую систему и восстановить максимальный объем данных.

Если жесткий диск смонтирован как /hd (например, загрузка производилась с компакт-диска), то введите команду:

```
путь_на_CD/chkfsys /hd
```

Если жесткий диск смонтирован как / (например, при загрузке по сети), введите команду:

```
сетевой_путь/chkfsys  /
```

В любом случае выполните следующее:

- запустите (если это возможно) программу chkfsys из любого другого места, кроме файловой системы, которую вы пытаетесь восстановить;
- обращайте внимание на появляющиеся при работе программы сообщения и дайте программе возможность исправить максимальное число неполадок.
- дальнейшие действия зависят от результатов работы программы chkfsys.

Если диск не восстанавливается

Если диск окажется совершенно не восстанавливаемым, можно попробовать использовать утилиту spatch (см. ранее) для восстановления отдельных файлов и каталогов. В некоторых случаях можно попробовать повторно инсталлировать ЗОСРВ «Нейтрино» и восстановить диск из резервной копии.

Если значительная часть файловой системы безвозвратно потеряна или потеряны важные файлы, то восстановление из резервной копии будет наилучшим вариантом.

Если файловая система не повреждена

Если файловая система не повреждена, но компьютер по-прежнему отказывается загружаться с жесткого диска, то, вероятно, повреждения следует искать:

- в программе загрузчика раздела в физическом блоке 1;
- в загрузчике ЗОСРВ «Нейтрино» в первом блоке раздела QNX 4.

Для перезаписи загрузчика раздела можно воспользоваться утилитой fdisk:

```
fdisk /dev/hd0 loader
```

Для перезаписи загрузчика системы можно воспользоваться утилитой dinit:

```
dinit -b /dev/hd0t79
```

Теперь ваша система должна успешно загружаться.

19. Обеспечение безопасности системы

Сейчас, когда все больше и больше компьютеров и других устройств подключаются к незащищенным сетям, вопросы безопасности становятся весьма актуальными. Слово "безопасность" может иметь много значений, но применительно к компьютерам под ней обычно понимается задача запрета неавторизованным пользователям выполнять на компьютере действия, которые вы им не разрешаете выполнять.

В книгах и Интернете можно найти обширную информацию, касающуюся безопасности. В данном разделе дается лишь краткое введение в тему о безопасности, представляются ссылки на внешние источники и ресурсы, а также обсуждаются особенности, применимые именно к ЗОСРВ «Нейтрино».

19.1. Общие вопросы безопасности ОС

Совершенно очевидно, что безопасность является очень важным фактором. Никто не хочет, чтобы кто-то получил контроль над его устройством и мог бы нарушить нормальную работу устройства. Представьте, какие последствия могут быть в случае, если кто-то остановит систему управления воздушным движением или нарушит нормальную работу оборудования больницы.

Важность соблюдения мер безопасности для каждой отдельной машины зависит от контекста:

- машина с установленным межсетевым экраном менее уязвима, чем машина, напрямую подключенная к сети общего пользования;
- наименьшая опасность грозит компьютеру, вообще не подключенному к сети.

Частью мер по обеспечению безопасности является установление степени риска. Введение классификации угроз по категориям может помочь в анализе проблем и увидеть, на что нужно обращать внимание в первую очередь.

Удаленные и локальные атаки

Можно поделить угрозы безопасности, называемые также эксплойтами (exploit), на две крупные категории.

- удаленный эксплойт. Атакующий подключается к машине через сеть и пользуется ошибками или уязвимостями в системе.
- локальная атака. Атакующий имеет учетную запись в системе, на которую производится атака. Он может использовать эту учетную запись для запуска неавторизованных задач.

Удаленные эксплойты

Удаленные эксплойты обычно гораздо опаснее, чем локальные. К счастью, воздействие удаленных эксплойтов гораздо легче предотвратить, и они менее распространены.

Например, вы запускаете сервер bind на порте 53 (разрешение имен DNS) компьютера, подключенного к сети. Допустим, конкретная версия программного обеспечения имеет уязвимость, из-за которой намеренно неправильно сформированный атакующим запрос может привести к запуску командного процессора от имени учетной записи root для другого порта машины. Атакующий может использовать эту уязвимость для подключения и эффективного "владения" данным компьютером.

Такой тип эксплойта часто называется атакой с переполнением буфера (buffer overrun) или с разрушением стека (stack-smashing). Более подробно этот вид атак описан в статье Алефа Уана (Aleph One) "Smashing the Stack for Fun and Profit" ("Разрушение стека для забавы и выгоды") на странице сайта **http://www.insecure.org/stf/smashstack.txt**. Для простого разрешения этой проблемы нужно просто знать, какие порты прослушиваются каждым из серверов, и нужно использовать самые последние версии программного обеспечения. Если машина подключена к сети, то не запускайте на ней больше служб, чем это действительно необходимо.

Локальные эксплойты

Локальные эксплойты гораздо более распространены, и их труднее предотвратить. Наличие локальной учетной записи подразумевает определенную степень доверия, и не всегда просто представить, как это кто-то может переступить через оказанное доверие. Большинство локальных эксплойтов включают в себя тот или иной вид превышения предоставленных полномочий, например, преобразование обычного пользователя в суперпользователя (учетная запись root).

Многие локальные атаки являются следствием неправильного конфигурирования системы (например, неправильно установлены права доступа к файлам) или из-за переполнения буфера при запуске исполняемой программы от имени пользователя root (когда идентификатор пользователя назначается программе путем установки бита setuid). В мире встраиваемых систем – где обычно используется ЗОСРВ «Нейтрино» – эта проблема для локальных пользователей неактуальна, тем более что для многих систем вообще не поставляется командный процессор (оболочка).

Эффективность атак

Эксплойты можно также классифицировать по производимому эффекту.

- атаки с целью получения контроля над ресурсами компьютера (takeover attack).

Результатом этого вида атак является получение полного доступа к ресурсам компьютера, по крайней мере, получение возможности выполнять действия, не предсказуемые для владельца, но вполне осознанно выполняемые атакующим.

- атаки типа "Отказ в обслуживании", или DOS-атаки (Denial Of Service).

Эти атаки приводят к нарушению нормального функционирования. В качестве примера можно привести массовое и частое "пингование" машины (использование извне команды ping) с целью замедления реакции машины на запросы из сети вплоть до полной невозможности пользоваться сетью. С DOS-атаками исключительно трудно иметь дело. Реагировать на них можно лишь по

фактическому результату, и нельзя предпринять никаких подготовительных действий.

Для примера, существует очень мало систем, которые злонамеренный локальный пользователь может "поставить на колени". Эти атаки часто можно минимизировать, используя, например, встроенную команду ulimit из командной оболочки ksh.

Используя приведенную классификацию, можно посмотреть на систему с точки зрения того, для каких атак система наиболее уязвима и что нужно сделать для предотвращения имеющейся уязвимости.

Вирусы

Под вирусом обычно понимается специальная инфицирующая программа, которая запускает на компьютере некоторый код (например, троянская программа). Для вирусов необходим хостовый компьютер и точка входа.

В качестве точек входа для вирусов используются:

- открытый интерфейс (например, ActiveX) – ничего подобного нет в ЗОСРВ «Нейтрино»;
- "дыра" в системе безопасности (например, возможность переполнения буфера) – это имеет место при наличии дефектов в конкретных услугах, основанных на стандартных промышленных программах общего применения; у этого варианта мало возможностей, поскольку мы поставляем только ограниченный набор стандартных (BSD) услуг.

Для вируса хосты являются интерфейсами с системными вызовами, которые доступны из точки входа (инфицированная программа), например, программа sendmail или HTTP-сервер. Хосты имеют привязку к платформе, поэтому, например, вирус для Linux вероятнее всего прекратит работу хоста под управлением ЗОСРВ «Нейтрино», как только он попытается сделать что-нибудь разрушительное.

Вирусы, которые распространяются через электронную почту, привязаны к конкретной ОС. Обычно их целью является ОС Windows, и они не могут нанести

вреда системам на базе ЗОСРВ «Нейтрино» по причине несовместимости. Большинство систем семейства UNIX не восприимчивы к вирусам, поскольку возможность провести (большие) разрушительные воздействия ограничена самим хостом. Мы пока еще не слышали ни об одном вирусе, который мог бы инфицировать систему на основе ЗОСРВ «Нейтрино».

Кроме того, поскольку установленные системы на основе ЗОСРВ «Нейтрино» настолько сильно специализированы для выполнения конкретных приложений, что в них зачастую просто не содержится программного обеспечения, открытого для вирусных атак (возможности входа пользователя в систему, Web-браузеры, электронная почта, серверы Telnet и FTP).

Общие вопросы безопасности для ЗОСРВ «Нейтрино»

ЗОСРВ «Нейтрино» относится к операционным системам класса UNIX, поэтому почти вся информация, касающаяся мер безопасности для UNIX (как общая, так и относящаяся к ОС Linux, BSD и т. д.), применима также и к ЗОСРВ «Нейтрино». Поиск в Интернете статей по безопасности UNIX или Linux даст вам много ссылок по этой теме. А также по этому вопросу существует большое количество литературы в книжных магазинах и библиотеках.

Для того чтобы ЗОСРВ «Нейтрино» была более гибкой в использовании и максимально привычной и знакомой при эксплуатации, мы позаимствовали из UNIX общую модель обеспечения безопасности. В нее включены понятия учетных записей пользователей и установления прав доступа к файлам, что обычно достаточно для всех наших клиентов. Во встраиваемых устройствах очень легко без дополнительных затрат ввести ограничения любой степени по доступу к системе. В отличие от встраиваемых, сверхбезопасными системами, требующими сертификации, обычно являются серверы.

Более подробно об этом см. раздел 3 и подразд. "Владение файлами и права доступа" раздела 6.

19.2. Безопасность в ЗОСРВ «Нейтрино»

Как было сказано в предыдущих разделах, ЗОСРВ «Нейтрино» имеет потенциальные уязвимости по отношению к тем же угрозам, что и другие ОС семейства UNIX. Кроме того, имеется ряд вопросов, которые характерны только для ЗОСРВ «Нейтрино».

Передача сообщений

Базовая модель операций основана на передаче сообщений между ядром ОС, администратором процессов и другими службами. В этой области есть место для потенциальных локальных угроз безопасности, которых могло бы не быть в системе, где все драйверы занимают одно и то же адресное пространство с ядром. Конечно, потенциальная слабость перевешивается доказанной силой такой модели, т. к. во встраиваемых системах локальные атаки обычно имеют второстепенное значение.

Более подробно о микроядерной архитектуре и передаче сообщений см. в главе по микроядерной архитектуре ЗОСРВ «Нейтрино» документа «Описание применения. Часть 1. Системная архитектура» КПДА.10964-01 31 01.

pdebug

Удаленный агент отладки pdebug запускается на целевой системе и взаимодействует с отладчиком gdb на хосте. Агент pdebug может быть запущен как специализированный сервер на порте, быть порожденным процессом от сервиса inetd на входящие соединения или порожденным процессом от сервиса qconn.

Агент pdebug обычно запускается от имени учетной записи root, поэтому кто-нибудь может выгрузить в удаленный компьютер (upload), загрузить на хост (download) или запустить на выполнение произвольный код с уровнем привилегий root. Этот агент был создан для взаимодействия с системами разработки, а не для использования в промышленных образцах. Поэтому не предусмотрено никаких средств аутентификации и безопасности, и разработка таких средств не планируется в будущем. См. следующий раздел.

qconn

Сервис qconn представляет собой сервер, который запускается на целевой машине и обрабатывает все входящие запросы от среды разработки IDE. Сервер qconn порождает процесс pdebug для обработки запросов на отладку, профилирует приложения, осуществляет сбор системной информации и т. д.

Подобно процессу pdebug, сервис qconn является по своей сути незащищенным, и он предназначен для работы совместно с системами разработки. В отличие от процесса pdebug, мы планируем встроить в сервис qconn модель безопасности с использованием определенной формы аутентификации. Это даст возможность применять qconn на промышленных машинах для реализации таких сервисов, как удаленное обновление и коррекция ошибок.

Qnet

Qnet – это прозрачный сетевой протокол ЗОСРВ «Нейтрино». Он описан в разделе 12 данного руководства и в главе 13 руководства "Операционная система реального времени ЗОСРВ «Нейтрино». "Описание применения. Часть 1. Системная архитектура" КПДА.10964-01 31 01.

С помощью протокола Qnet в файловой системе отображаются другие машины сети с ЗОСРВ «Нейтрино». При этом удаленные машины рассматриваются как расширения локальной машины. Кроме присвоения идентификатора пользователя по входящему соединению, не используется никаких других средств аутентификации. Поэтому соблюдайте осторожность при работе на машине, которая доступна из сети общего пользования.

Для того чтобы сделать протокол Qnet более безопасным, используйте ключи maproot и mapany, с помощью которых входящие соединения (с учетной записью root или какой-либо другой) отображаются на идентификатор заданного пользователя. Подробнее об этом см. в разделе по файлу lsm-qnet.so в руководстве "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01.

IPSec

Протокол IPSec является протоколом безопасности для уровня Интернета. Протокол можно использовать, например, для установления безопасного туннельного соединения между компьютерами или сетями. В него входят указанные далее два подпротокола.

- AH (Authentication Header, аутентификационный заголовок) — гарантирует целостность IP-пакетов и защищает их от изменений и подмены на промежуточных этапах передачи за счет присоединения криптографической контрольной суммы, вычисляемой с помощью односторонней хеш-функции.
- ESP (Encapsulated Security Payload, протокол шифрования) — защищает содержимое IP-пакетов от прослушивания за счет шифрования криптографическим алгоритмом с секретным ключом.

У протокола IPSec есть два режима работы:

- транспортный — защищает соединение типа peer-to-peer между узлами конечных пользователей;
- туннельный — поддерживает операции инкапсуляции типа IP-in-IP, предназначен для организации безопасных шлюзов, например, VPN-конфигураций.

Примечание. Поддержка протокола IPSec может изменяться по мере доработки протокола.

Более подробно по вопросу включения протокола IPSec см. подразд. "Распознавание устройств" раздела 8.

19.3. Установка межсетевого экрана

При построении зданий или судов для предотвращения распространения огня в них используют специально сконструированные стены. В компьютерах для предотвращения или ограничения доступа к определенным приложениям или системам и для защиты систем от злонамеренных атак используют межсетевые экраны

Для создания межсетевого экрана в среде ЗОСРВ «Нейтрино» применяется комбинация двух механизмов:

- фильтрация IP-пакетов для контроля над доступом к вашей машине;
- трансляция сетевых адресов (NAT), известная пользователям Linux как маскарадинг (masquerading), с помощью которой можно организовать выход нескольких компьютеров в сеть через общий внешний интерфейс.

Более подробную информацию см. в документации по OpenBSD по адресу ftp://ftp3.usa.openbsd.org/pub/OpenBSD/doc/pf-faq.pdf.

20. Точная настройка системы

20.1. Получение информации о статусе системы

В составе ЗОСРВ «Нейтрино» имеются следующие утилиты, которые могут быть использованы для точной настройки системы:

- hogs — вывести список процессов, которые в текущий момент времени занимают ЦПУ;
- pidin (Process ID INfo) — отобразить системную статистику;
- ps — вывести данные о статусе процессов;
- top —— отображение использования системы (Unix)

Более подробно об этом см. в руководстве по утилитам "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01.

Для отображения более подробных данных необходимо использовать tracelogger и Analysis Toolkit (см. подраздел «SAT»). SAT записывает в журнал события ядра, изменения состояния системы, с использованием специально-оборудованной версией ядра (procnto*-instr).

20.2. Улучшение производительности

После запуска утилиты hogs вы сможете узнать, какие процессы занимают наибольшее процессорное время. Например:

```
$ hogs -n -% 5
PID             NAME   MSEC   PIDS  SYSTEM
1                      1315    53%     43%
6          devb-eide    593    24%     19%
54358061         make    206     8%      6%

1                      2026    83%     67%
6          devb-eide    294    12%      9%

1                      2391    75%     79%
6          devb-eide    335    10%     11%
54624301   htmlindex    249     7%      8%
```

```
1                      1004    24%     33%
54624301   htmlindex   2959    71%     98%

54624301   htmlindex   4156    96%    138%

54624301   htmlindex   4225    96%    140%

54624301   htmlindex   4162    96%    138%

1                        71    35%      2%
6          devb-eide     75    37%      2%

1                      3002    97%    100%
```

Давайте посмотрим на результаты вывода. Первая итерация показывает, что процесс с PID=1 занимает 53% процессорного времени. Процесс с номером 1 всегда является администратором процессов, procnto. В данном случае это поток режима простоя (ожидания), который использует большую часть процессорного времени. Строка для процесса с именем devb-eide соответствует дисковым операциям ввода/вывода. Процессорное время использует также утилита make.

На второй итерации наибольшее процессорное время занимают процессы procnto и devb-eide, но из последующих итераций видно, что в дело вступает процесс htmlindex (программа, которая создает индекс ключевых слов для онлайновой документации), который отбирает на себя 96% процессорного времени. После завершения процесса htmlindex процессор оказывается занятым процессами procnto и devb-eide, пока идет запись HTML-файлов. Фактически большая часть занятости процессора приходится (включая цикл простоя) на процесс procnto.

Не беспокойтесь о том, что процесс htmlindex потребляет до 96% процессорного ресурса. На самом деле это хорошо: если запущена всего лишь одна программа, то она и должна использовать большую часть процессорного времени.

Если в системе запущено сразу несколько процессов одновременно, тогда утилита hogs оказывается весьма полезной. Выданная ей информация показывает, какие из процессов отбирают на себя большую часть процессорного времени. На основании этого можно перестроить систему приоритетов, отдавая преимущество

наиболее важным потокам. (Не забывайте, что в ЗОСРВ «Нейтрино» приоритеты являются свойством потоков, а не процессов.) Более подробно об этом см. в подразд. "Приоритеты" раздела 4.

Далее приводится несколько других рекомендаций, которые помогут улучшить производительность системы.

- для получения информации о запущенных в системе процессах можно использовать утилиту pidin. Например, можно получить список аргументов, которые были использованы при запуске процесса, состояние потоков процесса и память, которую процесс использует.
- на время реакции системы в большей степени воздействует число потоков с данным приоритетом, чем просто общее число потоков. Ключевым моментом для должного выполнения операций реального времени является установка для потоков реального времени таких приоритетов, чтобы был получен гарантированный необходимый отклик системы.
- посмотрите, действительно ли вам необходима оболочка Photon. Если нет, то можно отключить запуск оболочки при загрузке системы путем ввода команды:

```
touch /etc/system/config/nophoton
```

После этого нужно перезагрузиться. Это позволит уменьшить число запускаемых при старте системы процессов.

20.3. Уменьшение времени начальной загрузки

Вот несколько рекомендаций, которые помогут ускорить процесс начальной загрузки:

- если в вашей системе используется режим статической установки начальных параметров, то можно сделать установку и запуск драйверов самостоятельно, а не путем запуска программы распознавания устройств (enumerator);
- удалите как можно больше ненужных записей в системных файлах инициализации и в файлах образа ОС.

Более подробно об этом см. раздел 8.

20.4. Файловые системы и драйверы блочного ввода/вывода

Здесь приведены основные этапы для улучшения работоспособности файловой системы и драйверов блочного ввода/вывода:

Используйте ключи запуска для оптимизации работы дискового оборудования и драйверов. Это особенно важно для целевых устройств, реализованных в архитектурах, отличных от x86, и без жестких дисков (например, с Microdrive, Compact Flash). Отказ от использования самых быстрых из доступных режимов DMA (или даже переход на режим PIO) может до 10 раз замедлить скорость работы. Более подробно об этом см. раздел 15.

Оптимизируйте работу файловой системы, используя соответствующие ключи:

- определите, каким образом вы хотели бы сбалансировать параметры отказоустойчивости и производительности (см. далее);
- сконцентрируйте внимания на параметрах ключей cache и vnode; остальные масштабные параметры сводятся к этим двум;
- размер кэша по умолчанию устанавливается равным 15% от всего объема ОЗУ, но не менее 10 Мбайт; иногда при интенсивной работе системы этого значения может быть недостаточно;
- используйте ключ commit (глобально или в составе операции монтирования) для того, чтобы включить принудительно или отключить режим синхронной записи;
- рассмотрите возможность использования для хранения временных файлов RAM-диска (диска в оперативной памяти), например, /tmp.

Оптимизируйте код приложения в соответствии с приведенными далее рекомендациями:

- выполняйте чтение и запись большими порциями (оптимальным является размер блока в 16—32 Кбайт);

- выполняйте чтение и запись сразу нескольких секторов диска на границах блоков (обычно 512 байт, но это значение можно во время исполнения переопределить с помощью функций stat() или statvfs());
- старайтесь не использовать стандартных функций ввода/вывода, где это возможно. Применяйте функции open(), read() и write() вместо fopen(), fread() и fwrite(). В функциях семейства f* используется дополнительный уровень буферизации. Значение размера буфера по умолчанию задается константой BUFSIZ. Для задания другого значения размера буфера можно использовать функцию setvbuf();
- заранее устанавливайте размеры файлов, если вы знаете их предельный размер;
- используйте прямой ввод/вывод (режим DMA в пространство пользователя).
- используйте имена файлов не длиннее 16 символов. Если вы следуете этому правилу, то в файловой системе не будет использоваться файл .inodes, поэтому не будут использоваться какие-либо межблочные ссылки. Соответственно, будет на единицу меньше число записей на диск, а следовательно, будет меньше шансов повреждения файла во время записи при пропадании питания. Длинные имена файлов (более 48 символов) особенно сильно замедляют работу системы;
- при работе с утилитой dinit используйте ключ -i для предварительного увеличения размера файла .inodes. Это исключит необходимость манипулирования метаданными в окне исполнения во время потенциально возможного пропадания питания;
- работа с большими каталогами происходит медленнее, чем с маленькими, потому что в файловой системе используется линейный поиск.

Производительность и отказоустойчивость

При создании или конфигурировании файловой системы приходится находить баланс между характеристиками производительности и отказоустойчивости.

- в понятие отказоустойчивости включаются синхронизации пользовательских операций с реализацией этих операций, прежде чем пользователю будет дан ответ об успешном их выполнении.

Например, создание нового файла — с помощью функции create() — может потребовать выполнения всех необходимых циклов физической записи на диск, включая добавление нового имени файла в каталог файловой системы на диске, и только после этого клиент получает сигнал об успешном завершении операции.

- с точки зрения производительности, возможно, потребуется отделить стадию фактической реализации операции от выдачи ответа об ее окончании.

Например, при записи данных в файл — с помощью функции write() — может потребоваться немедленная выдача ответа клиенту, тогда как данные будут сохраняться в кэш-памяти с целью их объединения с другими записываемыми на диск данными, чтобы сформировать большой непрерывный блок для однократного последовательного доступа к диску. При этом существует опасность потери данных в случае отказа электропитания.

Поэтому вам нужно принять решение о том, как найти компромисс между удовлетворением требованиям отказоустойчивости и производительности в зависимости от конкретной инсталляции системы, ожидаемых от нее результатов и характеристик.

Обновление метаданных

Метаданными называются данные о данных или все служебные данные и атрибуты, включенные в блок сохраняемых данных пользователя. К ним относится, например, имя файла, используемые физические блоки, временные отметки об изменении или доступе к данным и т. д.

Самой дорогостоящей операцией для файловой системы является обновление метаданных. Это обусловлено двумя причинами:

- метаданные обычно размещаются на других цилиндрах диска по отношению к основным данным, и они даже оказываются отделенными друг от

друга (битовые маски, индексные дескрипторы, записи для каталогов), а следовательно, при работе с ними возникают задержки поиска;

- степень срочности записи метаданных обычно выше, чем обычных пользовательских данных (потому что от метаданных зависит целостность структуры файловой системы), а следовательно, на них влияют задержки при передаче.

Почти все операции в файловой системе (даже чтение файла, если не задан ключ noatime, см. описание функции io-blk.so в руководстве "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01) требуют, в той или иной степени, обновления метаданных.

Порядок обновления метаданных

Некоторые операции в файловой системе воздействуют сразу на несколько блоков диска. Например, рассмотрим ситуацию создания или удаления файла. В большинстве файловых систем имя файла (или ссылка на него, link) отделяется от фактических атрибутов файла (блок индексных дескрипторов, inode). Это согласуется с концепцией POSIX о жестких ссылках (hard links), множественных именах для одного и того же файла.

Обычно индексные дескрипторы (inodes) занимают на диске фиксированное место (файл .inodes для fs-qnx4.so или в заголовке группы каждого цилиндра для fs-ext2.so).

Создание файла с новым именем требует, таким образом, выделения свободной записи в блоке индексных дескрипторов, заполнение ее параметрами, касающимися нового файла, и помещения имени файла в соответствующий каталог. При удалении файла его имя удаляется из родительского каталога, а запись в блоке индексных дескрипторов помечается как свободная.

Эти операции должны выполняться именно в указанном порядке, чтобы предотвратить повреждение данных, которое может произойти при сбое питания между двумя записями. Обратите внимание на то, что при создании файла место для записи в блоке индексных дескрипторов должно выделяться до того, как

присваивается имя. В этом случае возможное разрушение при сбое питания коснется лишь выделенной записи индексных дескрипторов, которая не связана ни с каким именем (происходит утечка дискового пространства, которая позже может быть выявлена и исправлена при операции проверки файловой системы). Если выполнять операции в другом порядке, и произойдет сбой питания, то может появиться имя, которое ссылается на устаревшую или недействительную запись в блоке индексных дескрипторов. Это даст нераспознаваемую ошибку. Похожие аргументы применимы и к случаю смены порядка выполнения операций при удалении.

Для традиционных файловых систем единственный способ упорядочения операций записи состоит в синхронном выполнении первой записи (или, что более типично, всех записей, кроме последней, для последовательной записи нескольких блоков). Под синхронностью понимается немедленная запись с ожиданием полного завершения операций ввода/вывода, прежде чем будет продолжена дальнейшая работа. Синхронная запись – процедура дорогостоящая, потому что в нее входит позиционирование головок, прерывание работы любой активной потоковой передачи данных на диск и блокирование исполняемых потоков до полного окончания записи на диск. Потенциально это соответствует простою в течение миллисекунд.

Пропускная способность

Другим ключевым моментом является производительность последовательного доступа к файлу, или поточная пропускная способность (raw throughput), когда в файл записывается большой блок данных (или читается весь файл целиком). В самой файловой системе может быть обнаружен такой тип последовательного доступа, и будет предпринята попытка оптимизировать использование диска путем выполнения следующих действий:

- упреждающее чтение, при этом осуществляется доступ к вскоре ожидаемым новым данным, пока пользователь обрабатывает текущие данные;
- запаздывающая (отложенная) запись, при которой большое количество подготовленных для записи данных объединяются в единый непрерывный блок.

Наиболее эффективным способом высокопроизводительного доступа к диску является использование стандартных процедур POSIX, которые работают с дескрипторами файлов – open(), read() и write(), – потому что при этом осуществляется прямой доступ к файловой системе без вмешательства библиотеки libc.

Если вас волнует проблема производительности, то мы не рекомендуем использовать функции стандартного ввода/вывода (<stdio.h>), которые работают с переменными FILE, потому что в них вводится еще один слой программного кода и слой буферизации. Кроме того, по умолчанию размер буфера BUFSIZ установлен равным 1 Кбайт, поэтому весь доступ к диску ограничен размерами этого буфера, что приводит к большим издержкам в производительности за счет рассылки дополнительных сообщений и переключения контекстов.

Правда, есть несколько ситуаций, когда полезно использовать функции стандартного ввода/вывода, например, при построчной или посимвольной обработке текстового файла. В этом случае буфер размером 1 Кбайт, предоставляемый средствами стандартного ввода/вывода, существенно сокращает число посылаемых в файловую систему сообщений. Для улучшения производительности можно использовать функции:

- setvbuf() – для увеличения размера буфера;
- fileno() – для прямого доступа к дескриптору файла и для обхода процедуры буферизации во время работы с критическими с точки зрения производительности секциями программы.

Оптимизировать производительность можно, организовав доступ к диску порциями удобного размера. Этот размер должен быть достаточно большим для минимизации издержек на переключение контекстов и рассылку сообщений в ЗОСРВ «Нейтрино», но не настолько большим, чтобы превысить предельные возможности драйверов для блочных операций или повысить издержки на рассылку сообщений. Оптимальным является использование значения 32 Кбайт.

Иногда приходится выполнять доступ к файлу на границах блоков для всего множества секторов диска. Поскольку самой маленькой единицей доступа к

дисковому/блочному устройству является один сектор, то операции частичной записи при этом потребуют выполнения полного цикла чтения/модификации/записи. Получить значение оптимального размера для операций ввода/вывода можно, вызвав функцию statvfs(), хотя для большинства дисков принимается значение 512 байт/сектор.

И наконец, для ситуаций, требующих очень высокой производительности (например, потоковое видео), можно обойти все процедуры буферизации файловой системы и воспользоваться прямым доступом в память (DMA) между областью данных пользователя и диском. Но в этом случае надо принимать в расчет следующие факторы:

- такой вид доступа должен поддерживаться диском и дисковым драйвером;
- не следует ожидать никакой согласованности между непосредственно пересылаемыми данными и любыми данными в буферном кэше файловой системы;
- будет игнорироваться некоторая семантика интерфейса POSIX (например, файловый доступ или обновление времени модификации файла).

В настоящее время мы рекомендуем использовать режим DMA лишь в случае крайней необходимости. Не все дисковые драйверы корректно поддерживают этот режим, поэтому не существует средств отправки запроса дисковому драйверу для задания требований к его интерфейсу с целью обеспечения безопасной работы в режиме DMA. Неопытные пользователи могут иметь большие проблемы при переходе в такой режим!

В некоторых случаях, когда вы знаете полный размер конечного файла с данными, имеет смысл попробовать сразу установить его предварительный размер, а не передавать эту функцию файловой системе, которая автоматически наращивает размер файла по мере его записи. Это даст возможность файловой системе выдать один явный запрос на выделение памяти вместо нескольких "порционных" запросов. В некоторых файловых системах такая возможность существует, так что файл можно разместить более оптимально и непрерывно. Кроме того, уменьшится число обновлений метаданных, что происходит во время

фазы записи, а значит, увеличится производительность системы при записи, т. к. последовательный поток не будет разбиваться на фрагменты.

Для продления размера файла существует POSIX-функция ftruncate(). При стандартном использовании функции требуется, чтобы новое пространство данных было сначала сделано нулевым, подразумевая, что файл эффективно записывается дважды. Поэтому такой способ годится для случая, когда вы можете подготовить файл во время начальной фазы, когда вопрос с производительностью некритичен. Для продления размера файла без начального обнуления зоны данных существует и другая функция, devctl(), не входящая в состав интерфейса POSIX. Эта функция обеспечивает получение описанных выше преимуществ без затрат на стирание содержимого (см. параметр DCMD_FSYS_PREGROW_FILE в файле <sys/dcmd_blk.h>).

Конфигурация

Управлять балансом между производительностью и отказоустойчивостью можно либо глобально, либо через файлы конфигурации. Далее приведены два варианта такого управления.

- можно определить бит O_SYNC при открытии файла. Это означает, что все операции ввода/вывода с этим файлом (как с данными, так и с метаданными) будут выполняться синхронно.

Функции fsync() и sync() дают возможность при отложенной записи сбрасывать на диск содержимое кэша файловой системы по требованию. В противном случае любые подготовленные для записи данные передаются из кэша на диск под управлением глобального параметра blk delwri= (значение по умолчанию равно 2 сек, см. описание файла io-blk.so в руководстве "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01).

- управлять глобальным конфигурированием можно с помощью параметра commit= либо в файле io-blk.so, чтобы применить настройку ко всем файловым системам, либо через команду mount для воздействия на отдельный экземпляр монтируемой файловой системы. В качестве уровней воздействия используются

значения none, low, medium и high. Каждое из значений определяет синхронность или асинхронность записи метаданных или даже запись с задержкой по времени.

Примечание. При любом уровне, который менее отказоустойчив в сравнении с уровнем, задаваемым по умолчанию (medium), файловая система не будет гарантировать того же уровня целостности данных после сбоя из-за потери электропитания, поскольку в этом случае многоблочные обновления данных могут быть выполнены некорректно.

В следующих разделах иллюстрируется влияние на производительность различных конфигураций.

Влияние значения параметра commit при блочных операциях ввода/вывода

В табл. 20.1 показано, как значение параметра commit влияет на время создания и удаления файла при работе на компьютере x86 PIII-450 с диском UDMA-2 EIDE, функционирующем под управлением файловой системы QNX 4. Числа в таблице означают количество успешно созданных и удаленных за 1 сек файлов размером 0 Кбайт.

Таблица 20.1

Уровень commit	**Создано файлов**	**Удалено файлов**
high	866	1221
medium	1030	2703
low	1211	2710
none	1407	2718

Обратите внимание, что при значении commit=high все операции записи на диск выполняются синхронно, поэтому требуются значительные затраты на обновление записей каталогов и на работу POSIX-функции mtime в родительском каталоге. При значении commit=none все операции записи на диск оказываются отложенными (задержанными) в кэше. Поэтому несколько файлов могут быть созданы/удалены в блоке оперативной памяти без необходимости доступа к физическому диску (естественно, любой сбой электропитания приведет к потере этих файлов после перезапуска системы).

Размер буфера ввода/вывода

В табл. 20.2 показано, как размер буфера ввода/вывода влияет на последовательный доступ к файлу на компьютере x86 PIII-725 с диском UDMA-4 EIDE, работающем под управлением файловой системы QNX 4. Числа в таблице соответствуют скорости передачи данных в мегабайтах в секунду при записи и чтении файла размером 256 Мбайт.

Таблица 20.2

Размер буфера, Кбайт	Запись	Чтение
1	14	16
2	16	19
4	17	24
8	18	30
16	18	35
32	19	36
64	18	36
128	17	37

Обратите внимание, что скорость последовательного чтения данных при правильно выбранном размере буфера может удваиваться. Это происходит из-за того, что уменьшаются временные затраты на переключение контекстов и передачу сообщений. Заметьте, что чтение файла размером 256 Мбайт порциями по 1 Кбайт требует отправки 262 144 сообщений IO_READ, тогда как при размере буфера (порции) в 16 Кбайт требуется только 16 384 таких сообщений, 1/16 от совсем не маленького перерасхода.

Производительность при записи не показывает такого впечатляющего результата, потому что по умолчанию данные помещаются в буферный кэш отложенной записи и записываются большими непрерывными блоками под управлением таймера. Разница почувствуется при использовании бита O_SYNC. Граничным фактором в данном случае является необходимость периодического синхронного обновления битовой карты и индексных дескрипторов при распределении памяти по мере роста размера файла (см. далее конкретный пример или случай с перезаписью уже распределенного файла).

Двойное буферирование

В табл. 20.3 показано влияние двойного буферирования при работе стандартной библиотеки ввода/вывода на компьютере x86 PIII-725 с диском UDMA-4 EIDE, работающем под управлением файловой системы QNX 4. Числа в таблице соответствуют скорости передачи данных в мегабайтах в секунду при записи и чтении файла размером 256 Мбайт с размером буфера ввода/вывода 8 Кбайт.

Таблица 20.3

Сценарий	**Запись**	**Чтение**
Дескриптор файла	18	31
Стандартный ввод/вывод	13	16
setvbuf()	17	30

Здесь можно увидеть влияние задаваемого по умолчанию размера буфера (BUFSIZ, или 1 Кбайт) при стандартном вводе/выводе. Если поступает запрос на передачу данных размером 8 Кбайт, то это происходит путем выполнения 8 операций над блоками размером 1 Кбайт. Обратите внимание, как работа стандартного ввода/вывода согласуется с приведенным ранее тестом (см. подразд. "Размер буфера ввода/вывода" ранее в этом разделе) для значения размера 1 Кбайт, а случай с использованием дескриптора файла дает те же результаты, что и случай с размером буфера 8 Кбайт.

Когда используется функция setvbuf() для увеличения размера блока буфера для стандартного ввода/вывода до значения 8 Кбайт, то результаты сразу приближаются к оптимальному случаю использования дескриптора файла (маленькая разница возникает из-за усложнения программного кода и дополнительного использования функции memcpy() при передаче данных пользователя во внутренний стандартный файловый буфер ввода/вывода).

Сравнение функций, использующих файловый дескриптор, со стандартными функциями ввода/вывода

Вот еще один пример (табл. 20.4), где сравнивается доступ с использованием дескриптора файла и стандартный ввод/вывод на компьютере x86 PIII-725 с диском UDMA-4 EIDE, работающем под управлением файловой системы QNX 4. Числа в таблице соответствуют скорости передачи данных в мегабайтах в секунду при записи и чтении файла размером 256 Мбайт и при использовании файлового дескриптора (fd) и стандартного ввода/вывода (stdio).

Таблица 20.4

Размер буфера	**Запись (fd)**	**Чтение (fd)**	**Запись (stdio)**	**Чтение (stdio)**
32	1,5	1,7	10,9	12,7
64	2,8	3,1	11,7	14,3
128	5,0	5,6	12,0	15,1
256	8,0	9,0	12,4	15,2
512	10,8	12,9	13,2	16,0
1024	14,1	16,9	13,1	16,3
2048	16,1	20,6	13,2	16,5
4096	17,1	24,0	13,9	16,5
8192	18,3	31,4	14,0	16,4
16 384	18,1	37,3	14,3	16,4

Обратите внимание, насколько доступ через функцию read() чувствителен к размеру буфера. Это связано с тем, что при каждом вызове функции read() отправляется сообщение _IO_READ и при этом происходит переключение контекста и отправка сообщения файловой системе. Если каждый раз передается малый объем данных, то непроизводительные издержки операционной системы становятся ощутимыми.

Поскольку при стандартном вводе/выводе с помощью функции fread() используется внутренний буфер размером 1 Кбайт, то число отправляемых сообщений _IO_READ оказывается постоянным независимо от размера пользовательского буфера. Поэтому пропускная способность для всех случаев напоминает ситуацию с доступом через дескриптор файла с размером буфера 1 Кбайт (наблюдается небольшая деградация при меньших значениях размера буфера из-за увеличивающегося количества вызовов библиотеки libc). Таким

образом, вы должны на основании этих примеров выбрать в вашем приложении предпочтительный вариант использования процедур ввода/вывода при доступе к файлам.

Предварительное задание размера файла

В этом примере иллюстрируется, какое влияние оказывает предварительное задание размера файла с данными на компьютере x86 PIII-725 с диском UDMA-4 EIDE, работающем под управлением файловой системы QNX 4. Числа в табл. 20.5 соответствуют времени в миллисекундах, которое необходимо для создания и записи файла размером 256 Мбайт при размере буфера ввода/вывода 8 Кбайт.

Таблица 20.5

Сценарий	**Создание**	**Запись**	**Всего**
write()	0	15073	15 073 (15 сек)
ftruncate()	13908	8510	22 418 (22 сек)
devctl()	55	8479	8534 (8,5 сек)

Обратите внимание, насколько замедляет работу постепенное наращивание длины файла в результате вызовов функции по сравнению с установлением размера путем однократного вызова функции ftruncate(). В последнем случае файловая система должна только один раз выделить большой/непрерывный блок экстентов данных и обновить атрибуты метаданных индексных дескрипторов. Отметьте также, что время перезаписи уже выделенных блоков данных значительно меньше времени при динамическом выделении блоков (последовательные записи не прерываются необходимым периодическим синхронным обновлением битовой карты).

Несмотря на то, что полное время на предварительную установку размера файла и на перезапись оказывается больше, чем для случая с постепенным наращиванием размера, сам этап предварительной установки может быть осуществлен при выполнении фазы инициализации, где временные затраты не очень критичны. Зато впоследствии повышается производительность во время собственно записи файла.

Оптимальным вариантом будет использование предварительной установки размера файла без начального обнуления зоны данных (что делает функция devctl()), после чего в выделенную область производится перезапись реальных данных.

Точная настройка USB устройств хранения данных

При существовании в конфигурации хоста больших файлов (например, музыкальных) на USB устройстве хранения данных, необходимо убедиться, что конфигурация обеспечивает достаточное количество оперативной памяти для опережающего чтения данных больших файлов, таких как MP3. Изменение конфигурации можно произвести корректировкой значений cache и vnode, которые devb-umass передает io-blk.so с помощью опции blk.

Обычная начальная конфигурация для опции blk: `cache=512k,vnode=256`.

Однако, следует протестировать действие ключей в данной конфигурации, и только после этого подбирать значения для оптимальной работы.

20.5. Насколько маленькой может быть ваша система?

Наилучшим способом уменьшения размера системы является использование нашей среды разработки IDE для создания образа ОС. Рабочая среда "системный компоновщик" (System Builder) включает в себя инструмент Dietician (Диетолог), который помогает "похудеть" библиотекам, включаемым в образ системы. Более подробнее об этом см. в руководстве "IDE User’s Guide", а также в руководстве "Building Embedded Systems".

21. Лимиты системы

21.1. О каких лимитах идет речь

ЗОСРВ «Нейтрино» основана на микроядре, поэтому те ограничения, которые в других операционных системах были бы связаны с ядром, в данном случае будут зависеть только от того или иного администратора, реализующего соответствующую службу в ЗОСРВ «Нейтрино». Особенно это относится к файловым системам и тем случаям, когда одновременно применяется несколько разных файловых систем.

Многие ресурсы зависят от объема доступной памяти. Другие лимиты зависят от целевой системы. Например, виртуальное адресное пространство для процесса может составлять от 32 Мбайт на архитектуре ARM до 3,5 Гбайт на архитектуре x86.

Некоторые ограничения появляются в результате комплексного взаимодействия нескольких факторов. Если просто привести простые и очевидные значения граничных параметров без указания их взаимодействий, то это может ввести в заблуждение. Описание же всех взаимодействий может быть запутанным. Ключевым моментом, который нужно иметь в виду при чтении данного раздела, является то, что с конкретным ограничением может быть связано много дополнительных факторов.

21.2. Конфигурационные лимиты

Когда вы пытаетесь выяснить ограничения для системы, то нужно рассмотреть конфигурационные лимиты (configurable limit) — специальные переменные с атрибутами "только чтение", в которых сохраняется системная информация.

Примечание. В ЗОСРВ «Нейтрино» поддерживаются также конфигурационные строки (configurable string), которые похожи на

переменные окружения и часто используются совместно с ними. Подробнее об этом см. раздел 9.

Получить значения конфигурационных лимитов или строк можно с помощью утилиты getconf (POSIX). Поскольку утилита getconf относится к интерфейсу POSIX, то ее можно использовать в сценариях вместо жестко запрограммированных в ЗОСРВ «Нейтрино» значений лимитов, что позволяет адаптировать сценарии к другим конфигурационным параметрам POSIX.

Некоторые конфигурационные лимиты связаны с путями. Их имена начинаются с _PC_. Когда вы запрашиваете значения этих параметров, то при запросе должен быть указан путь (см. подразд. "Лимиты файловой системы" далее в этом разделе). Например, для получения максимальной длины для имени файла нужно ввести:

getconf _PC_NAME_MAX имя_для_пути

Другие ограничения связаны с системой в целом. Имена этих параметров начинаются с _SC_, и при их извлечении указание пути не требуется. Например, для получения максимального числа файлов, которые может открывать процесс, нужно ввести команду:

getconf _SC_OPEN_MAX

Вообще говоря, значения конфигурационных лимитов вы изменять не можете. Они называются "конфигурируемыми", потому что их значения могут изменяться системой.

В библиотеках ЗОСРВ «Нейтрино» имеется несколько различных функций, которые можно использовать в программах для работы со значениями конфигурационных лимитов, связанных с конфигурацией:

- pathconf() – предоставляются значения конфигурационных лимитов, имеющих отношение к пути;
- sysconf() – предоставляются значения конфигурационных лимитов общесистемного уровня;

- setrlimit() – производится изменение значений некоторых конфигурационных лимитов. Например, эта функция может использоваться для ограничения числа файлов, которые разрешается открывать процессу, но при этом данная величина зависит также от значения, указанного в ключе -F утилиты procnto.

21.3. Лимиты файловой системы

Файловые системы в ЗОСРВ «Нейтрино» не являются частью ядра или базовых компонентов операционной системы. Файловые системы представляют собой отдельно загружаемые процессы или библиотеки.

Под этим понимается следующее:

- для файловых систем ЗОСРВ «Нейтрино» не устанавливаются какие-либо общие правила или ограничения – ограничения определяются исключительно конкретным типом файловой системы и процессом, который предоставляет доступ к файловой системе;
- вы можете создать собственный процесс файловой системы или слой, который может почти прозрачно заместить или изменить многие из базовых параметров.

В последующих разделах будут приведены ограничения для поддерживаемых файловых систем. Имейте в виду следующее:

- длины имен файлов и путей приводятся в байтах, а не в символах;
- многие из поддерживаемых в ЗОСРВ «Нейтрино» файловых систем используют 32-битовый формат, что накладывает ограничение на размер файла в 2 Гбайт минус 1 байт; это, в свою очередь, ограничивает размер каталога, потому что на размер файла, в котором сохраняется информация о каталоге, распространяется то же ограничение на длину.

Запрос лимитов файловой системы

Вы можете узнать конфигурационные лимиты, имеющие отношение к путевому имени, чтобы выяснить свойства и лимиты, присущие конкретной

файловой системе. Далее приводятся обозначения и описание переменных, задающих лимиты:

- _PC_LINK_MAX — максимальное количество ссылок на файл;
- _PC_MAX_CANON — максимальное число байтов для канонического буфера вода терминала (отформатированная строка в буфере редактирования);
- _PC_MAX_INPUT — максимальное число байтов для входного буфера необработанных данных терминала;
- _PC_NAME_MAX — максимальное число байтов в имени файла (исключая завершающий символ null);
- _PC_PATH_MAX — максимальное число байтов в пути (исключая завершающий символ null);
- _PC_PIPE_BUF — максимальное число байтов, которое может быть записано атомарно в канальный буфер (pipe);
- _PC_CHOWN_RESTRICTED — если константа определена (т. е. не равна –1), то это означает, что использование функции chown() ограничено процессами с соответствующими привилегиями и изменениями группового ID файла на эффективный групповой ID процесса или на один из его дополнительных групповых ID;
- _PC_NO_TRUNC — если константа определена (т. е. не равна –1), то это означает, что при использовании компонентов имени для пути длиннее, чем значение, заданное константой _PC_NAME_MAX, приведет к сообщению об ошибке;
- _PC_VDISABLE — если константа определена (т. е. не равна –1), то ее значение соответствует специальному символу, который может использоваться для индивидуального отключения специальных управляющих символов в структуре termios.

Дополнительная информация приведена в разд. "Конфигурационные лимиты" ранее в этом разделе.

Файловая система QNX 4

Далее приведены лимиты для файловых систем QNX 4:

- длина имени файла — 48 или 505 байт, если перед монтированием существует файл .longfilenames (подробнее об этом см. в описании файловой системы QNX 4 в подразд. "Имена файлов" раздела 11);
- длина имени для пути — 1024 байта;
- размер файла — 2 Гбайт минус 1 байт);
- размер каталога — не существует никакого практического ограничения, хотя файлы, которые используются для управления содержимым каталога, имеют указанное стандартное ограничение на длину 2 Гбайт минус 1 байт, что соответствует возможности разместить в одном каталоге приблизительно 33 млн файлов. Вы вряд ли захотите иметь такое количество файлов в каталоге, потому что сканирование каталога происходит линейно, и большое число файлов сильно замедляет работу;
- размер файловой системы — 2 Гбайт*512; ограничение накладывается драйвером диска;
- размер диска — 264 байт; ограничение накладывается драйвером диска.

Файловая система Power-Safe (fs-qnx6.so)

Лимиты для файловой системы Power-Safe (поддерживаемой fs-qnx6.so):

- физический сектор диска – 32-бита (2 Тб), используется devb API.
- логический блок файловой системы – 512, 1024, 2048 или 4096 (устанавливается при начальной компоновке файловой системы).
- максимальная длина имени файла – 510 байт (UTF-8). Если имя файла в длину меньше 28 байт, оно сохраняется в записях каталога; если имя длиннее, оно сохраняется во внешнем файле, а записи каталога ссылаются на него.
- максимальный размер файла – 64-битная адресация. С размером блока 1Кб (по умолчанию), можно установить 256 блоковых указателей на блок, таким образом, что файл 16 × 256 × 1 Кб (4 Мб) запрашивает 1 уровень непрямых указателей. Если файл больше, то потребуется два уровня (т.е. 16 блоков 256 указателей блоков, содержащих другие 256 указателей на блоки), что дает максимальный размер файла 1 Гб. Для трех уровней непрямых указателей

максимальный размер файла будет 256 Гб. При размере блока 2 Кб, каждый блок поддерживает 512 указателей и соответственное масштабирование.

Файловая система Ext2

Далее приведены лимиты для файловых систем Linux Ext2:

- длина имени файла — 255 байт;
- длина имени для пути — 1024 байт;
- размер файла — 2 Гбайт минус 1 байт;
- размер каталога — 2 Гбайт минус 1; каталоги представляют собой файлы, в которых в качестве данных содержатся индексные дескрипторы и информация об именах файлов;
- размер файловой системы — 2 Гбайт*512;
- размер диска — 264 байт; ограничение накладывается драйвером диска.

Файловая система DOS FAT12/16/32

Далее приведены лимиты для файловых систем DOS FAT12/16/32:

- длина имени файла — 255 байт;
- длина имени для пути — 260 байт;
- размер файла — 4 Гбайт минус 1; используется файловая система с 32-битовым форматом;
- размер каталога — зависит от типа файловой системы:
- корневой каталог FAT12/16 имеет специальную структуру в том смысле, что его размер фиксирован и не может быть увеличен; его размер выбирается при форматировании, и он обычно рассчитан на 512 записей; для FAT32 такого ограничения нет;
- лимит каталогов FAT (для DOS-совместимых) – 64K записей.
- для длинных имен (не соответствующих стандарту 8.3) может потребоваться использование нескольких записей, что уменьшает возможный размер каталога;
- размер файловой системы – зависит от формата FAT:

- для FAT12 — 4084 кластера (максимальный размер кластера 32 Кбайт, следовательно, граничный размер равен 128 Мбайт);
- для FAT16 — 65 524 кластера (т. е. граничный размер равен 2 Гбайт);
- для FAT32 можно получить доступ к 268 435 444 кластерам (что соответствует граничному размеру в 8 Тбайт);
- размер диска — ограничен дисковым драйвером и файлом io-blk.

В перечисленных файловых системах не поддерживается установка прав доступа к файлам, но эту функцию можно эмулировать.

Файловая система CD-ROM (ISO 9660)

Далее приведены лимиты для файловых систем CD-ROM (ISO 9660):

- длина имени файла — 32 байта для базового стандарта ISO 9660, 128 байт для стандарта Joliet, 255 байт для стандарта Rockridge;
- длина имени для пути — 1024 байта;
- размер диска — в файловой системе используется 32-битовый формат (максимальный размер 2 Гбайт минус 1). Не разрешается создание чего-либо через использование файла fs-cd.so, он имеет атрибут "только чтение". Любые ограничения могут накладываться инструментам для создания образа (которые, надо надеяться, относятся к подмножеству стандарта ISO 9660). Размер диска также ограничен дисковым драйвером и файлом io-blk.

Примечание. Взамен fs-cd.so лучше использовать fs-udf.so, поддерживающий файловую систему ISO-9660 в дополнение к UDF. Дополнительную информацию о лимитах UDF, см. в «Файловая система UDF» далее в этом разделе.

Файловая система NFS2 и NFS3

Далее приведены лимиты для файловых систем NFS2 и NFS3:

- длина имени файла — 255 байт;
- длина имени для пути — 1024 байта;
- размер файла — 2 Гбайт минус 1; ограничение для 32-битовых файловых систем;

- размер каталога, размер файловой системы и размер диска — зависит от сервера; ограничение для 32-битовых файловых систем.

Файловая система CIFS

Далее приведены лимиты для файловых систем CIFS:

- длина имени файла — 255 байт;
- длина имени для пути — 1024 байта;
- размер файла — 2 Гбайт минус 1; ограничение для 32-битовых файловых систем;
- размер каталога, размер файловой системы и размер диска — ограничение для 32-битовых файловых систем.

Файловая система CIFS не поддерживает использование функций chmod или chown.

Встраиваемая файловая система

Далее приведены лимиты для встраиваемых файловых систем (для флэш-памяти):

- длина имени файла — 255 байт;
- длина имени для пути — 1024 байта;
- размер файла, размер файловой системы и размер диска — 2 Гбайт минус 1;
- размер каталога — ограничивается доступным пространством.

Записи для файлов, каталогов и файловых экстентов представляют собой связанные списки. Чем длиннее список, тем больше времени требуется для поиска места, где нужно присоединить данные или получить связанную с файлом статистику.

Встраиваемая транзакционная файловая система (Embedded Transaction filesystem (ETFS))

Лимиты для ETFS:

- длина имени файла – 91 байт.

- длина имени для пути – 1024 байт.
- размер файла – 2 GB − 1; 32-бит – лимит файловой системы.
- максимальное количество файлов – 32768 (15 бит).
- максимальное количество файлов по умолчанию – 4096.
- максимальный размер кластера – 4096.
- максимальный размер файловой системы – 64 Гб.

Для флеш-памяти NAND существуют дополнительные лимиты:

- поддерживается только секция одного уровня (single-level cell (SLC)) NAND flash;
- максимальный размер файловой системы 4 Гб;
- защита резервной области ECC поддерживается только 2 Кб и 4 Кб страницами памяти NAND;
- программное обеспечение ECC поддерживает только 1-битную коррекцию ошибок для каждого 256-байтного буфера;
- поддерживается только флеш-память NAND с размерами страниц памяти 512, 2048 и 4096 байт.

Для ETFS на NAND необходимо выполнить программную кодировку 1-битной коррекции ошибок (ECC) для данных в резервной области. Поддерживаемые конфигурации доступны для:

- 2 Кб страниц памяти устройства памяти NAND;
- 4 Кб страниц памяти устройства памяти NAND.

Установлено, что резервная область получает значение ECC от devio_postcluster(), и после этого записывает его на устройство памяти NAND. Для определения соответсвующего размера значения ECC необходимо использовать следующее:

- для 512 NAND, не доступно;
- для 2048 NAND, используется 64 байт ECC;
- для 4096 NAND, используется 128 байт ECC;

Для установки приоритета резервной области необходимо выполнить следующие изменения для BSP:

- для devio_readtrans() и devio_readcluster() — при чтении резервной области, в первую очередь сохраняется резервная область ECC, после этого задаются области резервной структуры 0xFF, которые запрашивают определение циклического контроля избыточности (CRC — проверка сохранности данных для устройства памяти NAND). Выполняется определение CRC, и если произошел сбой, для восстановления необходимо попробовать использовать другую резервную область значения ECC. Если резервная область ECC исправна, действия CRC игнорируются. Если резервная область ECC была скорректирована, устанавливается tacode в структуре транзакции ETFS_TRANS_ECC. Если ECC не было скорректирована, устанавливается tacode для ETFS_TRANS_DATAERR;
- для devio_postcluster() — после определения CRC и ECC для кластера данных, и определения CRC для резервной области, добавляется определение ECC для резервной области. При выполнении определения CRC, необходимо использовать 0xFF в качестве символа-заполнителя для резервной области ECC.

Файловая система UDF

Лимиты для файловой системы UDF:

- длина имени файла – 255 символов Unicode.
- длина имени для пути – 1024 байт.
- размер диска – данная файловая система использует 32-битные блоковые адреса, но сама файловая система 64-битная (> 4 Гб).

Какие-либо действия в файловой системе с использованием fs-udf.so запрещены. Данная файловая система доступна только для чтения.

Apple Macintosh HFS и HFS Plus

Лимиты для Apple Macintosh HFS (Hierarchical File System) и HFS Plus:

- длина имени файла – 31 символ MacRoman для HFS; 255 байт (Unicode) для HFS Plus.
- длина имени для пути – 1023 bytes.
- размер диска – данная файловая система использует 32-битные блоковые адреса, но сама файловая система 64-битная (> 4 Гб).

Какие-либо действия в файловой системе с использованием fs-mac.so запрещены. Данная файловая система доступна только для чтения.

Файловая система Windows NT

Лимиты для файловой системы Windows:

- длина имени файла – 255 символов.
- длина имени для пути – 1024 байт.
- размер файла – 4 GB − 1; используется формат файловой системы 64-бит.
- размер файловой системы – 264 - 1 кластеров.
- размер диска – ограничен драйвером диска и io-blk.

Данная файловая система доступна только для чтения.

21.4. Другие системные лимиты

Приведенные далее лимиты относятся ко всей системе в целом:

Процессы — максимум действующих одновременно – 4095, за исключением:

- на платформах ARMv6 (запускается procnto-v6), лимит – 255 процессов. MMU содержит восьмибитный ASID, который используется для маркирования не глобальных записей TLB. Каждому адресному пространству присвоен уникальный тэг, устанавливающий в MMU контекстный регистр контекстного переключения для определения используемого ASID. На данный момент не используется код ASID, устанавливается ограничение на 255 процессов;
- на более ранних платформах ARM лимит – 63 процесса.

В платформе ARM лимит установлен практически на количество разделенных адресных пространств; можно запустить больше процессов, если они будут происходить в разделенном адресном пространстве, благодаря vfork(), но это исключительный случай.

- пространство префикса (присоединения администратора ресурсов и т. д.) — ограничено имеющейся памятью;
- сеансы и группы процессов – 4095 (потому что на каждый сеанс или группу необходим, по крайней мере, один процесс);

- физическое адресное пространство – ограничения отсутствуют за исключением тех, что накладываются оборудованием (см. документацию по используемой элементной базе).

К каждому процессу применимы следующие лимиты:

- число потоков — 32 767;
- число таймеров — 32 767;
- приоритеты — от 0 до 255.

Приоритет с номером 0 используется для потока ожидания. По умолчанию приоритеты с номерами более 64 относятся к категории привилегированных, поэтому использовать их могут только пользователи с ID=0 (т. е. имеющие учетную запись root). Остальным пользователям доступны приоритеты от 1 до 63.

Изменить диапазон привилегированных приоритетов можно с помощью утилиты procnto при использовании ключа -P.

- выделение памяти: Поскольку реализация функции malloc() использует знаковое 32-битное целое для внутреннего представления размера, то невозможно выделить памяти больше 2 Гб. Если размер больше 2 Гб, эти функции возвращают ошибку ENOMEM:
  - asyncmsg_malloc()
  - calloc()
  - iofunc_lock_calloc()
  - malloc()
  - memalign()
  - realloc()
  - valloc()

Файловые дескрипторы

Полное число файловых дескрипторов жестко ограничено значением 32 767 для каждого процесса, но, скорее всего, ограничение будет задано с помощью ключа -F при использовании утилиты procnto или системной переменной

RLIMIT_NOFILE. Значение по умолчанию равно 1000, минимальное значение равно 100.

Примечание. Имейте в виду, что файловые дескрипторы используются сокетами, именованными семафорами, очередями сообщений, идентификаторами соединений (coids).

Для того чтобы получить установленное в текущий момент времени значение для этого параметра, используйте встроенную команду ksh, утилиту ulimit (см. в руководстве "Описание программы. Часть 1. Справочник по утилитам" КПДА.10964-01 13 01) или вызовите библиотечную процедуру getrlimit() (см. руководство по библиотекам "Library Reference").

Примитивы для синхронизации

Не существует никаких ограничений на количество мьютексов (mutex) и переменных для условий (condvars).

Нет также ограничений на количество непоименованных семафоров, но количество именованных семафоров ограничивается числом доступных дескрипторов (см. предыдущий раздел).

Лимиты для протокола TCP/IP

Число открытых соединений и сокетов ограничивается только доступной памятью и максимальным числом файловых дескрипторов на каждый процесс (см. подразд. "Файловые дескрипторы" ранее в этом разделе).

Разделяемая память

Число областей разделяемой памяти ограничено разрешенным виртуальным адресным пространством, которое зависит от архитектуры целевой системы.

Очереди сообщений

Количество очередей сообщений ограничено числом доступных файловых дескрипторов (см. разд. "Файловые дескрипторы" ранее в этом разделе).

Установленное по умолчанию максимальное число записей в очереди и установленный по умолчанию максимальный размер записи для очереди зависит от того, используете вы для реализации очереди традиционный подход (mqueue) или альтернативный (mq) (табл. 21.1).

Таблица 21.1

Атрибут	**Традиционный подход**	**Альтернативный подход**
Число записей	1024	64
Размер сообщения	4096	256

Более подробно об этом см. описание команд mqueue и mq в руководстве «Описание программы. Часть 1. Справочник по утилитам» КПДА.10964-01 13 01.

Ограничения, связанные с платформой

Таблица 21.2

Ограничение на	**x86**	**PPC**	**MIPS**	**SH-4**	**ARMv4**	**ARMv6**
Системное ОЗУ	64 Гбайт (36-битовая адресация)	64 Гбайт (36-битовая адресация)	1 Тбайт (40-битовая адресация)	512 Мбайт (29-битовая адресация)	4 Гбайт (32-битовая адресация)	512 MB (32-битовая адресация)***
ЦПУ*	8	8	8	1	1	1
Виртуальное адресное пространство**	3,5 Гбайт	3 Гбайт	2 Гбайт	2 Гбайт	32 Мбайт	2 Гбайт

Примечание. * – применяемое оборудование может определять дополнительные лимиты на количество ЦПУ. ** — приведены абсолютные максимальные лимиты для виртуального адресного пространства. Эти значения можно уменьшить, изменив параметр RLIMIT_AS с помощью функции setrlimit(). *** – 32-битная адресация устанавливает 4 Гб пространства данных, но оно не может быть все использовано для RAM системы на некоторых платформах. Некоторое количество пространства

данных зарезервировано для устройств, и некоторые платформы могут накладывать ограничения.

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц) в докум.	№ документа	Входящий № сопроводи-тельного докум. и дата	Подпись	Дата
Изм.	Изменен-ных	Заменен-ных	новых	аннули-рованных					