**СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ
АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
имени академика М. Ф. Решетнева**

**О.Н. Моргунова
В.В. Тынченко**

Операционная система FreeBSD

Вводный курс

Красноярск 2011

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Государственное образовательное учреждение
высшего профессионального образования
«Сибирский государственный аэрокосмический университет
имени академика М.Ф. Решетнева»

О. Н. Моргунова, В.В. Тынченко

Операционная система FreeBSD.

Вводный курс

Рекомендовано к изданию научно-методической комиссией факультета информатики и систем управления

Красноярск 2011

УДК 681.3.066
ББК 32.973.26
М 79

Рецензенты:
Доктор технических наук, профессор каф. Информационно-управляющих систем СибГАУ А.В. Мурыгин, кандидат технических наук, доцент каф. Экономических информационных систем и информационных технологий красноярского филиала МЭСИ Г.И. Орлов,

Моргунова, О. Н., Тынченко В.В.
М 79 Операционная система FreeBSD. Вводный курс : учеб. пособие / О. Н. Моргунова, В.В. Тынченко; Сиб. гос. аэрокосмич. ун-т. – Красноярск, 2011. – 132 с.

В учебном пособии изложены основы работы с операционной системой UNIX. В качестве представителя систем данного класса рассматривается система FreeBSD 8.2. Освещаются такие вопросы, как установка, настройка и администрирование операционной системы.

Пособие предназначено для студентов, обучающихся по направлениям 230200 «Информационные системы», 230100 «Информатика и вычислительная техника», 090106 – «Информационная безопасность телекоммуникационных систем». Оно может быть полезно широкому кругу пользователей, знакомых с основами программирования и операционной системой Windows, желающих самостоятельно познакомиться с операционной системой UNIX.

УДК 681.3.066
ББК 32.973.26

Оглавление

Введение

В настоящее время повышается спрос на специалистов, обладающих знаниями в области операционных систем (ОС). Это обусловлено, в частности, бурным развитием глобальной сети Интернет. Как вы знаете, самой массовой операционной системой является ОС Microsoft Windows, однако роль ОС UNIX неуклонно возрастает. Без этой операционной системы невозможно представить себе работу крупного Интернет-узла или информационной системы промышленного предприятия или организации. Следует сказать и о том, что все чаще операционная система UNIX (в лице таких ее представителей, как Linux и FreeBSD) рассматривается в качестве альтернативы системе Windows даже для конечных пользователей.

Операционная система UNIX имеет несколько разновидностей, среди которых можно найти как коммерческие, так и некоммерческие системы. Среди некоммерческих систем наиболее популярны ОС Linux и ОС FreeBSD. К сожалению, в рамках краткого практического учебного курса невозможно детально рассмотреть несколько операционных систем, поэтому мы вынуждены ограничиться только одной системой. Однако, к счастью, принципиальные черты всех разновидностей ОС UNIX являются сходными, поэтому, хорошо изучив одну из систем UNIX, вы сможете самостоятельно разобраться и с любой другой разновидностью этой операционной системы. Свой выбор мы остановили на ОС FreeBSD. Конечно, кто-то может сказать, что следовало выбрать для рассмотрения ОС Linux. Наверное, можно долго спорить, доказывая преимущества той или иной системы, но мы не ставим перед собой задачи сравнения операционных систем, а для того, чтобы дать представление о мире UNIX, операционная система FreeBSD вполне подходит.

Для успешного освоения предлагаемого вам практического курса от вас, уважаемый читатель, требуются знания основ языка программирования C и умение работать в среде ОС Windows на уровне пользователя.

Введем некоторые соглашения, касающиеся использования различных шрифтов для выделения тех или иных фрагментов текста. Команды, вводимые пользователем, т.е. вами, уважаемый читатель, выделяются полужирным шрифтом. Например:

```
cat file1 file2 > new_file
```

Команды часто имеют параметры, которые для наглядности представляются таким образом:

```
cp имя_исходного_файла имя_нового_файла
```

В этом случае при вводе команды вы должны вместо русских описательных параметров подставить реальные имена файлов, которые имеются на вашем компьютере.

Если вся команда не умещается на одной строке текста, то она переносится на вторую строку, но вы должны вводить ее на одной строке, например:

```
ls -l | awk '{ if ($1 ~ /^.........x/ && $1 !~ /^d/ ) print
$1, $9 }'
```

Результаты работы команд операционной системы или программ напечатаны моноширинным шрифтом, например, введя команду

```
ls -l
```

получим

```
-rw-r--r--   1 root  wheel     297 26 сен  2000 .login
drwx------   5 root  wheel     512  2 окт 23:07 .netscape
-rw-r--r--   1 root  wheel     380 19 мар  2001 .profile
-rw-------   1 root  wheel      24 22 сен 16:40 .rhosts
-rwxr-xr-x   1 root  wheel      49  8 май 17:12 Apache.start
drwxr-xr-x   3 stud  wheel     512 27 май 23:34 C++
-rw-r--r--   1 stud  wheel      23  9 апр  2001 aaa
drwxr-xr-x   4 stud   stud     512  9 окт 12:10 developments
-rw-r--r--   1 root  wheel       0 11 окт 12:15 ls.txt
drwx------   2 root  wheel     512  7 апр  2001 nsmail
-rw-r--r--   1 root  wheel  228241 11 окт 10:50 perlfunc.man
```

Тексты программ, которые приведены в учебном пособии, напечатаны также моноширинным шрифтом. Например:

```
foreach ( keys %height )
{
  # для сравнения на точное равенство числовых значений
  # используется оператор "==", на неравенство - >, <, >=, <=
  if ( $height{ $_ } == 165 )
  {
    print "Студент $_ имеет низкий рост: $height{ $_ }\n";
  }
  else
  {
    print "Студент $_ имеет высокий рост: $height{ $_ }\n";
  }
}
```

В пособии очень часто будут даваться указания ввести ту или иную команду. Это означает, что вам следует не только набрать на клавиатуре текст этой команды, но также нажать клавишу Enter для ее выполнения.

Для сокращения объема текста мы не будем повторять слова «Нажмите клавишу Enter» каждый раз.

Поскольку данное учебное пособие предназначено для тех, кто только начинает изучение операционной системы UNIX, то вам, возможно, потребуется помощь более опытного специалиста для первоначальной установки ОС FreeBSD на ваш компьютер. В главе 1 приводится краткая инструкция по установке этой операционной системы. В качестве базовой версии операционной системы принята версия 8.2. выпущенная в 2011 г.

Мы надеемся, что наше учебное пособие поможет вам получить основные знания и умения, необходимые для полноценной работы с ОС FreeBSD.

1. Установка операционной системы FreeBSD

Поскольку настоящий учебный курс рассчитан на студентов, которые уже имеют опыт работы с операционной системой Windows, то мы ограничимся лишь краткой инструкцией по установке операционной системы FreeBSD на ваш компьютер.

Как правило, на домашних компьютерах используется ОС Windows, и вряд ли целесообразно от нее отказываться. Поэтому, скорее всего, вам придется установить на ваш компьютер как Windows, так и FreeBSD.

Если на вашем компьютере уже установлена операционная система Windows, например, Windows XP, и вам не хочется ее «уничтожать», а затем переустанавливать заново, тогда вам следует подыскать одну из специальных программ, которые позволяют уменьшить размер раздела, занимаемого системой Windows, и тем самым освободить место для ОС FreeBSD. Порядок использования таких программ выходит за рамки нашего рассмотрения.

Если же на вашем компьютере еще не установлена никакая операционная система (или уже установлена ОС Windows, но вы готовы ее удалить и переустановить заново), тогда нам с вами будет проще.

Итак, приступаем к установке операционных систем. Описывая процедуру установки, мы считаем, что ваш компьютер имеет один жесткий диск. Первым рекомендуется установить FreeBSD, для Windows7 рекомендуется на диске выделить достаточно места.

1. Загрузите ваш компьютер с загрузочного CD/DVD диска FreeBSD. Для того чтобы компьютер использовал для начальной загрузки не жесткий диск, а дисковод CD/DVD дисков, необходимо указать это в утилите Setup, запустив ее перед началом загрузки компьютера. Порядок использования утилиты Setup описывается, как правило, в документации на материнскую плату компьютера.

2. При установке двух и более операционных систем на одном жестком диске необходимо разбить его на так называемые **разделы**. Поскольку все инструкции в процедуре установки Windows7 приведены на русском языке, то, следуя им, вы сможете все сделать правильно. Мы не будем описывать весь процесс детально, а покажем лишь стратегию работы.

3. Вам будет показана информация о существующей структуре разделов на жестком диске вашего компьютера. Если ОС Windows занимает 100 процентов объема диска (а это, вероятнее всего, так и есть), то вы должны удалить раздел, в котором располагается Windows7.

4. В начале процесса загрузки ОС FreeBSD вы увидите меню, предлагающее выбрать вариант загрузки системы и состоящее из семи пунктов. Вам следует выбрать первый пункт **Boot FreeBSD [default]**. Для этого необходимо нажать клавишу Enter либо просто подождать, пока через 10 секунд система сама не продолжит работу именно с этого пункта меню. Как мы уже говорили, описание наше будет кратким, поэтому ограничимся лишь указаниями о выполнении тех или иных действий без объяснения причин того или иного выбора.

5. После выполнения загрузки система выведет на экран список стран мира. Вам необходимо выбрать из этого списка нашу страну – **Russian Federation**. Сразу после выбора страны на экран выводится список раскладок клавиатуры, в котором по умолчанию курсор устанавливается на строке **Russia KOI8-R**. Это как раз то значение, которое нам нужно выбрать.

ПРИМЕЧАНИЕ. Для перемещения по всем разделам меню и списков, предлагаемых в процессе установки системы, и осуществления выбора тех или иных значений используется несколько клавишей. Выбор можно производить с помощью клавиши Enter или клавиши «пробел». Клавиша Tab служит для перемещения между верхней и нижней областями меню. Для возвращения в меню на один уровень вверх используйте клавишу Escape или с помощью клавиши Tab переходите к пункту **Cancel** и выбирайте его. Также в каждом меню есть пункт **Exit**, выбрав который, вы перейдете в меню верхнего уровня.

---

6. После выбора раскладки клавиатуры будет выведено главное меню процедуры установки операционной системы. Здесь предлагаются Различные варианты установки. Рекомендуем вам выбрать вариант установки **Custom**. Это процедура инсталляции для опытных пользователей. Если вы пока что таковым не являетесь, то это не беда: установив ОС FreeBSD несколько раз, вы приобретете необходимый опыт. Вы можете воспользоваться также и процедурой **Standard** (в этом случае в процессе установки выводится больше справочной и руководящей информации) или **Express** (эта процедура предназначена также для экспертов). Но поскольку вы выбрали строку **Custom**, то вам предлагается меню, включающее следующие пункты: **Exit, Options, Partition, Label, Distributions, Media, Commit**. Ваша задача – поочередно выбрать каждый из пунктов **Partition, Label, Distributions, Media, Commit** и выполнить необходимые действия.

7. Пункт **Exit** предназначен для выхода из процедуры установки системы, а пункт **Options** выбирать не требуется, поскольку те значения параметров, которые предлагаются в этом разделе по умолчанию, подходят для вашей установки системы. Если вы все же вошли в этот раздел, то для выхода нажмите клавишу Q.

8. Выберите пункт **Partition**. Если в вашем компьютере установлено два или более жестких диска, то с помощью клавишей управления курсором переместите указатель на требуемый диск и выберите его с помощью клавиши «пробел». В этом случае клавишу Enter для выбора диска использовать нельзя. Если же вы использовали именно ее, то сообщения об

ошибке выведено не будет, но впоследствии при выборе пункта меню **Commit** будет выдано сообщение о том, что процедуре установки не удается смонтировать корневую файловую систему. Будет выведен еще ряд сообщений об ошибках, и установка операционной системы завершится неудачно.

После выбора требуемого жесткого диска может быть выведено предупреждающее сообщение о том, что геометрия этого диска (т. е. число цилиндров, головок и секторов) является неверной, и поэтому процедура установки выбирает более правдоподобную геометрию. При наличии подобного сообщения просто нажмите кнопку **OK**, потому что, как правило, все бывает в порядке.

Если до установки FreeBSD весь жесткий диск занимала ОС Windows, то установите синюю полоску на строку с пометкой «NTFS/HPFS/QNX» и нажмите клавишу D для удаления этого раздела.

Если вы создавали дополнительный (расширенный) раздел для Windows7, то тогда будет и строка с пометкой «extended DOS». Этот раздел также нужно удалить, используя клавишу D.

В результате должна остаться одна строка, помеченная как «unused».

ПРИМЕЧАНИЕ. Для изменения единиц измерения дискового пространства можно воспользоваться клавишей Z.

Теперь нажмите клавишу C – создать раздел. Вам будет предложено указать размер раздела. По умолчанию предлагается все оставшееся дисковое пространство, объем которого выражен числом секторов размером по 512 байтов.

Если вы планируете установить ОС Windows, то укажите размер раздела, который будет занимать ОС FreeBSD.

Учтите, что для нормальной работы с ОС FreeBSD требуется не менее 15 гигабайтов памяти на жестком диске. Конечно, лучше зарезервировать 40–50 Гб, поскольку в состав дистрибутива FreeBSD входит графическая среда KDE, которая занимает довольно много места. Создавать раздел для Windows не нужно, поскольку он будет создан при установке Windows.

Вы можете указать размер раздела в килобайтах, мегабайтах, гигабайтах. Но обязательно нужно указывать единицы измерения, например, 1500M или 2G. Затем вам будет предложено указать код типа раздела. По умолчанию этот код для FreeBSD равен 165, изменять его не нужно – просто нажмите клавишу Enter.

Теперь нажмите клавишу S, чтобы сделать новый раздел активным. Это нужно для продолжения установки. На строке появится символ «**A**».

Если вы что-то сделали не так, как хотели бы, то сможете отменить все изменения, просто нажав клавишу U. Все клавиши и производимые ими действия описаны в нижней части экрана.

Клавиша D служит для удаления раздела. Так что вы можете, в принципе, удалить любой раздел, даже не являющийся разделом FreeBSD (например, раздел Windows с маркировкой «fat» или «NTFS/HPFS/QNX»).

Теперь нажмите клавишу Q для выхода. Вам будет предложено выбрать вид начального загрузчика операционных систем на вашем компьютере. Вы увидите меню из трех строк: **Standard, BootMgr** и **None**. При выборе первой строки будет установлен начальный загрузчик, который при загрузке компьютера выводит список операционных систем, установленных на компьютере. Для выбора требуемой системы из этого списка служат клавиши F1, F2 и т. д. Если в вашем компьютере два и более жестких диска, то этот загрузчик позволит также выбирать требуемый диск. Поэтому мы рекомендуем вам выбрать вторую строкую.

Теперь выйдите в меню, из которого вы выбирали пункт **Partition.**

9. Следующий пункт меню – **Label**. Поместите синюю полосу (курсор) на строку в верхней части экрана, описывающую вновь созданный раздел. Теперь вам нужно создать файловые системы внутри раздела. Начните с создания так называемого раздела подкачки (**swap**). Назначение этого раздела такое же, как и назначение файла подкачки в ОС Windows – он необходим для реализации механизма виртуальной памяти. Нажмите клавишу **C**, затем удалите предложенное число в появившемся окне диалога и введите число, равное двукратному объему оперативной памяти, имеющейся в компьютере. Не забудьте добавить букву M для указания единиц измерения. Например, если ваш компьютер имеет оперативную память объемом в 2Гб, тогда введите – 4096M.

В следующем окне диалога выберите строку **swap**. Раздел подкачки будет создан.

Теперь создайте четыре файловые системы.

ПРИМЕЧАНИЕ. Мы не будем обосновывать выбор того или иного объема файловых систем, чтобы не удлинять описание, скажем только, что предлагаемые величины – не догма, они являются средними или наиболее подходящими для начинающего специалиста по ОС FreeBSD.

Первой будет корневая файловая система. Снова нажмите клавишу C, затем удалите предложенное число и введите размер корневой файловой системы – 500M. Размер корневой файловой системы в версии FreeBSD 8.2 должен быть не менее 180 Мб. Мы предлагаем вам выделить для нее 500 Мб, чтобы у вас был запас свободного пространства. Он может пригодиться потому, что домашний каталог пользователя **root** будет создан в корневой файловой системе. Чтобы перенести его в файловую систему **/home**, требуются некоторые умения, которых у начинающего пользователя ОС FreeBSD может и не быть.

ПРИМЕЧАНИЕ. Если вы укажете меньший объем (например, забыв ввести букву M после числа), то будет выведено сообщение об ошибке что рекомендуемый размер не менее 180 Мб. В этом случае нужно просто переместить синюю полоску курсора на строку с описанием только что созданной корневой файловой системы и удалить ее, нажав клавишу D. Затем необходимо возвратить синюю полоску курсора на строку с именем раздела FreeBSD в верхней части экрана и повторить процедуру создания этой файловой системы.

Теперь выберите **fs** в предложенном меню, состоящем из двух строк, а после этого в ответ на приглашение **«Please specify a mount point...»** укажите так называемую точку монтирования, введя просто один символ **«/»** (без кавычек).

Создайте еще три файловые системы, указывая для них размеры и точки монтирования (будем исходить из размера раздела в 50 Гб.):

– файловую систему для каталогов пользователей: размер – не менее 5 Гб, точка монтирования – **/home** (обратите внимание на наличие символа «/»);

– файловую систему для программ и исходных текстов ядра операционной системы: размер – не менее 30 Гб, точка монтирования – **/usr** (обратите внимание на имя этой файловой системы – не **user**, а **usr**);

– файловую систему для временных файлов, которые формирует сама операционная система: для нее можно выделить оставшееся место в системе, но не менее 1Гб, точка монтирования – **/var**.

Кстати, создавая последнюю файловую систему, вы можете не вводить ее размер, а просто согласиться с размером, предложенным по умолчанию, нажав клавишу Enter. В этом случае все оставшееся место в разделе FreeBSD будет отведено под файловую систему **/var**.

ПРИМЕЧАНИЕ. Приводя примерные размеры файловых систем, мы исходим из того, что размер раздела для FreeBSD составляет 50 Гб. Если вы отвели на жестком диске для FreeBSD больше места, то увеличьте размеры файловых систем **/home** и **/usr**, а размеры корневой файловой системы и системы **/var** можете не увеличивать. Например, при величине раздела FreeBSD, равной 100 Гб, размеры файловых систем **/home** и **/usr** могут составлять 20 Гб и 70 Гб соответственно (не забывайте о том, что еще требуется значительное место для раздела подкачки **swap**). Конечно, для того, чтобы ваша система FreeBSD «чувствовала» себя комфортно, не пожалейте для нее хотя бы 50 Гб дискового пространства. Тогда вы сможете установить графическую среду KDE, включающую в себя много полезных программ и утилит.

В этой утилите также работает режим отмены всех действий (клавиша U) и режим удаления (клавиша D).

После завершения всех операций нажмите клавишу Q для выхода.

10. Пункт **Distributions**. Здесь вам необходимо выбрать модули операционной системы, которые вы хотите установить. Вы можете выбрать одну из стандартных конфигураций, а можете перейти к самой нижней строке **Custom**, что позволит вам осуществить более детальный выбор модулей. Конечно, если размер вашего жесткого диска позволяет, вы можете просто выбрать строчку **All** и установить все системные программы и все исходные тексты системных программ и ядра системы. Если же места маловато, то можно не устанавливать **ports**, **local**, **games**, а в разделе исходных текстов **src** следует выбрать только строку **sys** – исходные тексты ядра операционной системы. Что касается коллекции программ **ports**, то ее можно установить уже после установки операционной системы, если будет необходимость, при наличии достаточного объема свободного места на жестком диске (для нее требуется около 440 Мб). Однако мы рекомендова-

ли бы вам сразу установить все предлагаемые модули и исходные тексты операционной системы. При величине файловой системы **/usr**, равной хотя бы 15 Гб, места для этого будет достаточно.

11. Пункт **Media**. Выберите тип носителя, с которого производится установка, т. е. **CD/DVD** (при установке с CD-ROM).

12. Пункт **Commit**. Завершение подготовки к непосредственному копированию файлов. Вам будет задан вопрос: уверены ли вы? Вы отвечайте утвердительно.

ПРИМЕЧАНИЕ. На этом этапе еще можно, как говорится, дать задний ход и отказаться от установки ОС FreeBSD. При этом все изменения разделов жесткого диска, сделанные вами, будут отменены.

Процесс копирования файлов может занять от 5 минут до 1 часа в зависимости от набора выбранных модулей и мощности вашего компьютера.

13. После завершения копирования файлов вам будет предложено выполнить последние настройки конфигурации. Рекомендуется выбрать **Root password** для задания пароля суперпользователя **root**, а также **Time Zone** для выбора часового пояса. При вводе пароля суперпользователя учтите, что пароль нужно ввести дважды, при этом он не отображается знаками * или какими-либо другими знаками. При выборе часового пояса вам будет задан вопрос, смысл которого состоит в выяснении того, по какому времени «живет» ваш компьютер. Поскольку в России маловероятно использование времени по Гринвичу, то в качестве ответа на этот вопрос выберите **No**. Затем в предлагаемых вам списках континентов, стран и часовых поясов отыщите Россию в Европе и не забудьте о разнице во времени между Красноярском и Москвой в 4 часа (предполагаемые читатели этого руководства проживают в Красноярске).

14. После этого можете выходить из процедуры установки операционной системы, вынимать все диски из дисководов и перезагружать компьютер. Если после перезагрузки вы увидели на экране стартовое меню, состоящее всего из одной строки «F1 – FreeBSD», то это означает, что ОС FreeBSD успешно установлена. Таким образом, самый первый (и самый важный) шаг в освоении новой для вас операционной системы вы сделали.

2. Первое знакомство с операционной системой FreeBSD. Выполнение основных настроек операционной системы после ее установки

Поскольку предполагается, что вы должны суметь самостоятельно выполнить необходимые настройки операционной системы, то сначала вам нужно приобрести минимальные необходимые для этого знания и умения, после чего уже можно будет приступать к выполнению основных настроек системы.

В этой главе вы познакомитесь с ОС FreeBSD на уровне пользователя:

– научитесь входить в систему и выходить из нее;

– изучите основные команды для работы с файлами и каталогами;

– научитесь устанавливать программы в среде этой операционной системы.

Также в этой главе вы узнаете, как выполнить основные настройки ОС FreeBSD. В результате их проведения вы сможете создать для себя удобную рабочую среду.

2.1. Вход в систему и выход из нее

Сразу после загрузки операционной системы FreeBSD вы увидите приглашение

```
login:
```

Введите имя root в ответ на это приглашение и нажмите клавишу Enter. Затем вам будет предложено ввести пароль

```
Password:
```

Введите тот пароль, который вы выбрали при установке операционной системы, и нажмите клавишу Enter (пароль при вводе не отображается в виде символов *). Если вы не ошиблись при вводе имени пользователя и пароля, то вход в систему должен быть успешным. В начале командной строки находится символ #, который, является так называемым «приглашением» (prompt). Поскольку вы вошли в систему с правами суперпользователя **root**, то символ приглашения именно такой. Вы оказались в каталоге **/root**. Чтобы убедиться в этом, введите команду **pwd**. Эта команда печатает имя текущего каталога:

```
pwd
```

```
/root
```

Теперь выйдите из системы, введя команду:

```
exit
```

Снова войдите в систему, но выйдите из нее другим способом – нажав клавиши Ctrl-D.

Теперь вы умеете входить в систему и выходить из нее. Опять войдите в систему и продолжайте изучение ОС FreeBSD.

Для того чтобы выяснить, какая именно операционная система установлена на вашем компьютере, введите команду:

```
uname -a
```

2.2. Простые команды операционной системы UNIX

Прежде всего, сделаем важное замечание: в операционной системе UNIX, к одному из видов которой относится и система FreeBSD, различаются строчные («маленькие») и прописные (заглавные) буквы.

Изучение команд начнем с команды вывода текущей даты:

```
date
```

Кажется естественным ввести следом команду **time**, но этого пока делать не следует, т. к. ее назначение совсем другое, чем в операционных системах MS-DOS и Windows. В UNIX-подобных операционных системах системное время устанавливается также с помощью команды **date**.

Для просмотра списка всех пользователей, работающих в системе в настоящий момент, введите команду

```
who
```

Для определения имени пользователя, под которым вы зарегистрированы в системе, введите

```
who am i
```

или даже без пробелов между самой командой и ее параметрами:

```
whoami
```

В ОС FreeBSD есть удобные программы-календари. Таких программ две: **cal** и **ncal**. Попробуем выполнить первую:

```
cal
```

Теперь сделайте так (2011 – это год):

```
cal 2011
```

А теперь так (5 – это месяц, а 2011 – это год):

```
cal 5 2011
```

Теперь замените **cal** на **ncal** и повторите команды. Вы увидите, что изменился формат вывода календаря.

2.3. Работа с каталогами и файлами

В ОС UNIX, в отличие от ОС Windows, в качестве разделителя каталогов в путевых именах используется символ «/», а не «\». А символы «**.**» и «**..**» означают текущий и родительский каталоги, так же, как в MS-DOS и Windows.

Наверное, одна из наиболее часто используемых команд – команда для просмотра содержимого текущего каталога:

```
ls
```

Если вы не выполняли перехода в другие каталоги после входа в систему, то вы увидите содержимое каталога **/root.**

Эта команда имеет множество различных параметров. Часто используется параметр **-l** (латинская буква «el»), который заставляет эту команду вывести более подробную информацию о файлах и каталогах, содержащихся в текущем каталоге:

```
ls -l
```

Можно указать в команде **ls** имя каталога:

```
ls -l имя_каталога
```

Теперь создайте свой каталог с каким-либо осмысленным именем. При этом используйте латинские буквы (на данном этапе в вашей системе еще не проведена руссификация и вы не сможете вводить буквы русского алфавита). Команда такая:

```
mkdir имя_вашего_каталога
```

Для перехода в каталог служит команда **cd**. Перейдите в только что созданный вами каталог:

```
cd имя_вашего_каталога
```

Теперь создадим какой-нибудь файл без использования текстового редактора. Имена файлов не ограничиваются 8 символами, как в системе MS-DOS. В имени файла может быть более одного символа «.». В имени файла можно использовать те же символы, что и в системе MS-DOS.

Способ 1. Вывод какого-либо текста в несуществующий файл:

```
echo "Какой-то ваш текст" > имя_нового_файла
```

Символ > служит для переадресации стандартного вывода команды в файл.

Если файл с указанным вами именем уже существует, то он будет перезаписан. Если нужно ДОБАВИТЬ текст в файл, то вместо одного символа > нужно поставить два таких символа:

```
echo "Какой-то ваш текст" >> имя_существующего_файла
```

Способ 2. Использование переадресации таким вот образом:

```
> имя_нового_файла
```

Но в этом случае файл будет пустым (длина его – 0 байтов). Иногда требуется создать именно пустой файл, например, для записи в него каких-либо диагностических или системных сообщений, выдаваемых программой-сервером.

Способ 3. Копирование существующего файла командой **cp**:

```
cp имя_существующего_файла имя_нового_файла
```

Теперь переименуйте один из созданных вами файлов:

```
mv старое_имя новое_имя
```

Создайте еще один каталог командой **mkdir** и переместите один из ваших файлов в этот каталог (для того, чтобы лучше ориентироваться в иерархии каталогов, используйте команду **pwd**):

```
mv имя_файла имя_каталога
```

Теперь удалите один из файлов:

```
rm имя_удаляемого_файла
```

Ни в коем случае не удаляйте из вашего домашнего каталога **/root** те файлы, имена которых начинаются с символа точки «.» – это конфигурационные файлы.

Удалите из созданного вами «тренировочного» каталога все файлы, а затем удалите и сам каталог (он теперь пустой):

```
rmdir имя_удаляемого_каталога
```

2.4. Конвейер команд

Так называемый **конвейер команд** формируется для того, чтобы результат работы одной команды передать следующей команде. Для этого используется символ «|» (чтобы его было легче найти на клавиатуре, напомним, что он совмещен с символом «\»). В конвейере может быть более двух команд. Например, для подсчета числа строк, слов и символов в тексте можно использовать такую конструкцию:

```
cat имя_вашего_файла | wc
```

В этом конвейере команда **cat** выводит содержимое файла на стандартный вывод, а команда **wc** подсчитывает строки, слова и символы.

Конечно, в данном случае можно было бы применить и переадресацию ввода таким способом:

```
wc < имя_вашего_файла
```

Еще одно традиционное применение конвейера – для просмотра результатов работы какой-либо программы, если они не умещаются на одном экране, например, при просмотре длинного каталога:

```
ls -l | more
```

Команда **more** позволяет просмотреть выводимую информацию поэкранно.

ПРИМЕЧАНИЕ. Частое использование конвейеров команд является важной особенностью ОС UNIX.

2.5. Вызов справки по командам UNIX

В составе операционной системы FreeBSD присутствует большое количество справочной информации. Она представлена в различных форматах, в частности, в формате html. Но для предоставления оперативной информации по всем командам операционной системы, ее конфигурационным файлам, функциям языка программирования С служит специальная система электронных руководств **manual pages**, которые называют сокра-

щенно **man pages**. Эти электронные руководства имеют специальный внутренний формат и вызываются по команде **man.** Для получения справки, например, по команде **ls** введите:

```
man ls
```

Для просмотра справки используйте клавиши управления курсором, клавиши PageDown и PageUp, а для выхода – клавишу Q. При просмотре руководства **man** можно вызвать подсказку по работе с руководством, нажав клавишу H. Можно получить помощь по самой справочной системе manual pages, введя

```
man man
```

Иногда бывает необходимо выяснить, на каких man-страницах содержится та или иная информация. Сделать это можно с помощью команды **apropos**. Например, для выявления всех страниц, на которых упоминается конфигурационный файл **profile**, введите команду

```
apropos profile
```

2.6. Понятие виртуального терминала

Нажмите клавиши Alt-F2 (или Ctrl-Alt-F2) – вы видите новый экран для регистрации в системе. Обратите внимание на верхнюю строку. В ней есть пометка (ttyv1). Нажмите клавиши Alt-F3 (или Ctrl-Alt-F3) – вы видите еще один новый экран для регистрации в системе. Обратите внимание на верхнюю строку. В ней есть пометка (ttyv2). Можно продолжать так до Alt-F8 (после установки операционной системы именно таковы начальные настройки этой функции). Все эти экраны называются **виртуальными терминалами**. Заметьте, что счет терминалов начинается с нуля. Вернитесь на терминал 1 (клавиши Alt-F2). Теперь войдите в систему на новом терминале так же, как вы это уже делали раньше. Дальнейшие инструкции вы можете выполнять на любом из терминалов.

2.7. Установка самых необходимых программ

Конечно, можно долго спорить о том, какие программы являются самыми необходимыми или самыми удобными, или самыми универсальными. Однако нашей целью является дать вам, уважаемый читатель, основные умения и навыки работы с ОС FreeBSD. Но до тех пор, пока у вас этих знаний и умений нет, мы возьмем на себя смелость принимать решения за вас (т. е. утверждать, *что* является необходимым, важным, неот-

ложным и т. д.). Набрав некоторый опыт, вы сможете сформировать свои собственные мнения, вкусы и привычки, возможно, отличающиеся от наших.

Установка программ в операционных системах, относящихся к классу UNIX-подобных систем, может производиться двумя способами. Первый способ хорошо известен пользователям операционной системы Windows и заключается в установке заранее скомпилированной программы, которая поставляется в виде исполняемого файла (или целого комплекта исполняемых файлов и файлов других типов). Такой способ реализован в ОС FreeBSD в форме так называемых **пакетов (packages)**.

Второй способ заключается в установке программ непосредственно из исходных текстов. Компиляция этих текстов производится на компьютере пользователя.

2.7.1. Установка файлового менеджера Demos Commander

Выполним установку файлового менеджера **Demos Commander (deco)** из исходных текстов программ. Это довольно простой файловый менеджер, но он подходит для большинства видов работы с файлами в системе FreeBSD. Исходные тексты этого программного продукта можно получить в сети Интернет по адресу http://sourceforge.net/projects/deco/.

Теперь необходимо решить следующую задачу (конечно, это элементарный вопрос для того, кто уже имеет некоторый опыт работы в системе UNIX, но для вас пока что это действительно может являться задачей): каким-то образом перенести исходные тексты на ваш компьютер. Покажем два способа из целого ряда возможных. Первый способ такой. Записать архивный файл с исходными текстами на диск CD/DVD. Затем сделать этот диск доступным в системе FreeBSD. Для реализации этого необходимо выполнить операцию **монтирования** файловой системы диска CD в файловую систему FreeBSD. Данная операция выполняется командой

```
mount /cdrom
```

Теперь вам необходимо скопировать архивный файл с исходными текстами программы **Demos Commander** в каталог **/root** на жестком диске. Для этого с помощью команды **cd** перейдите в каталог **/cdrom**

```
cd /cdrom
```

В этом каталоге и находится теперь корневой каталог DVD диска. Имя архивного файла, скорее всего, **deco39.tgz** или **deco39.tar.gz** (в качестве имени архива в примерах команд мы будем использовать **deco39.tgz**). С помощью команд **cd** и **ls** найдите архивный файл на DVD диске и скопируйте его в каталог **/root** командой

```
cp deco39.tgz /root
```

Для упрощения вашей задачи на данном этапе освоения системы FreeBSD постарайтесь сделать так, чтобы на DVD диске не было русскоязычных имен подкаталогов на всем пути от корневого каталога до архивного файла с исходными текстами **deco**. В последующих главах нашего учебного пособия мы расскажем, как сделать так, чтобы русскоязычные имена каталогов и файлов не создавали проблем при монтировании файловых систем внешних носителей информации в системе FreeBSD.

После копирования файла можно удалить CD диск из дисковода. Но если вы попытаетесь это сделать, не выполнив предварительно операцию размонтирования файловой системы CD диска, дисковод диск вам не «отдаст». Поэтому сначала выйдите из каталога **/cdrom** (иначе операция размонтирования не получится, поскольку устройство используется):

```
cd /root
```

А теперь выполните операцию размонтирования (обратите внимание на правильное написание команды: **umount**, а не unmount):

```
umount /cdrom
```

Второй способ переноса архивного файла на ваш компьютер отличается от первого лишь типом носителя информации – речь идет об устройстве флэш-памяти. Вставьте это устройство в USB-разъем. Сразу после вставки устройства на экран будут выведены несколько строк текста повышенной яркости – операционная система опознает устройство и выводит информацию о его типе, емкости и т. д. Чтобы освободить командную строку от этих сообщений, нажмите один раз клавишу Enter. Теперь смонтируйте устройство флэш-памяти командой

```
mount_msdosfs /dev/da0s1 /mnt
```

Тем же способом, который описан для случая с CD диском, отыщите архивный файл **deco39.tgz** на устройстве флэш-памяти, содержимое которого теперь доступно через каталог **/mnt**. Скопируйте архив в каталог **/root**, а затем выйдите из каталога **/mnt** и размонтируйте устройство:

```
cd /root
umount /mnt
```

Итак, в каталоге **/root** у вас находится архивный файл с исходными текстами **deco39.tgz**. Его необходимо разархивировать с помощью архиватора **tar** (в последующих главах будет более подробно рассказано об этой программе). Введите команду:

```
tar xzvf deco39.tar.gz
```

В результате ее работы будет создан подкаталог **deco39**, в котором разместятся файлы исходных текстов. Перейдите в этот подкаталог и выполните ряд команд (обратите внимание на символы «./» в одной из команд):

```
cd deco39
./configure
make
make check
make install
```

Если на экран не было выведено сообщений об ошибках, то программа **deco** успешно установлена. Чтобы запустить ее, выйдите из системы с помощью команды **exit** или с помощью комбинации клавишей Ctrl-D, а затем войдите снова и введите команду

```
deco
```

Поскольку этот файловый менеджер похож на другие программы данного типа, мы не будем подробно описывать его использование. Сделаем только ряд замечаний.

1. Когда вы выполняете какую-нибудь команду из командной строки, предоставляемой **deco**, то после завершения работы этой команды панели файлового менеджера на экране не отображаются, давая место той информации, которую выводит введенная вами команда. Для их отображения нужно просто нажать клавишу Enter. Запускать файловый менеджер повторно, опять вводя команду **deco**, не нужно.

2. Для того, чтобы отредактировать команду, введенную в командной строке, необходимо иметь возможность перемещать курсор вдоль командной строки. Эта возможность появляется после нажатия клавишей Ctrl-P, при этом перемещение курсора по панелям **deco** становится невозможным. Повторное нажатие этой же комбинации клавишей отключает функцию перемещение курсора вдоль командной строки.

3. Для выхода из встроенного редактора (вызывается клавишей F4) и утилиты для просмотра файлов (клавиша F3) можно пользоваться не только клавишей F10, но и клавишей Esc. Но при нажатии клавиши Esc срабатывание происходит с задержкой около одной секунды. Чтобы этой задержки не было, нужно нажимать клавишу Esc дважды.

4. Чтобы русский текст отображался корректно при просмотре текстовых файлов с помощью утилиты, вызываемой по клавише F3, нажмите эту клавишу еще раз.

Добавим напоследок, что символ * слева от имени файла означает, что данный файл является исполняемым.

2.7.2. Установка текстового редактора joe

Выполним установку текстового редактора **joe** также из исходных текстов. Это многооконный редактор, имеющий богатые возможности для редактирования исходных текстов программ на различных языках программирования. Исходные тексты этого программного продукта можно получить в сети Интернет по адресу http://sourceforge.net/projects/joe-editor.

ПРИМЕЧАНИЕ. Вы можете выполнять все команды, приведенные ниже, в среде файлового менеджера **Demos Commander**. При этом можно пользоваться его способностью сохранять историю введенных команд. Эти команды можно просматривать с помощью комбинаций клавишей Ctrl-E (просмотр назад) и Ctrl-X (просмотр вперед). Выбрав нужную команду, ее можно при необходимости отредактировать (не забывайте, что для перемещения курсора по командной строке нужно сначала нажать клавиши Ctrl-P), а затем выполнить, нажав клавишу Enter.

Как и при установке файлового менеджера **deco**, смонтируйте носитель (CD диск или устройство флэш-памяти), на котором находится архивный файл **joe-3.5.tar.gz** (или **joe-3.5.tgz**). Затем с помощью команд **cd** и **ls** отыщите архивный файл и скопируйте этот архив в каталог **/root**. Затем выйдите из каталога **/cdrom** (или **/mnt**) и размонтируйте устройство:

```
cp joe-3.7.tar.gz /root
cd /root
umount /cdrom
```

Теперь разархивируйте архив с исходными текстами редактора **joe**:

```
tar xzvf joe-3.7.tar.gz
```

В результате будет создан подкаталог **joe-3.7**, в котором разместятся файлы исходных текстов. Перейдите в этот подкаталог и выполните ряд команд (обратите внимание на символы «./» в одной из команд):

```
cd joe-3.7
./configure
make
make install
```

Если на экран не было выведено сообщений об ошибках, то программа **joe** успешно установлена. Чтобы запустить ее, введите команду

```
joe
```

Если редактор запустится, то введите в его окне какой-нибудь текст (пока что это можно сделать только с использованием латинского алфавита). Попробуйте выполнить операцию удаления введенных символов с помощью клавишей Backspace и Delete. Как вы можете убедиться, клавиша

Delete работает так же, как и клавиша Backspace. Конечно, это не очень удобно, но вскоре мы покажем, как можно добиться от клавиши Delete традиционного функционирования.

Данный редактор имеет следующую особенность: многих из его функций активизируются при наборе комбинации из *трех* клавишей. Например, для получения подсказки необходимо использовать комбинацию Ctrl-K-H. Комбинации из трех клавишей нужно набирать таким образом: нажав клавишу Control и удерживая ее, нажать *поочередно* две символьные клавиши, в данном случае это клавиши K и H. Чтобы убрать с экрана текст подсказки, наберите комбинацию Ctrl-K-H еще раз.

Для сохранения введенного текста в файле служит комбинация клавишей Ctrl-K-D, для выхода из редактора (точнее, для закрытия файла) с сохранением внесенных изменений текста – Ctrl-K-X, а для выхода без сохранения изменений – Ctrl-C. Чтобы выделить блок текста, нужно установить курсор в начало нужного фрагмента и нажать клавиши Ctrl-K-B, затем перевести курсор в конец фрагмента текста и нажать клавиши Ctrl-K-K. Для копирования выделенного блока установите курсор в нужное место и нажмите клавиши Ctrl-K-C, а для перемещения блока – Ctrl-K-M. Мы рекомендуем изучить основные приемы работы с редактором с помощью его подсказки. В тексте этой подсказки в качестве обозначения клавиши Control используется символ ^. Рекомендуем также воспользоваться и электронным руководством, вызвав его с помощью команды

```
man joe
```

Теперь сделаем ряд настроек в конфигурационном файле редактора. Этот файл находится в каталоге **/usr/local/etc/joe** и называется **joerc**.

Прежде чем вносить изменения в файл **joerc**, сделайте его копию (в имя копии файла можно для наглядности добавить расширение orig, т. е. original):

```
cd /usr/local/etc/joe
cp joerc joerc.orig
```

Для внесения изменений в конфигурационный файл **joerc** вы можете воспользоваться либо встроенным редактором файлового менеджера **deco**, либо самим редактором **joe**, используя который, не забывайте о том, что клавиша Delete работает так же, как и клавиша Backspace.

Обратите внимание на то, что файл **joerc** содержит не только конфигурационные параметры, но и инструкции по их использованию.

Внесем следующие изменения в конфигурацию редактора.

1. Найдите строку (строка номер 60), в начале которой располагается параметр **-asis**. Он отвечает за правильное отображение букв русского алфавита (вопросы русификации операционной системы будут рассмотрены

немного позднее). Обратите внимание, то перед этим параметром в начале строки стоит один пробел. Нужно удалить этот пробел, тем самым параметр будет активизирован.

2. Поскольку использовать комбинации из трех клавишей не очень удобно, назначим для нескольких часто используемых операций функциональные клавиши:

– клавишу F1 для вызова подсказки (помощи). Для этого выполните следующие действия: перейдите в файле **joerc** в район строки номер 707; в этом месте находятся три строки, начинающиеся с ключевого слова help, которое является кодовым наименованием операции вызова экранной подсказки. Добавьте после этих строк еще одну строку с ключевым словом help, а в качестве кода клавишей введите «.k1» (кавычки вводить не нужно). Обратите внимание, что ключевое слово отделяется от кода клавишей двумя символами табуляции (табуляция используется и во всех остальных случаях, которые мы сейчас опишем);

– клавишу F5 для перехода к началу файла. Выполните аналогичные действия: перейдите в файле **joerc** в район строки номер 945, в этом месте находятся три строки, начинающиеся с ключевого слова bof, которое является кодовым наименованием операции перехода к началу файла. Добавьте после этих строк еще одну строку с ключевым словом bof, а в качестве кода клавишей введите «.k5» (кавычки вводить не нужно);

– клавишу F6 для перехода к концу файла. Поскольку схема действий вам уже ясна, будем давать лишь краткие указания: строка номер 972, ключевое слово eof (операция перехода к началу файла). Добавьте после последней из строк с этим ключевым словом еще одну строку с ключевым словом eof, а в качестве кода клавишей введите «.k6» (без кавычек);

– клавишу F7 для поиска текстовой строки в файле. В районе строки номер 982 есть строки с ключевым словом ffirst. Добавьте еще одну строку, а в качестве кода клавишей введите «.k7»;

– клавишу F2 для сохранения файла. В районе строки номер 1034 есть строки с ключевым словом save. Добавьте еще одну строку, а в качестве кода клавишей введите «.k2».

3. Исправьте «поведение» клавиши Delete по уже знакомой вам схеме. В районе строки номер 959 есть строки с ключевым словом delch. Добавьте еще одну строку, а в качестве кода клавишей введите «^?» (без кавычек).

ПРИМЕЧАНИЕ. Номера строк в файле **joerc** указаны приблизительно, т. к. они могут изменяться после вставки новых строк в процессе проведения настроек.

Мы сделали лишь самые необходимые настройки. Скажем еще несколько слов о работе с несколькими файлами одновременно. Чтобы открыть в редакторе несколько файлов, передайте их имена в командной строке:

```
joe file1 file2 file3
```

Для перехода между различными окнами редактора, содержащими открытые файлы, служат клавиши Ctrl-K-N (переход к следующему окну) и Ctrl-K-P (переход к предыдущему окну).

Открыть файл можно и непосредственно из редактора с помощью клавишей Ctrl-K-E. Получив приглашение для ввода имени файла, можно имя не вводить, а вместо этого нажатием клавиши Tab вывести на экран список файлов текущего каталога и выбрать файл из списка. Можно также перемещаться по структуре каталогов в поисках нужного файла.

Иногда встречается такая ошибка: невозможно сохранить файл. Это может объясняться тем, что начинающие пользователи операционной системы UNIX (в частности, FreeBSD), перемещаясь по каталогам системы, запускают редактор, находясь не в своем домашнем каталоге, а в том каталоге, права на запись файлов в который этот пользователь не имеет. Если введен уже значительный объем текста, потерять который нежелательно, то при сохранении файла, когда будет предложено ввести его имя, следует ввести не только имя, но и полный путь к нему, причем, этот путь должен вести в ваш домашний каталог. Например: **/home/stud/my_file.txt.**

2.7.3. Установка браузера lynx

Установку браузера **lynx** выполним путем компиляции исходных текстов программ.

Исходные тексты этого программного продукта можно получить в сети Интернет по адресу http://lynx.isc.org/lynx2.8.7/index.html.

Как и при установке файлового менеджера **deco**, смонтируйте носитель (CD диск или устройство флэш-памяти), на котором находится архивный файл lynx 2.8.7.tar.gz или lynx 2.8.7.tar.bz2. Затем с помощью команд **cd** и **ls** отыщите архивный файл и скопируйте этот архив в каталог **/root.** Затем выйдите из каталога **/cdrom** (или **/mnt**) и размонтируйте устройство:

```
cp lynx 2.8.7.tar.gz /root
cd /root
umount /cdrom
```

Теперь разархивируйте архив с исходными текстами браузера **lynx**:

```
tar xzvf lynx 2.8.7.tar.gz
```

В результате будет создан подкаталог lynx 2-8-7, в котором разместятся файлы исходных текстов. Перейдите в этот подкаталог и выполните ряд команд (обратите внимание на символы «./» в одной из команд):

```
cd lynx 2-8-7
./configure
make
```

```
make install
make install-help
make install-doc
```

Если на экран не было выведено сообщений об ошибках, то программа **lynx** успешно установлена.

Проверьте работоспособность установленного браузера. Для этого можно воспользоваться системной документацией, представленной в формате html. Введите в командной строке:

```
lynx /usr/share/doc/handbook/book.html
```

В нижней строке экрана приведена краткая подсказка. Полезной функцией является функция поиска в html-документе. Она активизируется с помощью клавиши «/». Также может потребоваться вызвать модуль настройки параметров. За эту функцию отвечает клавиша O (options). Для просмотра русскоязычных документов может потребоваться явно указать браузеру языковые параметры:

– **Display character set** (нажатием клавиши Enter откройте список и выберите из него значение **Cyrillic (KOI8-R)**);

– **Assumed document character set** (выберите из списка значение **koi8-r**).

ПРИМЕЧАНИЕ. Значения в этих списках расположены не по алфавиту.

Просмотр страниц выполняется с помощью клавишей PageDown и PageUp, а также клавиши «пробел». Для выбора нужной ссылки используйте клавиши «Курсор вверх/вниз», а для перехода по этой ссылке нажимайте клавишу Enter. Возвратиться к предыдущей странице можно с использованием клавиши «Курсор влево». Для завершения работы с браузером служит клавиша Q.

2.8. Локализация системы

Локализация системы дает возможность работать не только с англоязычными текстами, но и с текстами на других языках. Естественно, нас интересует возможность использования русского языка, т. е. «русификация» системы. Подробное описание всех вопросов локализации можно найти в системной документации **/usr/share/doc/handbook/book.html**. Эта же документация представлена и на русском языке:

/usr/share/doc/ru_RU.KOI8-R/books/handbook/book.html.

Покажем, как провести процедуру локализации для русского языка.

ПРИМЕЧАНИЕ. Для проведения «русификации» в операционной системе UNIX (в т. ч. FreeBSD), как правило, используется кодировка символов koi8-r (а в системе Windows – cp1251).

1. Необходимо внести изменения в файлы **/etc/profile**, **/etc/csh.login**, **/etc/rc.conf**, **/etc/ttys**. Рекомендуем вам сначала сделать их резервные копии:

```
cd /etc
cp profile profile.orig
cp csh.login csh.login.orig
cp rc.conf rc.conf.orig
cp ttys ttys.orig
```

2. Добавьте в файл **/etc/profile** следующие строки:

```
LANG=ru_RU.KOI8-R; export LANG
MM_CHARSET=KOI8-R; export MM_CHARSET
```

3. Добавьте в файл **/etc/csh.login** следующие строки:

```
setenv LANG ru_RU.KOI8-R
setenv MM_CHARSET KOI8-R
```

4. Добавьте в файл **/etc/rc.conf** следующие строки:

```
keyrate=”fast”
keymap=”ru.koi8-r”
moused_enable=”YES”
mousechar_start=3

font8x16=”koi8-r-8x16”
font8x14=”koi8-r-8x14”
font8x8=”koi8-r-8x8”
```

Параметр **keyrate** устанавливает короткую паузу между нажатием символьной клавиши и началом повторений вводимого символа, а также высокую скорость повторений символа. Два параметра управляют работой «мыши». Причем, второй из них необходим для того, чтобы при перемещении указателя «мыши» по экрану не искажались символы на экране. Вы можете закомментировать этот параметр (поставив символ # в начале строки) и перезагрузив компьютер, посмотреть на эффект, производимый при перемещении указателя «мыши» по русскому тексту на экране.

Последние строки указывают системе, какими шрифтами она должна выводить символы на экран.

5. В файле **/etc/ttys** измените тип терминала cons25 на cons25r в восьми строках, задающих параметры работы для псевдотерминалов ttyv*.

6. Перезагрузите компьютер и попробуйте ввести русские буквы в командной строке до запуска файлового менеджера **deco** и после его запуска. Переключение русских и латинских символов выполняется клавишей CapsLock.

Если русские буквы ввести невозможно, значит, вы ошиблись при корректировании указанных файлов. Возможная ошибка – наличие пробелов между именем параметра, знаком равенства (присваивания) и его значением. Пробелы недопустимы!

Вы можете с помощью «мыши» выделять текст в текстовом редакторе или командной строке и вставлять его в нужное место экрана с помощью клавишей Shift-Insert.

2.9. Создание учетных записей пользователей

В ОС UNIX есть самый главный пользователь, который имеет максимальные полномочия на выполнение всех операций в системе. Его имя – **root**. До сих пор вы работали в системе, регистрируясь под именем именно этого пользователя. Следует учитывать, что работать с правами суперпользователя (или «под root'ом», как говорят UNIX-гуру) нужно очень аккуратно, т. к. последствия ошибки могут быть очень серьезными. Поэтому не удаляйте те файлы, назначение которых вам не известно. Однако, с другой стороны, не стоит и чрезмерно опасаться возможного риска, иначе вы ничему не научитесь. Мы рекомендовали бы вам поступать таким образом: прежде чем вносить изменения в какой-либо системный файл, сначала сделайте его копию, добавив к имени файла-копии суффикс, например, «.orig». А в самих системных файлах можно делать пометки для памяти, указывая дату внесения изменения. Как правило, в системных файлах строки-комментарии начинаются с символа #. Приведем в качестве небольшого примера фрагмент файла **rc.conf**:

```
keymap="ru.koi8-r"
moused_enable="YES"
mousechar_start=3
```

От советов и предостережений перейдем непосредственно к делу. Для создания учетной записи пользователя (учетные записи – **user accounts** – в русском переводе называют еще бюджетами пользователей, или просто «пользователями») служит программа **adduser**. Она написана на языке shell (речь о нем будет идти немного позднее) и находится в каталоге **/usr/sbin**. Запустите эту программу, набрав на клавиатуре:

```
adduser
```

Далее следуйте указаниям, которые программа выводит на экран. Рассмотрим эти указания в порядке их поступления.

```
Username:
```

Вам предлагается совершить самый ответственный шаг – ввести имя нового пользователя. Оно служит для входа в систему. Можно использовать только латинские буквы нижнего регистра, цифры и символы «-» и «_». В качестве примера введите имя **stud**.

```
Full name []:
```

Предлагается ввести полное имя нового пользователя. Оно служит не для входа в систему, а только для справки (к тому же, его можно изменить потом). Можно использовать НЕ только латинские буквы нижнего регистра, цифры и символы «-» и «_».В качестве примера введите: **Student's account**.

```
Uid (Leave empty for default):
```

Предлагается указать числовой идентификатор пользователя. В файловой системе ОС UNIX с файлами и каталогами связаны не имена пользователей, а именно эти числовые идентификаторы. Но при выводе информации программой, например, **ls**, вместо числовых значений выводятся имена пользователей (для наглядности), которые хранятся в файле паролей. Если у вас нет особых оснований для выбора какого-то конкретного номера, то просто нажмите клавишу Enter.

```
Login group [stud]:
```

Предлагается ввести имя группы, в которую по умолчанию входит данный пользователь, в качестве которого мы выбрали **stud**. При этом имя группы предлагается сделать таким же, как и имя пользователя. Это удобно при работе в многопользовательской среде, когда есть необходимость ограничить доступ других пользователей к домашнему каталогу вашего пользователя.

Если вы работаете на компьютере коллективного пользования, то соглашайтесь на предлагаемый вариант, т. е. нажмите клавишу Enter. Если же это ваш домашний компьютер, то вы можете указать другую группу, а можете также оставить предлагаемую по умолчанию: ничего страшного не случится.

```
Login group is stud. Invite stud into other groups? []:
```

Предлагается включить нового пользователя в другие группы, существующие в системе. Нажмите клавишу Enter, чтобы пока что не включать его в другие группы. Включить пользователя в другие группы вы легко сможете потом.

```
Login class [default]:
```

Вам предлагается указать класс (тип) пользователя. От этого класса зависит объем ресурсов, предоставляемый системой данному пользователю. Если вы просто нажмете клавишу Enter, то ничего страшного не случится: в этом случае будет использоваться тип по умолчанию, который определяется в файле **/etc/login.conf** и имеет имя **default**. Как правило, того набора системных ресурсов, который предоставляется пользователю по умолчанию, вполне достаточно для типичного пользователя ОС UNIX.

```
Shell (sh csh tcsh nologin) [sh]:
```

Вам предлагается указать командный интерпретатор для вашего нового пользователя (о командных интерпретаторах речь пойдет в следующих главах). Нужно выбрать из предложенного списка. Вариант, предлагаемый по умолчанию, помещен в скобки. Если вы не знаете, какой вариант выбрать, то можете просто нажимать клавишу Enter в ответ на все предложения.

```
Home directory [/home/stud]:
```

Укажите имя домашнего каталога для вашего нового пользователя. Предлагается имя, совпадающее с именем пользователя. Как правило, такое имя и оставляют. Так что нажимайте клавишу Enter.

```
Use password-based authentication? [yes]
```

Задается вопрос о том, требуется ли использовать пароль для аутентификации пользователя, т. е. определения его подлинности. Рекомендуем согласиться с предложенным значением, т. е. нажать клавишу Enter.

```
Use an empty password? (yes/no) [no]
```

Использовать «пустой» пароль? Рекомендуем так не делать даже дома, чтобы не сформировалось устойчивое беспечное отношение к проблемам безопасности. Поэтому нажмите клавишу Enter.

```
Use a random password? (yes/no) [no]
```

Использовать пароль, который случайным образом сформирует операционная система? Рекомендуем сейчас выбрать предложенный вариант «no», чтобы ввести пароль самостоятельно. Также нажмите клавишу Enter.

```
Enter password:
```

Введите пароль, например, **student**. При вводе он не отображается даже знаками *. После его ввода появится сообщение

```
Enter password again:
```

Вы должны ввести пароль еще раз. Если совпадения паролей нет, вы получите сообщение

```
Passwords didn't match!
```

А затем вам будет предложено ввести пароль вновь. После того, как вы корректно введете пароль, будет выведен запрос о необходимости блокирования учетной записи сразу после ее создания:

```
Lock out the account after creation [no]:
```

Поскольку блокировать новую учетную запись не нужно, то нажмите клавишу Enter, чтобы выбрать вариант по умолчанию.

Теперь на экран будет выведена вся информация о новой учетной записи, для того чтобы вы могли ее еще раз проверить.

```
Username    : stud
Password    : ******
Full name   : Student's account
Uid         : 1001
Class       :
Groups      : stud
Home        : /home/stud
Shell       : /bin/sh
Locked      : no
OK? (yes/no):
```

В последней строке этого блока информации содержится вопрос к вам. Для подтверждения создания учетной записи введите «yes» и нажмите клавишу Enter. Будет выведено информационное сообщение:

```
adduser: INFO: Successfully added (stud) to the user database
```

Вам будет предложено создать еще одну учетную запись пользователя:

```
Add another user? (yes/no):
```

На этот вопрос вы можете ответить как утвердительно (введя «yes»), так и отрицательно (введя «no»). Ответьте утвердительно и создайте еще одну учетную запись пользователя. Она пригодится потом при изучении операции удаления пользователя из системы.

Теперь вы можете попытаться войти в систему под именем нового пользователя. Для этого перейдите на следующий терминал, нажав клавиши Alt-F2 (или Alt-F3, если второй терминал уже занят) и введите имя пользователя и пароль

```
login: stud
Password:
```

Войдя в систему, вы увидите, что системное приглашение изменилось. Для суперпользователя root в качестве приглашения использовался символ #, а для обычных пользователей используется символ $. Далее вы можете запустить файловый менеджер **deco**. Изменения ожидают вас и здесь – цвет интерфейса изменился с желтого на синий.

2.10. Первая программа на языке C/C++ в операционной системе UNIX

В состав операционной системы FreeBSD входит и компилятор языка C. Мы предлагаем вам написать простейшую программу, в качестве которой во всех учебниках традиционно предлагается программа «Привет, мир».

```
#include <stdio.h>

int main( void )
{
  printf( ”Привет, мир!\n” );
  return 0;
}
```

Введите текст этой программы в текстовом редакторе **joe** и сохраните его в файле с расширением «.c». Для компиляции введите в командной строке:

```
cc имя_вашего_исходного_файла
```

Если вы являетесь поклонником языка программирования C++, то исходный текст будет таким:

```
#include <iostream>
using name space std;
void main( void )
{
  cout <<”Привет, мир!\n”;
  return;
}
```

и команда для компиляции исходного файла будет такой:

```
g++ имя_вашего_исходного_файла
```

Если компилятор выдаст сообщения об ошибках, то откорректируйте исходный текст программы в редакторе **joe**. Если все пройдет успешно, то в текущем каталоге будет создан исполняемый файл с именем **a.out**. Запустите его таким способом:

```
./a.out
```

Символы «./» означают текущий каталог. Здесь необходимо сделать важное замечание, а именно: текущий каталог «.» не является «поисковым» по умолчанию в системе UNIX. Это означает, что когда вы вводите имя какой-нибудь программы, то система UNIX не производит поиск этой программы в текущем каталоге. Это сделано в интересах безопасности. Однако можно изменить настройки системы таким образом, что при поиске исполняемых файлов будет просматриваться и текущий каталог. Для этого нужно в команду **PATH**, которая содержится в файле **.profile**, находящемся в домашнем каталоге пользователя, добавить символ текущего каталога, т. е. точку:

```
PATH=.:/bin:/usr/bin:/usr/local/bin
```

К сожалению, имя **a.out** совершенно неинформативно. Гораздо удобнее, когда имя исполняемого файла является более осмысленным. Конечно, можно исполняемый файл переименовать, но можно изменить команду компиляции с тем, чтобы использовалось не имя по умолчанию **a.out**, а то имя, которое вам нужно:

```
cc -o имя_исполняемого_файла имя_вашего_исходного_файла
```

Однако при запуске вашего исполняемого файла не забывайте про символы «./».

2.11. Перезагрузка и останов операционной системы

Для перезагрузки системы можно использовать два способа. Первый способ заключается в одновременном нажатии комбинации клавишей Ctrl-Alt-Delete. Второй способ, использование команды **reboot.**

Для останова системы также можно использовать комбинацию клавишей Ctrl-Alt-Delete: когда компьютер в процессе перезагрузки начинает аппаратный тест, нужно нажать кнопку выключения электропитания.

Существует и специальная команда **shutdown**. Чтобы немедленно остановить систему, введите

```
shutdown -p now
```

Контрольные вопросы и задания

1. Напишите исходный текст программы, которая вывела бы на экран монитора любой небольшой текст. Откомпилируйте программу и выполните исполняемый файл. Какое имя он имеет по умолчанию? Как задать другое имя исполняемого файла при компиляции? Для чего нужно перед именем исполняемого файла указывать символы «./»?

2. Как получить справку по командам ОС UNIX? Как получить справку по функциям языка С?

3. Что такое виртуальный терминал? Каким образом можно переключаться между виртуальными терминалами?

4. Как используется переадресация ввода/вывода? Как можно дополнить имеющийся файл новыми данными?

5. Что такое конвейер команд? Приведите пример его использования.

6. Какие команды для просмотра календаря вы знаете? В чем различие между ними?

7. Отображается ли пароль в виде символов * при входе в систему?

3. Файловая система операционной системы FreeBSD

Эта глава познакомит вас с файловой системой ОС FreeBSD. Вы сможете:

– изучить способ задания привилегий (прав) доступа к файлам и каталогам;

– познакомиться с таким важным понятием как монтирование файловой системы;

– узнать, что такое бит смены идентификатора пользователя (группы) и для чего он служит;

– познакомиться со ссылками.

3.1. Сведения о правах доступа к файлам в системе UNIX

Для просмотра содержимого каталогов используется команда **ls**. Если указать для нее параметр **-l** (это латинская строчная буква «el»), то можно получить подробный список, например:

```
-rw-r--r--   1 root  wheel     297 26 сен  2000 .login
drwx------   5 root  wheel     512  2 окт 23:07 .netscape
-rw-r--r--   1 root  wheel     380 19 мар  2001 .profile
-rw-------   1 root  wheel      24 22 сен 16:40 .rhosts
-rwxr-xr-x   1 root  wheel      49  8 май 17:12 Apache.start
drwxr-xr-x   3 stud  wheel     512 27 май 23:34 C++
-rw-r--r--   1 stud  wheel      23  9 апр  2001 aaa
drwxr-xr-x   4 stud   stud     512  9 окт 12:10 developments
-rw-r--r--   1 root  wheel       0 11 окт 12:15 ls.txt
drwx------   2 root  wheel     512  7 апр  2001 nsmail
-rw-r--r--   1 root  wheel  228241 11 окт 10:50 perlfunc.man
```

Опишем назначение каждой колонки этого списка. В первой колонке показаны права доступа к файлу или каталогу. Они представляют собой десять символов. Первый из них обозначает тип файла: если этот символ d, то это каталог, если же это символ «-», то это обычный файл. Существуют и другие типы файлов, но пока мы о них говорить не будем. Следующие девять символов нужно рассматривать как три группы по три символа. Предварительно нужно сказать немного о так называемых **группах пользователей**. Как вы знаете, для входа в систему нужно ввести входное имя пользователя и пароль. Администратор операционной системы может объединять пользователей в группы, исходя из конкретных соображений. Иногда группа может состоять только из одного пользователя (точнее, из пользователей, имеющих одинаковое имя), например, **stud**. В нашем случае группа называется так же: **stud**. Иногда это делается из соображений безопасности: имя группы совпадает с именем пользователя.

Теперь вернемся к правам доступа к файлам. Первая тройка символов относится к пользователю, который является **владельцем** файла. Как правило, это тот пользователь, который создал данный файл. Вторая тройка символов относится к **группе пользователей** (или просто группе), в ко-

торую владелец данного файла может входить, а может и не входить. Последняя тройка включает всех остальных пользователей (т. е. тех, которые не являются владельцем файла и не входят в указанную группу). Права доступа бывают трех видов: право на запись в файл, право на чтение из файла и право на выполнение этого файла. Есть еще модификации этих прав, но в данный момент мы о них говорить не будем. Если право предоставлено, то оно обозначается одним из символов r, w и x (чтение, запись и выполнение соответственно). Если право не предоставлено, то в соответствующей позиции стоит символ «-». Для каталога право на выполнение (x) означает возможность поиска файлов в нем. Для назначения прав доступа к файлам и каталогам используется команда операционной системы **chmod**. Данная команда вызывается таким образом:

```
chmod режим_доступа имя_файла
```

Режим доступа может указываться двумя способами: абсолютным или символическим. Они равноправны, хотя и не во всех случаях одинаково удобны. Мы рассмотрим абсолютный способ, поскольку он кажется более простым в использовании для новичка. Итак, каждому праву на доступ к файлам ставится в соответствие восьмеричное число согласно приведенному ниже правилу.

Права для владельца файла:

0400	владелец имеет право на чтение из файла
0200	владелец имеет право на запись в файл
0100	владелец имеет право на исполнение файла; если это каталог, то владелец имеет право на поиск файлов в каталоге

Права для группы пользователей:

0040	члены указанной группы имеют право на чтение из файла
0020	члены указанной группы имеют право на запись в файл
0010	члены указанной группы имеют право на исполнение файла; если это каталог, то члены группы имеют право на поиск файлов в каталоге

Права для всех остальных пользователей:

0004	они имеют право на чтение из файла
0002	они имеют право на запись в файл
0001	они имеют право на исполнение файла; если это каталог, то они имеют право на поиск файлов в каталоге

Право на запись в каталог означает возможность создавать файлы в этом каталоге, а также удалять их из него.

Для формирования режима доступа к файлу в абсолютной форме нужно сложить восьмеричные значения для элементарных прав. Например:

0400 + 0200 + 0100 + 0040 + 0010 + 0004 + 0001 = 0755,

что означает права на чтение, запись и выполнение для владельца файла, права на чтение и выполнение для членов группы и остальных пользователей. Команда, которая выполнит такое назначение прав для файла, например, **abc.pl**, будет такой (ноль в начале числа можно не указывать):

```
chmod 755 abc.pl
```

Теперь мы можем вернуться к описанию колонок вывода команды **ls -l**. Вторую колонку мы пока пропустим. Третья и четвертая колонки показывают имена пользователя и группы, пятая – размер файла в байтах, затем идут день и месяц создания файла, в следующей колонке выводится время создания файла или год создания (это определяется тем, как давно был создан файл: для файлов, созданных менее полугода назад, выводится время создания, а для более старых файлов – год создания). Последняя колонка – имя файла.

3.2. Понятие файловой системы в операционной системе UNIX

В ОС UNIX, в отличие от MS-DOS и Windows, нет понятия логического диска (например, A:, C: и т. д.). Однако понятие **корневого каталога** также существует. С него мы и начнем. Это самый верхний уровень в иерархии файловой системы. Корневой каталог обозначается символом «/». Обратите внимание на наклон этого символа: он наклонен не в ту же сторону, как в ОС MS-DOS, а в противоположную. Вообще, в ОС UNIX используется именно такой символ для разделения имен каталогов в пути к файлу. Это принципиально важно.

Следующим важным понятием является понятие **файловой системы**. Это совокупность файлов, которая рассматривается как единое целое. Конечно, такое определение является несколько упрощенным, но на данном этапе оно нас устроит. При инсталляции ОС UNIX может быть создано несколько файловых систем, если это необходимо. В нашей с вами системе FreeBSD, которая является полноценным представителем мира UNIX, как правило, создаются следующие файловые системы:

– корневая файловая система (/);

– файловая система для домашних каталогов пользователей (**/home**);

– файловая система для хранения программ, как системных, так и пользовательских (**/usr**);

– файловая система для хранения различной информации, которая накапливается в процессе работы операционной системы (**/var**).

В скобках указаны так называемые **точки монтирования** файловых систем. В ОС UNIX для включения какой-либо файловой системы в общее дерево, берущее начало от корневого каталога, используется специальная команда **mount**, которая выполняет операцию **монтирования** файловой системы. Можно просмотреть смонтированные файловые системы с помощью команды **mount**. Введите ее в командной строке и вы получите примерно такой результат:

```
/dev/ad2s4a on / (ufs, local)
devfs on /dev (devfs, local)
/dev/ad2s4d on /home (ufs, local, soft-updates)
/dev/ad2s4e on /usr (ufs, local, soft-updates)
/dev/ad2s4f on /var (ufs, local, soft-updates)
```

В этих строках первая колонка означает так называемый **файл устройства**, который связан с данной файловой системой. Очень важно знать, что в ОС UNIX все внешние устройства компьютера, в том числе и жесткие диски, рассматриваются как обычные файлы. Во второй колонке указан каталог, на который выполняется операция монтирования. Для того чтобы попасть в эту файловую систему, нужно просто перейти в ее каталог с помощью обычной команды **cd**. Например

```
cd /usr
```

В скобках указан тип файловой системы, например, **ufs** – тип файловой системы, применяемой в FreeBSD. Слово **local** означает, что эта система находится на локальном диске вашего компьютера, а не на другом компьютере (в ОС UNIX есть возможность смонтировать и файловую систему, находящуюся на другом компьютере).

Для того чтобы посмотреть степень заполнения файловых систем, введите команду **df**. Вы получите примерно такой вывод на экран:

```
Filesystem  1K-blocks    Used    Avail Capacity  Mounted on
/dev/ad2s4a    198126   67688   114588    37%    /
devfs               1       1        0   100%    /dev
/dev/ad2s4d   1012974      24   931914     0%    /home
/dev/ad2s4e  17851758 1287242 15136376     8%    /usr
/dev/ad2s4f    724612    2042   664602     0%    /var
```

В первой колонке также указывается имя файла устройства, связанного с этой файловой системой. В нашем примере ad2 – это номер диска, у вас может быть и другой номер, например, ad10. Номер диска зависит от номера контроллера на плате, к которому вы подсоединили диск.

Во второй колонке – размер файловой системы, выраженный в блоках по 1 Кб. Далее указаны использованный (занятый) и оставшийся объемы. Под заголовком **Capacity** указана степень занятости в процентах. Парадокс заключается в том, что иногда эта величина может быть более 100 процентов, а объем оставшегося пространства может выражаться отрицательным числом. Объясняется это тем, что файловая система имеет некоторый резерв, который не используется до определенного времени. В последней колонке указана точка монтирования. Соотношения объемов файловых систем определяется исходя из потребностей конкретной инсталляции. Как правило, корневую файловую систему не делают слишком большой, для нее достаточно 120 Мб. Более подробные рекомендации относительно определения размеров файловых систем были приведены в главе 1.

3.3. Еще о привилегиях доступа к файлам

Перейдите в каталог **/etc** и найдите в нем файлы **passwd** и **master.passwd**. Попытайтесь их просмотреть. Вы увидите, что доступа к файлу **master.passwd**, в котором хранятся пароли пользователей системы, у пользователя **stud** нет. Однако любой пользователь системы, в том числе и **stud**, имеет возможность сменить свой пароль (как это сделать, будет показано позднее), который записывается в этот самый файл **master.passwd**. Парадокса тут нет. Доступ к файлу паролей осуществляется не непосредственно, а через программу **passwd**, которая и производит запись нового пароля в этот файл. Найдите эту программу в каталогах системы (с помощью программы **whereis**) и перейдите в каталог, где она находится. Просмотрите информацию о ней с помощью команды

```
ls -l passwd
```

Вы увидите вместо буквы «x» в привилегиях доступа для владельца, которым является суперпользователь **root**, букву «s», также вы увидите, что право на запуск этой программы имеют все пользователи системы, а, значит, и **stud**. Эта буква «s» означает, что при запуске данной программы пользователем **stud** она оказывается запущенной как бы от имени суперпользователя **root**, который, как вы уже видели в каталоге **/etc**, имеет право на запись в файл паролей **master.passwd**. Такая возможность достигается за счет так называемого **бита смены идентификатора пользователя**. Существует аналогичный **бит смены идентификатора группы**. Для назначения такой комбинации прав доступа, как у программы **passwd**, нужно в команде **chmod** указать значение 4555, например:

```
chmod 4555 my_program
```

Обратите внимание на то, что раньше мы использовали только три цифры в этой команде, а в данном случае нужно использовать четыре цифры. Первая как раз и назначает этот самый «волшебный» бит. Для бита смены идентификатора пользователя служит значение 4, а для бита смены идентификатора группы – 2. При назначении обоих битов нужно указать значение 6. Впоследствии мы еще вернемся к данному вопросу, поэтому, если вы не все до конца поняли, у вас еще будет возможность лучше разобраться в тонкостях дела.

Для полноты картины упомянем программы **chown** и **chgrp**, которые служат для смены владельца файла и владельца группы. В данный момент вы не сможете использовать эти программы, т. к. для этого нужны полномочия суперпользователя **root**. Их использование мы изучим в последующих главах, в которых вы поработаете с правами суперпользователя **root**.

3.4. Ссылки

В ОС UNIX есть так называемые **ссылки**. Причем они бывают двух видов: жесткие и символические. Жесткая ссылка – это, попросту говоря, еще одно имя файла. В ОС UNIX информация о файлах хранится не в каталогах, как в MS-DOS, а в специальных структурах, которые называются **inodes**, или **индексные дескрипторы**. В каталоге же хранится только имя файла и номер его inode. Поэтому существует возможность создать имена в разных каталогах (и в одном тоже), которые ссылаются на один и тот же inode, т. е. на один и тот же файл. Это бывает удобно в ряде случаев, когда простое копирование файлов не подходит. Для создания жестких ссылок используется команда **ln**:

```
ln исходный_файл новая_ссылка
```

Например:

```
ln my_file my_file.2
```

Проделайте это с каким-нибудь вашим файлом, а затем введите команду

```
ls -il
```

Вы увидите, что в выводе команды **ls** добавилась одна колонка. Теперь в первой колонке указаны номера индексных дескрипторов для файлов данного каталога. Обратите внимание, что у двух ваших файлов эти номера одинаковые. Однако это не две копии файла. Это легко проверить, изменив один из этих файлов, а затем, просмотрев другой: вы увидите, что изменения отражены в обоих. Еще одно новшество: в третьей колонке (пе-

ред именем пользователя) вы увидите значение 2. Это число жестких ссылок, которое возросло до двух после создания вами новой ссылки. Жесткие ссылки можно создавать, указывая на файл из другого каталога.

Другая разновидность ссылок – символические. Для их создания нужно указать параметр **-s** в команде **ln**:

```
ln -s my_file my_file.3
```

Они являются специальными файлами, в которых просто хранится имя того файла, на который указывает данная ссылка. Они не являются копиями файлов. У них номера индексных дескрипторов уже свои, не совпадающие с номерами у исходных файлов. Проверьте это с помощью команды

```
ls -il
```

Внимательно изучите информацию, выводимую этой командой на экран. Обратите внимание на букву «l» (латинская строчная буква «el») в колонке прав доступа к файлу. Она указывает на тип файла – символическая ссылка. Если жесткие ссылки можно создавать только в пределах одной файловой системы, например, **/usr**, то символические такого ограничения не имеют. Это важное их свойство.

Контрольные вопросы и задания

1. Отобразите сведения о правах доступа к файлам какого-либо каталога и расскажите о них. Что бит выполнения означает в применении к каталогам?

2. Каким образом можно выяснить имя владельца файла или каталога?

3. Назначьте одному из ваших файлов такие права, чтобы владелец имел полный доступ к файлу, пользователи группы могли читать файл и запускать его на выполнение, а для всех остальных пользователей файл был бы недоступен.

4. Смените права доступа у вашего каталога так, чтобы владелец мог производить чтение, запись и поиск в этом каталоге, а члены группы и другие пользователи – только чтение и поиск.

5. Перечислите существующие файловые системы в операционной системе и кратко объясните их назначение.

6. Просмотрите степень заполнения файловых систем.

7. Что такое ссылки? В чем отличие между жесткими и символическими ссылками? Как создать символическую ссылку? Есть ли разница при обращении к файлу по различным именам, которые являются жесткими ссылками?

8. Если для символической ссылки изменить права доступа к файлу, то изменяться ли права доступа к файлу, на который ссылается символическая ссылка? Создайте символическую ссылку на файл или на каталог, затем удалите их.

9. Можно ли создать жесткую ссылку на каталог? Каким образом можно узнать количество жестких ссылок на файл?

10. Что такое бит смены идентификатора владельца (группы)? Приведите пример его использования в ОС UNIX.

11. Что такое монтирование файловой системы? Как оно выполняется?

12. Найдите программу **passwd** и объясните права доступа для нее.

13. Что такое индексные дескрипторы inodes? Отобразите на экране номера индексных дескрипторов для файлов в каталоге?

4. Структура каталогов в операционной системе FreeBSD

Для дальнейшего продвижения вперед необходимо познакомиться со структурой каталогов одного из представителей мира UNIX – операционной системы FreeBSD, чтобы лучше ориентироваться в системе и знать, где находятся некоторые очень важные файлы и каталоги.

Для того чтобы вы получили представление о структуре каталогов ОС FreeBSD, мы пройдем по всему дереву, начиная с корневого каталога **/**.

Исследуем основные каталоги, находящиеся в корневом каталоге. Начнем с каталога **/bin**. Обратите внимание на наличие символа «/». Он указывает на то, что путь к каталогу является абсолютным, т. е. он прослеживается от корневого каталога. Этот каталог помещается непосредственно в корневой файловой системе. В нем находятся основные системные команды. Для нас интересной будет команда **ps**. Она позволяет просмотреть список процессов, выполняющихся в системе в данный момент. Для этого выполните команду

```
ps -ax
```

В первой колонке указан номер процесса, а в последней – команда операционной системы, с помощью которой был порожден этот процесс. Существует целый ряд процессов, которые порождаются ядром ОС. В списке процессов они располагаются ближе к его началу. В последней колонке для них указана не команда операционной системы, а ключевое название процесса в квадратных скобках.

Каталог **/boot** содержит файлы, используемые при начальной загрузке ОС. Для того чтобы определить тип какого-либо файла в ОС UNIX, можно использовать удобную команду **file**. Введите:

```
file /boot/boot
```

и вы получите информацию о типе файла.

```
mbr: x86 boot sector
```

Введите

```
file /boot/mbr
```

Команда **file** показывает, что это тоже загрузочный сектор (вывод не умещается на одной строке):

```
boot: x86 boot sector; partition 4:ID=0xa5, active, starthead
0 startsector 0, 50000 sectors, code offset 0x3c
```

В подкаталоге **/boot/kernel** находится файл **kernel**. Это – **ядро операционной системы**, т. е. ее главный исполняемый файл.

Каталог **/cdrom** используется для монтирования дисковода CD/DVD дисков. О том, как смонтировать такой диск, говорилось в главе 2.

Каталог **/dev** содержит файлы устройств. В более старых версиях ОС FreeBSD для создания файлов устройств использовалась специальная программа. Но в версии FreeBSD 8.2 для управления файлами устройств служит специальная файловая система **/dev**.

Каталог **/etc** содержит файлы конфигурации ОС FreeBSD. Кратко познакомимся с основными из них. Рекомендуем вам просматривать каждый файл. Детально изучать их пока что нет необходимости, но беглый просмотр будет очень полезен.

В подкаталоге **/etc/defaults** находятся версии основных конфигурационных файлов операционной системы, которые используются по умолчанию.

В подкаталогах **/etc/periodic** и **/etc/rc.d** находятся системные программы, написанные на языке shell.

Теперь кратко опишем назначение основных конфигурационных файлов, хранящихся с каталоге **/etc**.

crontab – файл для задания времени выполнения периодически повторяющихся заданий в системе (например, создание архивных копий базы данных).

fstab – файл, содержащий список файловых систем, которые монтируются (т. е. подключаются) при загрузке ОС.

group – список пользовательских групп, существующих в системе. Обратите внимание на группу **wheel**. Это группа привилегированных пользователей. Есть также и группа **stud**. Это та группа, в которую входит ваш идентификатор (имя) пользователя. Конечно, если ваша ОС настроена не совсем так, как мы описывали в главе 2, то группы **stud** может и не быть, но группа **wheel** есть всегда.

hosts – файл, который в ряде ситуаций используется для организации компьютерной сети.

inetd.conf – файл, управляющий сетевыми сервисами ОС.

master.passwd – файл паролей. Доступ к нему разрешен только суперпользователю системы, который имеет имя **root**.

profile – конфигурационный файл, устанавливающий различные параметры, которые будут общими для всех пользователей системы, использующих командный процессор (shell) **/bin/sh**.

rc – командный файл (по-другому такие файлы называют **скриптами**), который выполняется при начальной загрузке системы. Это очень важный файл.

rc.conf – конфигурационный файл, который используется при начальной загрузке системы. Это также очень важный файл, который содержит, например, ряд параметров, необходимых для «русификации» системы.

syslog.conf – конфигурационный файл для подсистемы сбора информации о различных событиях, процессах и аварийных ситуациях, происходивших в системе.

ttys – конфигурационный файл для настройки виртуальных терминалов (см. главу 2).

Каталог **/home** содержит домашние каталоги пользователей ОС FreeBSD. Имена каталогов совпадают с именами пользователей.

Каталог **/mnt** создается в системе всегда. Он служит обычно в качестве временной точки монтирования файловых систем, например, можно к нему примонтировать файловую систему FAT32 или NTFS, используемые в ОС Windows. В этом случае, перейдя в каталог **/mnt**, мы получим возможность видеть файловую систему ОС Windows.

Каталог **/sbin** содержит различные системные команды ОС UNIX. Можно использовать команду **man** для получения справки по всем этим командам. Например, введите

```
man mount
```

и вы получите справку по команде монтирования файловых систем.

Каталог **/sys** содержит исходные тексты программ, из которых формируется ядро ОС FreeBSD. Если вы войдете в этот каталог, то текущим станет не каталог **/sys**, а каталог **/usr/src/sys**, потому что **/sys** является так называемой ссылкой на каталог **/usr/src/sys.**

Каталог **/tmp** содержит временные файлы, создаваемые в процессе работы операционной системы. Администраторы операционной системы зачастую удаляют его и вместо него создают символическую ссылку с этим же именем **/tmp** на каталог **/var/tmp**. Это делается для того, чтобы избежать накопления временных файлов в корневой файловой системе.

Каталоги **/usr** и **/var** содержат множество самых разнообразных подкаталогов, которые мы сейчас рассмотрим.

Как вы помните из предыдущей главы, эти каталоги являются точками монтирования файловых систем. Конечно, разные системные администраторы могут создать различные файловые структуры на дисках своих компьютеров, но в данном случае мы имеем в виду типовую конфигурацию, пригодную, по крайней мере, для начального изучения операционной системы FreeBSD. Итак, заходим в каталог **/usr** (команда: **cd /usr**). Первый каталог, который мы видим – **X11R6.** Это символическая ссылка, в нем располагается система **X Window**, с которой вы познакомитесь позднее, а также много других программ. В его подкаталоге **bin** находятся выполняемые файлы – различные программы.

В подкаталоге **man** находятся электронные руководства по системе **X Window**, которые вызываются по команде **man**.

Вернитесь в каталог **/usr** и перейдите в его подкаталог **bin**. В нем много программ, в том числе и знакомые вам **more, uname**, компилятор языка С – **cc**. В этом каталоге есть не только «настоящие» программы (т. е. программы в машинном коде), но также и скрипты, например, **apropos**. Просмотрите его содержимое – это текстовый файл. Строки, начинающиеся с символа #, являются комментариями. Кстати говоря, если бы мы захотели узнать, где находится программа **apropos** (или любая другая программа), мы могли бы воспользоваться командой **whereis**:

```
whereis apropos
```

Она выдаст нам следующее сообщение:

```
apropos: /usr/bin/apropos /usr/share/man/man1/apropos.1.gz
```

В этом сообщении указан полный путь не только к самому скрипту **apropos**, но также и к электронному руководству (вызываемому по команде **man apropos**). В имени файла руководства **apropos.1.gz** цифрой 1 указан номер раздела электронного руководства, в котором помещено описание команды **apropos**. Символы **gz** указывают на то, что этот файл является сжатым при помощи команды **gzip** (о ней мы расскажем позднее). Скажем попутно и о самой команде **apropos**. Она позволяет выявить имена всех электронных руководств, в которых упоминается то или иное слово или выражение. Если вас интересует, например, процесс начальной загрузки операционной системы, то попробуйте ввести

```
apropos boot
```

Следующий подкаталог в каталоге **/usr** – **include**. В нем находятся заголовочные файлы, например, всем известный файл **stdio.h**.

Следующий подкаталог в каталоге **/usr** – **lib**. В нем находятся библиотеки компилятора С, который устанавливается в системе FreeBSD по умолчанию. Причем имеются их версии как для статической, так и для динамической компоновки (последние имеют расширение «.**so**»).

Следующий интересующий нас подкаталог в каталоге **/usr** – **libexec**. В нем находятся так называемые **программы-демоны**. Можно с некоторой долей условности сказать, что они являются аналогами резидентных программ в ОС MS-DOS. Эти программы запускаются при старте операционной системы. Ссылки на них можно увидеть в файле **/etc/inetd.conf** (не забудьте, если путь к файлу начинается с символа «/», то это абсолютный путь, т. е. путь от корневого каталога «/»). Некоторые из этих программ запускаются не при старте системы, а в других случаях.

Следующий интересующий нас подкаталог в каталоге **/usr** – **local**. В нем находятся в основном программы, устанавливаемые пользователями системы (точнее, устанавливаемые администратором, имеющим доступ к системе с правами пользователя **root**). Здесь есть уже традиционный подкаталог **bin**, в котором вы увидите знакомые вам программные файлы **deco** и **joe**. В подкаталог **lib** (полный путь – **/usr/local/lib**) устанавливаются, как правило, дополнительные библиотеки для различных программных продуктов (например, пакеты языка Perl, о котором мы будем говорить в одной из следующих глав), а также конфигурационные файлы.

Следующий интересующий нас подкаталог в каталоге **/usr/local** – **man**. В нем находятся электронные руководства (manual pages) для тех программ, которые установлены в каталог **/usr/local/bin.**

Загляните также и в подкаталог **etc** в каталоге **/usr/local** (полный путь – **/usr/local/etc**). В нем находятся различные конфигурационные файлы, например, **joerc** и **lynx.cfg.**

Возвращаемся в каталог **/usr** и переходим в его подкаталог **sbin**. В нем находится множество системных программ, в том числе и программы для создания и удаления учетных записей пользователей системы: **adduser** и **rmuser**. Они интересны потому, что написаны на языке shell. Посмотрите их при помощи программы **more** (команда: **more adduser**) или просто нажав клавишу F3, если вы работаете в файловом менеджере **deco**. Это весьма содержательные программы, в них будет чему поучиться, когда мы приступим к ознакомлению с языком shell.

Возвращаемся в каталог **/usr** и переходим в его подкаталог **share**. В нем находится множество подкаталогов. Не поленитесь заглянуть в каждый из них: там много интересных и полезных вещей. Перечислим некоторые из этих подкаталогов.

calendar – различные календарные даты, в т. ч. русские праздники, важнейшие исторические события, важнейшие даты из компьютерной истории и т. д.

dict – словари английских слов. Осторожнее: тут есть огромные файлы, которые программа для просмотра текстовых файлов, встроенная в файловый менеджер **deco,** осилить не сможет. Поэтому лучше воспользуйтесь командой **more** или редактором **joe.**

doc – техническая документация**.** Если вы увидите файлы с расширением «**.gz**», то для их просмотра можно использовать команду **zcat** в сочетании с командой **more**, вызванной с использованием «конвейера» (символ «|»), например:

```
zcat malloc.ascii.gz | more
```

examples – примеры программ и конфигурационных файлов на все случаи администраторской жизни.

info – документация в формате info. Для просмотра используйте команду **info -f имя_info_файла**, например:

```
info -f dc.info.gz
```

mk – make-файлы, которые используются при компиляции различных системных программ, в том числе при конфигурировании ядра операционной системы.

skel – файлы-шаблоны, используемые при конфигурировании пользователей системы (это прототипы файлов **.profile** и им подобных, которые вы можете увидеть в вашем домашнем каталоге **/home/stud**).

Снова возвращаемся в каталог **/usr** и переходим в его подкаталог **src.** В нем размещаются исходные тексты операционной системы (если они были установлены в процессе ее инсталляции). В каталоге **/usr/src/sys** находятся исходные тексты ядра системы. Это очень важный каталог. При инсталляции системы его нужно устанавливать ОБЯЗАТЕЛЬНО.

Теперь рассмотрим каталог **/var**. Он содержит, в основном, информацию, которая изменяется в процессе функционирования операционной системы и отражает ее текущее состояние как в краткосрочном, так и в долгосрочном плане. Перечислим некоторые из подкаталогов.

db – здесь перечислены программные продукты, установленные из так называемых пакетов (packages), т. е. уже скомпилированных модулей. В каждом из подкаталогов, посвященных конкретному пакету, есть несколько текстовых файлов. В них приводится описание программы, указаны каталоги операционной системы, в которые установлены файлы этого программного продукта и т. п.

log – здесь находятся файлы-журналы, которые формируются в процессе работы различных системных программ. Посмотрите файлы **maillog** (в нем отражаются операции электронной почты) и **messages** (в нем отражаются процедуры загрузки и основные системные события, такие, например, как регистрация суперпользователя в системе или запуск системных программ). Есть в этом каталоге и файлы с расширениями «**.gz**» и цифрами. Это прежние версии файлов-журналов, которые периодически архивируются системой в автоматическом режиме (настройками этих режимов можно управлять из каталога **/etc**).

mail – здесь находятся файлы, выполняющие функции почтовых ящиков пользователей. Их имена совпадают с именами пользователей, работающих в системе. Выполните команду **ls -l** в каталоге **/var/mail** и посмотрите на права доступа и имена владельцев этих файлов. Как вы видите, доступ к файлам имеют только их владельцы.

run – в этом каталоге среди прочих находятся файлы с расширением «**.pid**». В них записаны номера самых важных процессов, которые работают в системе в настоящий момент. Посмотрите даты и время создания этих файлов с помощью команды **ls -l.**

tmp – здесь находятся временные файлы, используемые различными программами.

И последний каталог, который мы рассмотрим – **/usr/local/share/doc/freebsd**. В нем представлена техническая документа-

ция, в том числе на русском языке в подкаталоге **ru** или **ru_RU.KOI8-R**. В подкаталоге **handbook** находится полное руководство по системе FreeBSD (на английском языке), а в подкаталоге **ru_RU.KOI8-R/books/handbook** это же руководство представлено на русском языке.

На этом наш краткий обзор структуры каталогов ОС FreeBSD завершен. Дальнейшее углубленное изучение упомянутых в этой главе терминов, файлов, программ будет продолжено в следующих главах.

Контрольные вопросы и задания

1. Как называется и где находится файл ядра операционной системы FreeBSD?

2. Объясните назначение файлов из каталога **/etc**: **rc.conf**, **inetd.conf**, **ttys**, **profile**.

3. С помощью какой программы можно проводить конфигурирование системы после инсталляции, например, установку различных пакетов?

4. В каком каталоге находятся исходные тексты ядра ОС?

5. Каким образом можно определить тип любого файла в ОС UNIX?

6. Для чего нужна команда **whereis**?

7. Какие файлы хранятся в каталоге **/dev**?

8. Выведите на экран список всех выполняющихся процессов.

5. Работа с файлами в ОС FreeBSD

В этой главе показаны основные приемы работы с файлами в ОС FreeBSD. Прочитав ее, вы узнаете:

– методы архивирования и сжатия файлов;

– способы поиска файлов в каталогах файловой системы и поиска символьных строк внутри файлов, содержащихся в определенных каталогах;

– способы просмотра длинных текстовых файлов и сравнения файлов с целью выявления отличий.

5.1. Архивирование и сжатие файлов

Архивирование и сжатие файлов – это не одно и то же в системе UNIX (как, наверное, и в других ОС). Архивирование – это создание резервных копий, а сжатие – уменьшение объема файлов.

Для сжатия файлов используется программа **gzip**. Пользоваться ею просто:

```
gzip имя_сжимаемого_файла
```

Например:

```
gzip myfile.txt
```

В результате получим вместо исходного файла его сжатый вариант с расширением «**.gz**», в данном случае – **myfile.txt.gz**.

Для выполнения обратной процедуры (извлечения файла) есть несколько способов, один из них – использование этой же программы **gzip**, но с параметром **-d**. Например:

```
gzip -d myfile.txt.gz
```

В результате получим вместо сжатого файла его несжатый вариант уже без расширения «**.gz**», в данном случае – **myfile.txt**.

Иногда бывает необходимо просмотреть сжатый текстовый файл, не разворачивая его в исходное состояние. В этом случае можно воспользоваться программой **zcat**. Для нашего условного примера это будет так (поскольку файл может быть большим, то используйте команду **more** через «конвейер»):

```
zcat myfile.txt.gz | more
```

Создайте какой-нибудь текстовый файл и проделайте на нем все эти операции.

Существует еще одна программа для сжатия файлов – **compress**. Она используется аналогично программе **gzip**, но добавляет расширение «**.Z**». Для просмотра таких файлов можно также использовать программу **zcat**. Например:

```
zcat myfile.txt.Z
```

Чтобы заархивировать файлы, в ОС UNIX традиционно используется программа **tar**. В стародавние времена с ее помощью записывали резервные копии файлов на магнитную ленту (что отражено в названии программы – **tape archiver**). Чтобы заархивировать все файлы в каталоге (включая и подкаталоги), нужно ввести команду:

```
tar cvf имя_создаваемого_архива *
```

Символ * означает – все файлы. При использовании программы **tar** желательно указывать имя архива с расширением «**.tar**», т. к. сама программа (в отличие от **gzip** и **compress**) расширение не добавляет. Расширение поможет вам легче узнавать **tar**-архивы при просмотре каталогов. Например:

```
tar cvf my_archive.tar *
```

Если вы забыли указать расширение, то можно потом переименовать архив командой **mv**. Вспомните и о программе **file**, которая поможет вам узнать тип файла.

Можно при использовании программы **tar** еще и сжимать файлы. По умолчанию **tar** только собирает файлы текущего или указанного каталога в один общий файл, но объем этого архива будет даже чуть больше суммы объемов входящих в него файлов (из-за наличия в архиве служебной информации). Не спешите смеяться: бывают ситуации, когда и такая форма использования **tar** оказывается полезной. Однако **tar** умеет и сжимать архивы. Для этого нужно указать в его параметрах запуска параметр «**z**», а расширение архиву дать такое – «**.tgz**», что означает **tar + gzip**. Для сжатия вызывается программа **gzip** неявным образом. Например:

```
tar czvf my_archive.tgz *
```

Чтобы просмотреть содержимое несжатого **tar**-архива, сделайте так:

```
tar tvf my_archive.tar
```

Чтобы просмотреть содержимое сжатого **tar**-архива, сделайте так:

```
tar tzvf my_archive.tgz
```

Чтобы извлечь содержимое несжатого **tar**-архива, сделайте так:

```
tar xvf my_archive.tar
```

Чтобы извлечь содержимое сжатого **tar**-архива, сделайте так:

```
tar xzvf my_archive.tgz
```

5.2. Полезные команды для работы с большими файлами

Мы рассмотрим несколько полезных команд для работы с файлами большого размера. Начнем с команды **tail**. Выполните следующую команду:

```
tail -n 5 myfile
```

Здесь **myfile** – это имя какого-нибудь файла из текущего каталога. Вы увидите, что на экран будут выведены 5 последних строк этого файла. Если же ввести команду **tail** без параметра, например:

```
tail myfile
```

то по умолчанию выводятся 10 последних строк файла.

Следующая команда – **head** – позволит нам вывести несколько первых строк файла. Ее синтаксис аналогичен команде **tail**. Для вывода 5 первых строк введите

```
head -n 5 myfile
```

По умолчанию выводятся 10 первых строк файла:

```
head myfile
```

Эти команды можно соединять при помощи «конвейера». Например, если нам нужно выбрать из файла строки с 5-й по 7-ю, то мы можем поступить таким образом:

```
head -n 7 myfile | tail -n 3
```

Есть еще одна полезная команда – **split**. Она умеет разделять файл на части требуемой длины. Длина фрагментов может выражаться как в байтах, так и в строках (для текстовых файлов). Разделим файл **myfile** на фрагменты по 100 байтов:

```
split -b 100 myfile
```

Обратите внимание на имена полученных файлов-фрагментов. Для разделения файла на фрагменты по 10 строк поступим так:

```
split -l 10 myfile
```

Иногда бывает необходимо выполнить обратную задачу – «склеить» файлы (т. е. произвести их конкатенацию). Для этого используют команду **cat** таким образом:

```
cat file1 file2 file3 > file_result
```

Таким образом, можно соединять как текстовые, так и двоичные файлы.

Следующая команда – **sort**. Она позволяет отсортировать файл. Эта команда имеет богатые возможности, которые вы можете изучить сами по электронному руководству (**man sort**). Мы же выполним только простейшую сортировку:

```
sort myfile
```

Отсортированный файл направляется на стандартное устройство вывода, т. е. на экран. При необходимости можно перенаправить его в файл:

```
sort myfile > newfile
```

Очень полезная команда **diff**, которая позволяет сравнить два файла. Создайте небольшой текстовый файл – строк 10–12. Сделайте его копию, а затем в файле-копии удалите одну строку из середины файла, а еще в одной строке измените что-нибудь. Теперь выполните команду:

```
diff myfile myfile_copy
```

После этого поменяйте местами имена файлов в командной строке:

```
diff myfile_copy myfile
```

Посмотрите, в чем разница между выводами двух этих вариантов команд. Символы «d», «c», «a» означают, что для приведения этих файлов к одинаковому состоянию нужно строку номер такой-то **удалить**, строку номер такой-то **заменить** строкой номер такой-то, а строку номер такой-то **добавить** после строки номер такой-то. Внимательно посмотрите содержимое своих файлов, и вам будет понятно, что сообщает команда **diff.**

Если команда **diff** применяется для сравнения двоичных файлов, то в случае наличия различий в них команда выведет на экран лишь сообщение:

```
Binary files .... differ
```

5.3. Поиск файлов и поиск символьных строк в файлах

Часто бывает нужно найти тот или иной файл в файловой системе. В этом поможет команда **find**. Это очень мощная команда, но мы изучим пока что только малую часть ее возможностей (для любознательных совет все тот же: **man find**). Итак, чтобы найти файл с именем, например, **profile**, введите

```
find / -name profile -print
```

В этой команде символ «/» обозначает корневой каталог – мы начинаем поиск с него. Если требуется выполнить поиск, начиная не с корневого каталога, то вместо символа «/» укажите путь к этому каталогу. Поиск производится во всех подкаталогах. Следующий параметр: **name**. Он указывает имя искомого файла. Параметр **print** предписывает команде **find** вывести имена найденных файлов на экран. После запуска вышеуказанной команды на экран время от времени будут выводиться сообщения:

```
find: .... Permission denied
```

Они появляются потому, что у пользователя **stud** нет прав на чтение некоторых каталогов. Чтобы этих сообщений не было, нужно поступить так (пробел между цифрой 2 и символом > ставить НЕ НУЖНО):

```
find / -name profile -print 2>/dev/null
```

Эта команда требует некоторого пояснения. Как известно (это справедливо для ОС MS-DOS и UNIX), существуют устройства стандартного ввода (**stdin**), вывода (**stdout**) и ошибок (**stderr**). Они имеют номера дескрипторов файлов соответственно 0, 1 и 2. В ОС UNIX есть возможность отделить сообщения об ошибках от стандартного вывода. Именно это мы и проделываем в модифицированной версии команды для поиска файла: мы перенаправляем сообщения об ошибках в файл с именем **/dev/null**. Этот файл является файлом специального устройства, которое просто поглощает выводимые сообщения, не позволяя им попасть на экран. Наша модифицированная команда выводит на экран только полезную информацию. Можно еще модифицировать эту команду, перенаправив ее стандартный вывод (т. е. полезную информацию о найденных файлах) также в файл:

```
find / -name profile -print 2>/dev/null > file_names
```

Команда find рекурсивно обходит указанные каталоги и файлы, проверяет для них выполнение указанных условий и может вдобавок выполнять с найденными файлами указанные действия.

ПРИМЕЧАНИЕ. Пример некоторых ключей команды **find.**

-atime *N*	Последний раз к файлу обращались *N* дней назад.
-mtime *N*	Последнее изменение файла было *N* дней назад.
-newer *другой_файл*	Файл был изменен позже, чем *другой_файл*.
-size [±]*N*[cwbkMG]	Размер файла равен *N* блокам, если указано *+N*, тогда размер файла больше *N*, *-N* — меньше. Символ после *N* означает размер блока. *b* — 512 байт, *c* — байт, *w* — 2 байта, *k* — килобайт, *M* — мегабайт, *G* — гигабайт.

Ключи-условия команды **find**:

-size +500	- файлы размером БОЛЬШЕ 500*512 байт
-mtime -3	- дата модификации МЕНЬШЕ 3-х дней
-newer filename	- дата модификации нашего файла МЕНЬШЕ, чем у заданного файла filename

---

Например, уничтожить все файлы с окончаниями *.bu, *%, которые не менялись больше месяца.

```
find   / \( -name "*.bu" -o -name "*%" \) -type f\
     -atime +30 -exec  rm  {} \;
```

Запись "rm {} \;" - обозначает команду, которая будет выполняться для всех таких найденных файлов. Вместо значка "{}" будет подставляться каждый раз имя найденного файла. Естественно, что таким образом мы их все и уничтожим.

И еще одна очень полезная команда – **grep**. Она позволяет выполнять поиск информации в файлах. Например, мы ищем программы, написанные на языке Perl, которые находятся в каталоге **/usr/sbin** (прежде чем вводить следующую команду, перейдите в этот каталог):

```
grep "/bin/perl" *
```

Строка «/bin/perl» является искомой, а звездочка означает – искать во всех файлах текущего каталога. На экран будут выведены строки, в которых присутствует данная строка в виде подстроки. Можно вместо звездочки указать шаблон имен файлов (аналогично тому, какой применяется в ОС MS-DOS). Например:

```
grep "/bin/perl" a*
```

или

```
grep "/bin/perl" *user
```

При использовании команды **grep** можно также добавить параметр **2>/dev/null** (пробел между цифрой 2 и символом > ставить НЕ НУЖНО). Чтобы провести поиск таких файлов во всей иерархии **/usr** сделайте так:

```
grep -R "/bin/perl" /usr/*
```

Здесь параметр **-R** указывает на поиск по всем подкаталогам.

И наконец, команда **du**, которая показывает объем дискового пространства, занимаемого каталогом и всеми его подкаталогами (вспомните, что символ «.» обозначает текущий каталог):

```
du .
```

А можно, например, так:

```
du /usr
```

Если переменная среды **BLOCKSIZE** (об этой переменной будет сказано позднее) имеет значение, равное «K», то программа **du** показывает объем в килобайтах. Эта программа имеет параметр, который позволяет при необходимости изменять единицы, в которых ведется счет:

```
du -k .
```

В этом случае будет показан объем в килобайтах, который занимают файлы и подкаталоги текущего каталога (не забудьте поставить символ «.», который означает текущий каталог).

Контрольные вопросы и задания

1. С помощью электронного руководства **man** попытайтесь разобраться с командой **find** (**man find**) и найдите файлы, принадлежавшие заданной группе, например, группе **stud.**

2. С помощью команды **find** найдите все файлы, которые принадлежат одному пользователю, например, **stud.**

3. Познакомьтесь самостоятельно с командой **nohup**, используя электронное руководство.

4. Сравните содержимое двух текстовых файлов, например, старой и новой версий исходного текста программы.

5. Выполните команду, которая осуществляет поиск какой-нибудь фразы во всех файлах данного каталога, включая и все подкаталоги (например, найдите файлы в каталоге **/etc**, в которых есть строка «linux»).

6. С помощью команды **grep** (используя ее параметры) выведите на экран строки, которые встречаются в файле несколько раз и пронумеруйте эти строки.

7. Посмотрите, сколько места занимает на диске какой-нибудь каталог, например, ваш текущий каталог. В каких единицах измерения указан размер каталога? Посмотрите итоговый размер файловой системы **/var.**

8. Выведите на экран 10 последних строк, 10 первых строк текстового файла. Выведите на экран шесть строк из файла: с 4-ой по 9-ую строку.

9. Отсортируйте содержимое файла с помощью команды **sort**, а результат выполнения программы перенаправьте в файл.

10. Разделите файл на фрагменты по 100 байтов, а затем склейте их опять в один файл и сравните полученный файл с исходным с помощью команды **diff.**

11. В чем разница между двумя архивными файлами, если у одного файла расширение «.**tar**», а у другого «.**tgz**»? Заархивируйте и одновременно сожмите ваш файл. Заархивируйте все файлы в текущем каталоге в один архивный файл.

12. Сожмите какой-нибудь файл с помощью команды **gzip**, а затем распакуйте его и сравните исходный файл с распакованным.

6. Текстовый редактор vi

Эта совсем небольшая глава посвящена текстовому редактору **vi.** Можно сказать, что **vi** – легендарный редактор. Мы опишем основные приемы работы в нем.

Этот текстовый редактор является одним из базовых средств ОС UNIX. Знакомство с ним – обязательно, поскольку он присутствует во **всех** версиях ОС UNIX. Бывают случаи, когда в системе не установлено никаких редакторов, кроме этого, так что элементарные знания редактора **vi** вам не повредят. К тому же среди уважающих себя UNIX-гуру считается правилом хорошего тона владение этим редактором.

Несмотря на свою внешнюю невзрачность, редактор **vi** имеет множество возможностей. Мы познакомимся только с самыми основными из них, а именно: ввод текста, удаление текста, запись изменений в файл, выход из редактора. Тем же, кто хочет узнать больше об этом замечательном редакторе, можно дать все тот же совет: используйте электронную справочную систему – **man vi.**

Для запуска редактора и создания нового файла введите:

```
vi my_file.txt
```

Вы увидите черный экран, внизу которого есть строка состояния. Если попытаться просто вводить текст, то у вас ничего не получится. Данный редактор имеет два режима работы: командный и, условно говоря, «рабочий», т. е. режим редактирования. При запуске редактор переходит в командный режим. Чтобы начать вводить текст, нажмите клавишу i. Попробуйте вводить текст, как в обычном редакторе. Попробуйте удалить символ. Внимательно наблюдайте реакцию редактора. Если вы переместите курсор на строку вверх, то редактор сам перейдет в командный режим. Если же вы хотите перевести его в командный режим принудительно, то нажмите клавишу Escape. Иногда не повредит и двукратное нажатие клавиши Escape, если вы не уверены, что редактор уже переведен в командный режим.

Приведем краткую сводку команд редактора:

a ввести текст ПОСЛЕ курсора;

i ввести текст ПЕРЕД курсором;

o ввести пустую строку ПОД строкой, на которой стоит курсор;

O ввести пустую строку НАД строкой, на которой стоит курсор;

dd удалить строку, на которой стоит курсор;

x удалить символ, на котором стоит курсор;

yy скопировать в буфер строку, на которой стоит курсор;

p вставить строку из буфера ПОСЛЕ строки, на которой стоит курсор.

Например, чтобы удалить строку текста, на которой стоит курсор, сначала нужно перейти в командный режим, а уже затем дважды нажимать клавишу d. Однако если вам требуется удалить более одной строки текста, то достаточно однократного перехода в командный режим, после чего вы можете удалять строки, перемещаясь по ним с помощью клавишей управления курсором или команд j и k (см. ниже).

Для перемещения по файлу можно использовать такие команды (в дополнение к клавишам управления курсором):

h переместить курсор ВЛЕВО на один символ;

l (буква «el») переместить курсор ВПРАВО на один символ;

j переместить курсор ВНИЗ на одну строку;

k переместить курсор ВВЕРХ на одну строку.

Для выполнения поиска в файле перейдите в командный режим, нажав клавишу Escape, а затем нажмите клавишу «/» и введите искомый текст, после чего нажмите клавишу Enter. Если такой текст есть в файле, то курсор будет установлен на первый символ этого фрагмента текста.

Команды для записи изменений в файл (после ввода команд нужно нажимать клавишу Enter):

:w записать файл с тем же именем, какое он имел при запуске редактора;

:w new_file_name записать файл с именем new_file_name.

Команды для выхода из редактора (после ввода команд нужно нажимать клавишу Enter):

:q прекратить редактирование файла и выйти из редактора. Если файл был изменен, а изменения в файле не были сохранены, то редактор не позволит выйти;

:q! прекратить редактирование файла и выйти из редактора. Если файл был изменен, а изменения в файле не были сохранены, то они будут потеряны. Говоря коротко, эта команда позволяет выйти из редактора без сохранения изменений.

ПРИМЕЧАНИЕ. Обратите внимание на символ ":".

Ну, вот теперь вы сможете блеснуть мастерством и на глазах у какого-нибудь изумленного приверженца графических оболочек запросто отредактировать файл в редакторе **vi**.

Контрольные вопросы и задания

1. Какие режимы работы имеет редактор **vi**? Как перейти в командный режим?

2. Какая команда служит для ввода текста в режиме добавления?

3. Как удалить строку текста?

4. Как сохранить отредактированный файл? Как выйти из редактора без сохранения внесенных изменений?

5. Создайте текстовый файл в редакторе **vi** и потренируйте основные навыки работы в нем: ввод текста, удаление текста, добавление пустых строк и удаление строк. Этих элементарных знаний хватит для редактирования, например, файла паролей в команде **vipw**, о которой речь пойдет позднее.

7. Основы программирования на языке Perl

Изучая операционную систему UNIX, в частности, FreeBSD, нельзя обойти вниманием и язык программирования Perl. В данной главе вы сможете познакомиться с основами программирования на этом языке, т. е.:

– научиться создавать программы на языке Perl и запускать их;
– изучить типы данных этого языка;
– изучить управляющие конструкции, применяемые в этом языке;
– изучить основные приемы работы с файлами;
– познакомиться с регулярными выражениями языка Perl.

7.1. Вводные сведения

Perl – зто язык программирования, который первоначально предназначался для обработки текстовых файлов: поиска и замены строк символов, сортировки, подсчета количеств каких-либо строк и т. п. Этот язык относится к классу так называемых языков сценариев (по-английски – **scripting languages**), поэтому часто программы, написанные на языке Perl, называют **скриптами**.

В настоящее время Perl является одним из языков, применяемых для программирования Internet-приложений. В предыдущих версиях FreeBSD он входил в состав операционной системы «по умолчанию». В в версиии FreeBSD8.2. Perl не входит в состав ОС «по умолчанию», его нужно дополнительно устанавливать. Ряд системных программ в этой FreeBSD написан на языке Perl. Этот язык отличается тем, что позволяет создавать полезные программы, используя даже небольшое подмножество функций языка. Существует реализация Perl для Windows. По своему синтаксису Perl очень похож на язык C, поэтому тем, кто знаком с языком C, будет сравнительно легко изучить Perl.

Для установки **perl** (из пакетов) выполните следующие действия: запустите команду **sysinstall**, затем войдите в меню **configure,** затем в меню **packages** и выберите **CD/DVD**.

Если вы не знаете где на диске находится дистрибутив, то тогда выберите пункт меню **all,** иначе - выберите **Lang.** Затем - **Perl 5.10.1_3**, затем **Install** и **OK.** Для проверки выберите команду **Perl –v.** Если все прошло успешно, то должно отобразиться сообщение версия **v5.10.1**, тогда все готово.

7.2. Первая программа на языке Perl в операционной системе UNIX

Запустите редактор **joe** и создайте в нем следующий текст:

```
#!/usr/bin/perl -w
print "Первая программа на языке Perl\n";
exit( 0 );
```

Сохраните его под именем, например, **first.pl**. Расширение «.**pl**» необязательно, но, как правило, программы на языке Perl имеют именно такое расширение. Чтобы вашу программу можно было исполнить, вы должны назначить соответствующие права доступа к файлу программы. Тем, кто забыл, как это сделать, напоминаем:

```
chmod 755 first.pl
```

Теперь можно запустить эту программу. Не забывайте о том, что в ОС UNIX по умолчанию в текущем каталоге не производится поиск программ. Если забыли, что нужно делать, чтобы все-таки запустить программу, посмотрите главу 2.

Теперь объясним назначение каждой строки этой маленькой программы. Первая строка указывает полный путь к интерпретатору языка Perl. Хотя он является интерпретатором, но работают программы на языке Perl очень быстро, т. к. сначала производится преобразование исходного текста во внутренний формат, а затем уже выполнение. Поэтому многократно выполняемые фрагменты программного кода преобразуются во внутренний код только один раз. Символы **-w** в первой строке служат указанием для проверки объявлений всех переменных в программе. В нашей программе пока нет переменных. Обратите внимание на символы **#!** в начале первой строки. Их наличие обязательно. Полный путь к интерпретатору Perl может быть не только **/usr/bin/perl**, но и **/usr/local/bin/perl**. Как правило, сразу после инсталляции операционной системы FreeBSD интерпретатор Perl помещается в каталог **/usr/bin**, а в том случае, когда администратор системы устанавливает более новую версию Perl, он помещает ее в каталог **/usr/local/bin**. Но зачастую просто делают символическую ссылку в каталоге **/usr/local/bin** на **/usr/bin/perl**, позволяя таким образом использовать в программах оба пути: **/usr/bin/perl** и **/usr/local/bin/perl**, которые указывают на один и тот же интерпретатор Perl.

В следующей строке помещен вызов функции **print**. В языке Perl имеется и функция **printf**, которая по своим возможностям аналогична такой же функции из языка C, но в простых случаях используется функция print. Обратите внимание на отсутствие круглых скобок – это не ошибка. Perl допускает их отсутствие в том случае, когда оно не мешает интерпретатору произвести синтаксический разбор выражения. Символы **\n** означают переход на новую строку.

В последней строке помещена функция **exit** для завершения программы. Она возвращает значение 0 операционной системе.

Как и в языке C, в конце оператора ставится точка с запятой.

7.3. Типы данных языка Perl

В языке Perl имеется три типа данных: **скаляры**, **массивы** и **ассоциативные массивы** (по-другому они называются **хеш-массивами**, или просто хешами). Покажем способы объявления переменных для этих типов данных, но сначала приведем некоторые объяснения. Если переменная объявлена с помощью ключевого слова **my**, то это означает, что переменная будет локальной, т. е. ее область видимости ограничена той функцией, в которой она объявлена, а время жизни – временем выполнения этой функции. Такие переменные можно объявлять и вне тела какой-либо функции (процедуры). В этом случае переменная доступна во всех процедурах, определения (тела) которых располагаются в тексте программы после ее объявления. Как правило, для объявления переменных используется **my**. Приведем несколько примеров объявлений переменных (обратите внимание на символы **$**, **@**, **%** перед именами переменных, массивов и хеш-массивов):

Скаляры

```
my $student;              # неинициализированная переменная
my $student = "Иванов"; # инициализированная переменная -
                          # строка
my $age = 19;             # инициализированная переменная -
                          # целое значение
my $weight = 75.8;        # инициализированная переменная -
                          # значение с десятичной частью
```

Символом # обозначаются комментарии. Все символы на строке, расположенные после этого символа, игнорируются при выполнении программы.

Скалярные переменные не имеют конкретного типа, подобного типу данных в языке C. Важным отличием от языка C является то, что все заботы о выделении памяти для строк символов, массивов и хеш-массивов берет на себя Perl. В объявлении

```
my $student = "Иванов";
```

значением переменной $student будет не указатель на строку, а сама строка. Поэтому для соединения (конкатенации) строк используется простая операция «.»:

```
my $last_name = "Иванов ";
my $first_name = "Иван";
my $full_name;
$full_name = $last_name . $first_name;
```

При выполнении операций над скалярами Perl определяет по **содержимому** переменной, как с ней поступить.

Массивы

```
# массив фамилий студентов
# (обратите внимание на символ @ перед именем массива)
my @students = ( 'Иванов', 'Петров', 'Сидоров' );
# пустой массив
my @empty_array = ();
```

Для обращения к элементу массива используется тот же подход, что и в языке C: $students[2] даст нам значение 'Сидоров'. Обратите внимание, что при обращении к элементу массива символ @ изменяется на $. Счет элементов в массиве начинается с нуля. Символьные строки ограничиваются либо одинарными, либо двойными кавычками. В ряде случаев это имеет *принципиальное значение*, что будет показано в дальнейшем. Для записи значения в массив можно просто присвоить значение элементу:

```
$students[ 5 ] = 'Новиков';
```

Если в массиве нет элемента с номером 5 (а наш массив @students более короткий – всего три элемента), то в этом случае необходимое число элементов будет добавлено автоматически. При этом элементы с индексами 3 и 4 получат неопределенные значения, которые в языке Perl обозначаются как **undef**.

Хеш-массивы

```
# пустой хеш-массив
# (обратите внимание на символ % перед именем хеш-массива)
my %empty_hash = ();
# хеш-массив, в котором записаны сведения о росте студентов
my %height = ( 'Иванов' => 176,
               'Петров' => 165,
               'Сидоров' => 190
             );
```

В хеш массивах данные хранятся парами: так называемый **ключ** и значение. Символы => используются для удобства, но можно вместо них ставить запятую. В нашем примере символьные строки 'Иванов', 'Петров', 'Сидоров' являются ключами, а соответствующие им числа – значениями. Чтобы извлечь требуемое значение из хеш-массива, нужно знать его ключ, но не нужно знать место хранения (как индекс в обычном массиве). Поэтому, чтобы получить рост студента Иванова, мы пишем:

```
$height{ 'Иванов' }
```

Обратите внимание, что при обращении к элементу символ % изменяется на $, а ключ помещается в фигурные скобки. Для записи значения в хеш-массив можно просто присвоить значение элементу с указанным ключом:

```
$height{ 'Новиков' } = 187;
```

7.4. Вторая программа на языке Perl в операционной системе UNIX

Запустите редактор **joe** и создайте в нем следующий текст:

```
#!/usr/bin/perl -w
print "Вторая программа на языке Perl\n";

# скалярные переменные
my $last_name = "Иванов ";
my $first_name = "Иван";
my $full_name;

# используем операцию конкатенации строк
$full_name = $last_name . $first_name;

# используем операцию конкатенации строк
print "Полное имя: ". $full_name . "\n";

# теперь вставим имя переменной внутрь строки в двойных
# кавычках
print "Повторим еще раз: $full_name\n";

# то же, но кавычки одинарные

# ПРИМЕЧАНИЕ. Когда используются двойные кавычки, тогда все
# переменные внутри такой строки заменяются их значениями.
# В случае одинарных кавычек такой подстановки значений
# не происходит.
print 'Найдите отличия от предыдущей команды: $full_name\n';

# пропустим пустую строку
print "\n";

# Массивы

# массив фамилий студентов
my @students = ( 'Иванов', 'Петров', 'Сидоров' );
# пустой массив
my @empty_array = ();
```

```
# используем операцию конкатенации строк
print $students[ 0 ] . "\n";

# теперь вставим имя массива внутрь строки в двойных кавычках
print "$students[ 1 ]\n";

# запишем значение в массив
$students[ 5 ] = 'Новиков';

# теперь выведем значения (обратите внимание на
# предупреждения)
print "$students[ 3 ]\n";
print "$students[ 4 ]\n";
print "$students[ 5 ]\n";

# Хеш-массивы

my %empty_hash = ();  # пустой хеш-массив

# хеш-массив, в котором записаны сведения о росте студентов
my %height = ( 'Иванов' => 176,
               'Петров' => 165,
               'Сидоров' => 190
             );

# получим рост студента Иванова
print $height{ 'Иванов' } . "\n";

# можно и так
print "Рост студента Петрова: $height{ 'Петров' }\n";

# запишем значение в хеш-массив
$height{ 'Новиков' } = 187;

# теперь выведем его
print "$height{ 'Новиков' }\n";

exit( 0 );
```

Перед запуском программы не забудьте предоставить владельцу файла с помощью команды **chmod** право выполнения программы.

7.5. Процедуры языка Perl

Для создания процедур (функций) используется ключевое слово **sub**. Запустите редактор **joe** и создайте в нем следующий текст:

```
#!/usr/bin/perl -w

print "Процедуры в языке Perl\n";
```

```
proc_1();

# печатаем строку, возвращаемую процедурой proc_2()
print proc_2() . "\n";

exit( 0 );

sub proc_1
{
  print "Процедура proc_1\n";
}

sub proc_2
{
  print "Процедура proc_2\n";
  # возвращаем строку (а не указатель, в отличие от языка C)
  return "FINISH";
}
```

Перед запуском программы не забудьте назначить право выполнения программы владельцу файла.

7.6. Организация выбора и циклов в языке Perl

Конструкции для организация выбора и циклов в языке Perl похожи на аналогичные конструкции в языке C. Рассмотрим их на примере очередной программы.

```
#!/usr/bin/perl -w

#use strict;

print "Конструкции выбора и циклы на языке Perl\n";

# объявим скалярные переменные
my $i;      # индекс в массиве
my $stud;   # фамилия студента

# массив фамилий студентов
my @students = ( 'Иванов', 'Петров', 'Сидоров' );

# запишем значение в массив
# ПРИМЕЧАНИЕ. Perl сам увеличит длину массива на три элемента.
$students[ 5 ] = 'Новиков';

print "Вывод массива\n";
```

```
# теперь выведем значения (обратите внимание на
# предупреждения)
# ПРИМЕЧАНИЕ. Этот цикл организован, как на языке C.
#             Переменная $#students хранит индекс последнего
#             элемента массива.
for ( $i = 0; $i <= $#students; $i++ )
{
  print "Элемент номер $i равен: $students[ $i ]\n";
}

print "\nВывод массива с проверкой значений\n";
# теперь будем выполнять проверку значений перед выводом
# их на печать
for ( $i = 0; $i <= $#students; $i++ )
{
  # обратите внимание на операцию сравнения eq - она означает
  # проверку на равенство строковых значений переменных,
  # для проверки на неравенство служит операция ne,
  # (например, $students[ $i ] ne 'Петров')
  if ( $students[ $i ] eq 'Петров' )
  {
    print "Элемент 'Петров' имеет номер $i\n";
  }
  # обратите внимание: не else if, а elsif; undef означает
  # отсутствие определенного значения, обычно такое значение
  # имеют неинициализированные переменные
  elsif ( $students[ $i ] eq undef )
  {
    print "Элемент номер $i не имеет определенного " .
          "значения\n";
  }
}

print "\nВывод массива без использования индекса в массиве\n";

# теперь покажем организацию цикла без использования индекса
# в массиве
foreach $stud ( @students )
{
  print "Студент $stud\n";
}

print "\nДальнейшее упрощение цикла с использованием "
     ."переменной \$_\n";
# ПРИМЕЧАНИЕ. Обратите внимание на символ "\" перед
#             символом "$". Это называется экранированием.
#             Оно необходимо, т.к. символ "$" имеет в языке
#             Perl специальное значение.

# Теперь покажем организацию цикла также и без использования
# переменной, которой присваиваются поочередно значения
# элементов массива
foreach ( @students )
```

```
{
  # переменная $_ является одним из самых часто используемых
  # элементов языка Perl. Значение ей присваивается неявно,
  # без прямого участия программиста. Эта переменная всегда
  # присутствует во всех операциях, когда происходит перебор
  # элементов массива, чтение строк из файла и т.д. Она
  # позволяет избежать обилия переменных.
  if ( $_ eq undef )   # переменная $_ равна текущему элементу
  {
    print "Неопределенное значение элемента\n";
  }
  else
  {
    print "$_\n";
  }
}

print "\nВыход из цикла с использованием оператора last\n";

# теперь покажем выход из цикла
foreach ( @students )
{
  # переменная $_ равна текущему элементу
  if ( $_ eq 'Петров' )
  {
    print "Мы нашли Петрова - выходим из цикла\n";
    last;                  # выход из цикла
  }
  else
  {
    print "$_\n";
  }
}

# Хеш-массивы

# хеш-массив, в котором записаны сведения о росте студентов
my %height = ( 'Иванов' => 176,
               'Петров' => 165,
               'Сидоров' => 190
             );

print "\nВывод хеш-массива\n";

# получить значения всех элементов хеш-массива можно так:
# ПРИМЕЧАНИЕ. В этом случае функция keys выбирает из
#             хеш-массива значения всех ключей (т.е. фамилии
#             студентов) и формирует из них массив,
#             а конструкция foreach перебирает все элементы
#             этого массива, присваивая поочередно их значения
#             переменной $stud_name.
foreach $stud_name ( keys %height )
{
```

```
  print "Рост студента $stud_name: $height{ $stud_name }\n";
}

print "\nВывод хеш-массива с использованием переменной \$_\n";

# получить значения всех элементов хеш-массива можно так:
# ПРИМЕЧАНИЕ. В этом случае функция keys выбирает
#             из хеш-массива значения всех ключей
#             (т.е. фамилии студентов) и формирует из них
#             массив, а конструкция foreach перебирает все
#             элементы этого массива, присваивая поочередно их
#             значения "теневой" переменной $_.
foreach ( keys %height )
{
  # для сравнения на точное равенство числовых значений
  # используется оператор "==", на неравенство: >, <, >=, <=
  if ( $height{ $_ } == 165 )
  {
    print "Студент $_ имеет низкий рост: $height{ $_ }\n";
  }
  else
  {
    print "Студент $_ имеет высокий рост: $height{ $_ }\n";
  }
}

exit( 0 );
```

Эта программа выводит информации столько, что она не умещается на одном экране. Для просмотра всей выведенной информации есть три способа. Первый заключается в использовании известной вам команды **more** таким образом:

```
./program.pl | more
```

Второй способ заключается в том, чтобы позволить программе вывести все, что она хочет, а затем нажать клавишу Scroll Lock. После ее нажатия можно использовать клавиши управления курсором для просмотра экрана вверх и вниз. В таком экранном буфере хранится примерно 130–150 строк текста (т. е. пять–шесть объемов экрана). Для прекращения просмотра буфера опять нажмите клавишу Scroll Lock.

Третий способ – это использование переадресации стандартного вывода в файл, который затем можно просмотреть с помощью любого доступного средства:

```
./program.pl > some_file.txt
```

Попробуйте убрать модификатор -w, вы заметите, что предупреждения на экран выводиться не будут. Попробуйте теперь добавить в начало файла после самой первой строки **#!/usr/bin/perl** такую строку:

```
use strict;
```

Вообще-то эта строка уже приведена в тексте программы, но она закомментирована. Вы просто удалите символ #, чтобы раскомментировать ее, и вы увидите, что теперь Perl требует от вас предварительного объявления всех переменных. Объявите переменную $stud_name в любом месте программы до ее первого использования.

7.7. Работа с файлами

Работать с файлами можно, направляя их на стандартный ввод вашей программы с помощью переадресации ввода. Создайте программу:

```
#!/usr/bin/perl -w

print "Работа с файлами\n";

my $i = 0;

# ПРИМЕЧАНИЕ. Подумайте, как запустить эту программу, чтобы
#             подать ей какой-нибудь файл на стандартный ввод.
#             Считываем файл построчно со стандартного ввода и
#             построчно выводим его, нумеруя строки
while ( <STDIN> )
{
  # вспомните, что означают символы "." В данном случае
  # переменная $_ "впитывает" в себя целую строку файла
  print ++$i . " " . $_;
}

exit( 0 );
```

Теперь покажем, как открыть файл внутри программы.

```
#!/usr/bin/perl -w

print "Работа с файлами\n";

my $i = 0;

# переменная ARGV является массивом, хранящим параметры,
# переданные программе в командной строке.
# ВАЖНОЕ ОТЛИЧИЕ от языка C: имя самой программы не является
# элементом этого массива, таким образом, элемент $ARGV[ 0 ] -
# это первый параметр, $ARGV[ 1 ] - второй параметр и т.д.
```

```
if ( $#ARGV != 1 )     # требуем ДВА параметра
{
  # выводим сообщение об ошибке на стандартное устройство
  # ошибок.
  # ПРИМЕЧАНИЕ. Обратите внимание на отсутствие запятой
  # после слова STDERR.
  print STDERR "Укажите параметры: имя входного файла и " .
               "имя результирующего\n";
  exit( 1 );    # возвращаем операционной системе значение 1
}

# откроем файл в режиме чтения

# ПРИМЕЧАНИЕ. Функция open() при успешном открытии файла
#             возвращает ненулевое значение. Символы "$!"
#             означают текст сообщения об ошибке, выдаваемый
#             операционной системой. Попробуйте указать
#             неверное имя исходного файла, чтобы увидеть это
#             сообщение.
#             Функция die аналогична функции exit(), но
#             предварительно выводит сообщение, указанное
#             в кавычках.
#              Операция "||" - это логическое "ИЛИ". Логическое
#             "И" обозначается "&&".
#             READ - дескриптор файла: их записывают без
#             кавычек, как правило, заглавными буквами.
open( READ, "< $ARGV[ 0 ]" ) ||
    die "Не могу открыть файл $ARGV[ 0 ]: $!\n";

# откроем файл в режиме записи.
# ПРИМЕЧАНИЕ. Для открытия файла в режиме дополнения вместо
#             префикса ">" нужно поставить ">>", в режиме
#             чтения/записи "+<" или "+>>"
open( WRITE, "> $ARGV[ 1 ]" ) ||
    die "Не могу открыть файл $ARGV[ 1 ]: $!\n";

# считываем файл построчно и построчно выводим его,
# нумеруя строки
while ( <READ> )
{
  # вспомните, что означают символы "." В данном случае
  # переменная $_ "впитывает" в себя целую строку файла.
  # ПРИМЕЧАНИЕ. Обратите внимание на отсутствие запятой после
  #             слова WRITE.
  print WRITE ++$i . " " . $_;
}

# закроем файлы:
close( READ ) || die "Не могу закрыть файл $ARGV[ 0 ]: $!\n";
close( WRITE ) || die "Не могу закрыть файл $ARGV[ 1 ]: $!\n";

exit( 0 );
```

7.8. Регулярные выражения языка Perl

Регулярные выражения (regular expressions) – это средство, предназначенное для анализа и обработки символьных строк: поиска шаблонов, замены одних фрагментов текста на другие. Регулярные выражения являются очень мощным средством, позволяющим выполнить сложные операции, используя компактные операторы.

```
#!/usr/bin/perl -w

print "Работа с регулярными выражениями\n";
my $i = 0;

# ПРИМЕЧАНИЕ. Подумайте, как запустить эту программу, чтобы
#             подать ей какой-нибудь файл на стандартный ввод.

# Считываем файл построчно со стандартного ввода и построчно
# выводим его, нумеруя строки, отбрасывая при этом комментарии
# (как полные строки, так и комментарии, которые находятся на
# одной строке с операторами), отбрасывая также и пустые
# строки (или строки, состоящие из одних пробелов).
# ПРИМЕЧАНИЕ. Наша программа очень простая, поэтому она
#             выполняет работу с небольшой ошибкой, которая
#             может проявиться, если на вход этой программе
#             подать, например, ее собственный текст.
#             Попробуйте найти проявление ошибки.
while ( <STDIN> )
{
  # в данном случае переменная $_ "впитывает" в себя целую
  # строку файла
  # ПРИМЕЧАНИЕ. Данное регулярное выражение означает
  #             следующее: во-первых, оператор =~ служит для
  #             выполнения действий над строкой, которая
  #             хранится в переменной $_; символ "s" означает
  #             замену одного фрагмента строки на другой -
  #             заменяемый фрагмент помещается между первой
  #             парой символов "/", а заменяющий - между
  #             второй парой символов "/". В данном случае
  #             заменяющим фрагментом является пустая строка.
  #             Внутри заменяемого фрагмента символ "#"
  #             понимается буквально, символ "." означает
  #             любой символ, символ "*" означает, что
  #             количество этих самых любых символов может
  #             быть также любым - от 0 и больше. После
  #             применения такого регулярного выражения
  #             к переменной $_ она изменится. Если в ней
  #             хранилась строка-комментарий, то останется
  #             пустая строка. Если комментарий занимал лишь
  #             часть строки, то он будет отброшен. Если же в
  #             строке не было символа "#", то она не
```

```
  #             претерпит никаких изменений.
  $_ =~ s/#.*//;

  # теперь сравним полученную строку с требуемым шаблоном,
  # который допускает наличия в строке только пробельных
  # символов (пробел, табуляция, новая строка), на что
  # указывает комбинация "\s". Символ "+" указывает, что этих
  # символов должно быть не меньше одного.
  # Символы "^" и "$" означают начало и конец строки
  # соответственно. Они не хранятся в строке, а являются
  # условными.
  # ПРИМЕЧАНИЕ. Обратите внимание, что в предыдущем шаблоне
  #             был символ "s" перед шаблоном, а внутри
  #             шаблона было три символа "/". В данном шаблоне
  #             нет символа "s" перед первым символом "/", а
  #             самих символов "/" всего два. Это объясняется
  #             тем, что в первом случае мы должны были найти
  #             определенный фрагмент текста и заменить его
  #             другим (пусть даже пустым) фрагментом, а во
  #             втором случае нам нужно лишь найти
  #             определенный фрагмент,не модифицируя его.
  if ( $_ =~ /^\s+$/ )
  {
    # переход к началу цикла, минуя оставшуюся часть тела
    # цикла
    next;
  }

  # вспомните, что означают символы "."
  print ++$i . " " . $_;
}

exit( 0 );
```

7.9. Вызов справки по языку Perl

Для вызова справки по языку Perl используйте команду **man**:

```
man perl
```

Обратите внимание, что справочная информация по языку разделена на отдельные секции, список которых приведен в заглавной справочной странице (мы говорим «страница», т. к. полное название электронных руководств в ОС UNIX – manual pages, а на жаргоне они называются просто «маны»). Например, для получения справки по структурам данных, используемым в языке Perl, нужно ввести

```
man perldata
```

Для просмотра справки используйте клавиши управления курсором, а для выхода – клавишу Q.

Для получения информации о параметрах запуска Perl введите

```
perl -h
```

Контрольные вопросы и задания

1. К классу каких языков относится язык Perl?

2. Для чего нужна первая строка программы на язык Perl:

```
#!/usr/bin/perl -w
```

3. Какие типы данных есть в этом языке?

4. Что такое хеш-массивы? Что такое **ключ** в хеш-массиве? Как можно обратиться к элементу хеш-массива? Как можно записать новое значение в хеш-массив?

5. Какие способы организации циклов на языке Perl вы знаете?

6. Что означает переменная **$_** и как она используется?

7. Каким образом можно соединить две символьных строки на языке Perl? Может ли функция возвращать символьную строку? В чем отличие от языка С?

8. Что такое **регулярные выражения**? Приведите пример простейшего из них?

9. Как открыть файл в программе на языке Perl в режиме дополнения, в режиме чтения/записи, в режиме только чтения?

10. Каким образом можно организовать построчное чтение текстового файла в программе на языке Perl?

11. Что означает выражение $ARGV[0] в языке Perl? В чем отличие от языка С?

12. Внимательно изучите все программы, приведенные в этой главе. Все комментарии в этих программах носят учебный характер и потому их нужно тщательно изучить. Затем попытайтесь модифицировать программы и, выполняя их, смотрите, что у вас получается. Модифицирование может заключаться в добавлении новых переменных, изменении числа элементов

в массивах или хеш-массивах, оформлении отдельных фрагментов программы в виде процедур, открытии в программах файлов в различных режимах (вместо режима дополнения – в режиме чтения) и т. д. Фантазируйте, ведь программирование – процесс творческий. Иначе ничему не научитесь! Руководствуйтесь известным афоризмом К. Пруткова: «Бросая в воду камешки, смотри на круги, ими образуемые, чтобы такое занятие не было пустою забавою». За точность воспроизведения цитаты не ручаемся, но смысл именно такой.

8. Основы программирования на языках сценариев (scripting languages) в ОС UNIX

Имея представление о языке программирования Perl, вам будет легче познакомиться с основными языками сценариев, использующимися в ОС UNIX. Нашей с вами целью в данной главе будет:

– познакомиться с языком awk;
– познакомиться с потоковым редактором sed;
– освоить основы программирования на языке shell;
– научиться создавать командные файлы на языке shell и запускать их;
– изучить управляющие конструкции, применяемые в этом языке.

8.1 Язык awk

Язык программирования awk предназначен для обработки текстовых файлов. Он ориентирован на их **построчную** обработку. В строках обрабатываемого файла можно отыскивать какие-либо подстроки и, в зависимости от успеха или неудачи этого поиска, выводить строки файла на стандартный вывод или не выводить. Язык awk позволяет проводить поиск и замены строк символов, подсчета количеств каких-либо строк и т. п. Этот язык так же, как и Perl, относится к классу так называемых **языков сценариев** (по-английски – scripting languages). Он часто используется в системном программировании в среде ОС UNIX. Этот язык по своему синтаксису очень похож на язык C, поэтому тем, кто знаком с языком C, будет сравнительно легко изучить awk. Многие операции в awk точно такие же, как и в C, например, логические операции, операции сравнения.

Язык awk часто используется в системных скриптах, написанных на языке shell, с которым мы познакомимся в следующих разделах этой главы.

8.1.1. Первая программа на языке awk в операционной системе UNIX

Тексты программ на языке awk зачастую весьма короткие, поэтому для начала мы рассмотрим самый простой способ оформления текста программы и запуска awk. Введите в командной строке такую команду:

```
awk '{ print }' file_name
```

Здесь **file_name** – это любой текстовый файл, например, текст программы на языке Perl. Вы увидите, что будет выведен на экран весь текст вашего файла. Обратите внимание на то, что кавычки – одинарные. В данном случае можно использовать и двойные кавычки, но все же привыкайте

к использованию одинарных кавычек. Конечно, практической пользы от этой простой команды не особенно много, но даже такая команда показывает все компоненты, необходимые для запуска awk:

awk сам интерпретатор языка;
'{ print }' текст программы (пока что она совсем короткая);
file_name имя обрабатываемого файла.

Поясним работу этой программы. Язык awk имеет в своем арсенале богатый набор различных функций. Функция **print** – одна из них. В данном случае она просто печатает одну строку файла, а поскольку awk обрабатывает **все** строки файла **file_name**, то данная функция применяется ко всем строкам этого файла, и в результате он полностью выводится на экран.

8.1.2. Вторая программа на языке awk в операционной системе UNIX

Усложним эту программу, а заодно повторим использование операции конвейеризации, т. е. объединения нескольких команд в одну цепочку. Введите в командной строке такую команду (здесь параметром команды **ls** является латинская строчная буква «el», а вертикальная черта – это и есть символ «конвейера», который на клавиатуре совмещен с символом «\»):

```
ls -l | awk '{ print $1, $9 }'
```

Нажав клавишу Enter, вы получите список файлов, состоящий всего из двух полей: поля прав доступа и имени файла (или каталога). В этой программе используются переменные $1 и $9. Они означают номера полей в обрабатываемых строках. По умолчанию awk «считает», что поля отделяются одно от другого пробелом. Если вы запустите команду **ls -l**, то увидите, что она выводит девять полей, разделенных пробелами. Попробуйте убрать запятую между переменными $1 и $9. Что видите? А теперь сделайте так:

```
ls -l | awk '{ print "Права доступа: " $1 " Имя файла: " $9 }'
```

Внимательно изучите то, что выводит данная команда.

8.1.3. Регулярные выражения языка awk

Допустим, что нам нужно вывести только список подкаталогов текущего каталога. Мы знаем, что в выводе команды **ls** каталоги обозначаются символом d в первом поле, в котором указаны права доступа. Введите команду и выполните ее:

```
ls -l | awk '{ if ($1 ~ /d/) print $1, $9 }'
```

Как видите, awk «знает» об условных операторах **if**. Но это еще не все, на что он способен. Выражение **$1 ~ /d/** является примером **регулярного выражения**, аналогичного тем регулярным выражениям, которые применяются в языке Perl. Данное регулярное выражение означает следующее: есть ли в первом поле текущей записи символ d? Наличие двух символов «/» обязательно.

Перечислим основные правила формирования регулярных выражений, а затем приведем несколько примеров их использования.

. символ «точка» соответствует любому символу;

^ означает начало строки (но сам символ ^в строку не включается);

$ означает конец строки (но сам символ $ в строку не включается);

[abc...] любой из перечисленных символов;

[^abc...] любой из символов, *кроме* перечисленных в этом списке;

str1|str2 либо строка str1, либо строка str2;

r+ означает один или более символов r (где r – это условное обозначение регулярного выражения или просто любого символа, например, выражение «-+» означает один или более символов «-»);

r* означает ни одного, один или более символов r (где r – это условное обозначение регулярного выражения или просто любого символа, например, выражение «-*» означает ни одного, один или более символов «-»);

r? означает ни одного или ровно один символ r (где r – это условное обозначение регулярного выражения или просто любого символа, например, выражение «-?» означает ни одного либо ровно один символ «-»);

(str) обозначает группу символов. Это бывает удобно, когда необходимо найти строки входного потока записей, в которых целая группа символов может повторяться более одного раза;

А теперь рассмотрим примеры использования регулярных выражений.

1. Выбрать все файлы, право на исполнение которых имеют члены группы:

```
ls -l | awk '{ if ($1 ~ /^.....x/ && $1 !~ /^d/ ) print $1,
$9 }'
```

В этой команде два регулярных выражения, соединенных оператором «логическое И», т. е. символами **&&**. Действие логических операторов аналогично языкам C и Perl. Первое регулярное выражение означает следующее: если в седьмой позиции от *начала* первого поля находится символ x, то это означает наличие права на исполнение у членов группы. При этом

нас не интересует, какие символы находятся на шести первых позициях, поэтому мы используем метасимвол «.», который используется для обозначения *любого* символа. А второе регулярное выражение означает, что первым символом первого поля не должен быть символ d, т. е. это не должен быть каталог. При выполнении двух этих условий awk выводит информацию о файле. Вы можете вместо двух переменных $1 и $9 указать еще и, например, $2. Попробуйте также использовать переменную $0. Вы увидите, что она соответствует целой строке. Когда вы пишете **print** без указания переменных, то подразумевается именно $0, т. е. вся строка.

2. Выбрать все файлы, право на исполнение которых имеют все пользователи, не являющиеся владельцем файла, а также не входящие в данную группу:

```
ls -l | awk '{if ($1 ~ /x$/ && $1 !~ /^d/ ) print $1, $9}'
```

Выражение $1 ~ /x$/ означает *последний* символ в первом поле. При этом символ $ лишь «привязывает» наше регулярное выражение к концу строки, но сам не является частью строки.

3. Выбрать все файлы и каталоги, имена которых начинаются с точки:

```
ls -al | awk '{if ( $9 ~ /^\./ ) print }'
```

Выражение $9 ~ /^\./ означает, что первый символ в девятом поле должен быть точкой. Поскольку символ «.» имеет специальное значение, то приходится его, как говорят, экранировать символом «\», чтобы он воспринимался буквально, т. е. именно как точка. При этом символ ^ лишь «привязывает» наше регулярное выражение к началу строки, но сам не является частью строки.

4. Выбрать все файлы, право на исполнение которых имеют все пользователи (в том числе владелец файла, а также члены группы):

```
ls -l | awk --posix '{ if ($1 ~ /..x..x..x/ && $1 !~ /^d/ )
print $0 }'
```

В данном случае мы проверяем первое поле в каждой строке, выводимой командой **ls -l**, на наличие символа x в конце каждой тройки символов, соответствующей владельцу файла, группе и всем остальным пользователям.

5. Убрать все строки из текста программы на языке Perl, которые содержат только комментарии: Чтобы у вас не сложилось представление о том, что awk можно использовать только в сочетании с командой **ls**, выполните такую команду (в ней **my_prg.pl** означает какой-нибудь файл с текстом программы на языке Perl):

```
awk '{ if ( $0 !~ /^[ ]*#/ ) print $0 }' my_prg.pl
```

Переменная $0 обозначает всю входную/выходную строку в целом. Данная команда отбрасывает все строки, являющиеся комментариями. Такие строки начинаются с символа #, которому могут предшествовать пробелы. Число пробелов может быть от нуля и больше, что описывается выражением **[ ]*** (в квадратных скобках находится ровно один пробел), причем они должны находиться в начале строки, на что указывает наличие символа ^ в регулярном выражении. Вы увидите, что данная команда выводит текст программы на языке Perl, отбросив все строки-комментарии (однако те строки, в которых комментарии находятся на одной строке с операторами, будут оставлены).

8.1.4. Встроенные переменные языка awk

В языке **awk** есть встроенные переменные. Перечислим основные из них:

FS разделитель полей во входной записи (по умолчанию – пробел);

NF общее число полей в текущей записи (строке);

NR общее число обработанных записей (строк);

OFS разделитель полей в выходной (результирующей) записи (по умолчанию – пробел);

ORS разделитель записей в выходном потоке (по умолчанию – новая строка «\n»);

RS разделитель записей во входном потоке (по умолчанию – новая строка «\n»).

Прежде чем показать примеры, мы должны рассказать еще о двух очень важных компонентах программы на языке awk. Это блоки BEGIN и END. Эти блоки выполняются не для каждой строки входного потока, а только один раз: блок BEGIN выполняется перед началом обработки входного потока, а блок END – после обработки последней строки в потоке.

Следующий пример показывает, каким образом можно изменить разделитель полей с пробела (который был разделителем по умолчанию) на двоеточие и заодно подсчитать число обработанных строк, что в данном случае означает число файлов в каталоге (эта команда не умещается на одной строке и при вводе она будет автоматически продолжена на новой строке):

```
ls -l | awk 'BEGIN { OFS=":" }{ print $1, $2, $3, $9 } END {
print "Общее число файлов: " NR }'
```

8.1.5. Переменные языка awk

В языке awk есть не только встроенные переменные, но и переменные, определяемые пользователем. Покажем их использование на примере вычисления общего размера файлов в каталоге:

```
ls -l | awk 'BEGIN { total = 0 } { print; total += $5 } END { print "Общий объем файлов: " total }'
```

Как вы можете легко проверить, пятое поле в выводе команды **ls -l** содержит размер файла. В блоке BEGIN объявляется и инициализируется переменная total. Обратите внимание, что она используется без символа $. В главном блоке происходит суммирование объемов файлов, а в блоке END выполняется вывод итогового значения.

8.1.6. Функции и другие возможности языка awk

Мы рассмотрели только одну функцию – **print**. Однако есть много других полезных функций: функции для обработки строк, функции для обработки чисел. Кроме этого, в awk можно создавать и функции пользователя. Язык awk умеет организовывать циклы **while** и **for**. Посмотрите описание awk, используя команду **man awk**.

8.2. Потоковый редактор sed

Возможности потокового редактора sed покажем на нескольких примерах. Этот редактор ориентирован на построчную обработку текстовых файлов. Он позволяет выполнять замены одних подстрок на другие. Для этого можно использовать как буквальные подстановки, так и регулярные выражения, аналогичные тем, которые только что были рассмотрены при ознакомлении с awk. Сначала рассмотрим простейший случай – замену одной подстроки на другую:

```
sed 's/файл/ФАЙЛ/g' original_file > modified_file
```

Эта команда считывает файл с именем **original_file**, заменяет в нем все подстроки «файл» на подстроки «ФАЙЛ» и выводит обработанный текст на стандартный вывод, который мы перенаправляем в новый файл **modified_file**. Символ s означает замену подстроки, а символ g – глобальную замену, т. е. замену всех таких подстрок в текущей строке входного файла. Если убрать символ g, то при наличии в строке исходного файла более одной указанной подстроки будет заменена только первая из них. Попробуйте убрать символ g и выполнить эту команду, подобрав такой

файл, в котором есть строки, содержащие две или более заменяемых подстроки.

А теперь опять воспользуемся программой на языке Perl в качестве «подопытного кролика». И опять уберем из нее комментарии, расположенные на отдельной строке (символ ^ указывает на то, что поиск подстроки начинается с самого начала строки; символы «.*» имеют то же значение, что и в языке awk):

```
sed 's/^#.*/здесь был комментарий/g' original.pl > modified.pl
```

Здесь **original.pl** – это имя какой-нибудь программы на языке Perl, находящейся в текущем каталоге, а **modified.pl** – это имя нового (результирующего) файла.

Можно заменить строки-комментарии на пустые строки:

```
sed 's/^#.*//g' original.pl > modified.pl
```

Редактор sed часто используется в системных скриптах, написанных на языке shell, с которым мы познакомимся в следующем разделе.

8.3. Язык программирования shell

Язык программирования shell предназначен для создания системных и пользовательских программ в среде ОС UNIX. Он похож на командный язык MS-DOS, но гораздо мощнее. Точнее говоря, командный язык MS-DOS представляет собой упрощенный вариант языка shell. Этот язык так же, как и Perl, относится к классу так называемых языков сценариев (по-английски – scripting languages). Он часто используется в системном программировании в среде ОС UNIX. Этот язык по своему синтаксису очень похож на язык C, поэтому тем, кто знаком с языком C, будет сравнительно легко изучить shell.

8.3.1. Командные файлы

В ОС UNIX широко используются командные файлы. Напишем простейший командный файл, состоящий всего из двух строк:

```
#!/bin/sh
echo "Это мой первый командный файл"
```

Для его исполнения можно поступить так:

```
sh ./my_command_file
```

Но можно предварительно назначить этому файлу права на исполнение с помощью команды **chmod**, а затем выполнить его так:

```
./my_command_file
```

ВАЖНОЕ ПРИМЕЧАНИЕ. Первая строка командного файла указывает операционной системе, какой интерпретатор команд использовать.

Посмотрев файлы в каталогах **/usr/sbin**, **/usr/bin**, вы можете найти системные командные файлы. Они зачастую очень сложны.

8.3.2. Более сложный пример командного файла на языке shell

Основы программирования на shell рассмотрим на примере. Напишем командный файл и дадим ему какое-нибудь имя:

```
#!/bin/sh
# ---------------------------------------------------------
# Программа: учебная программа для ознакомления с основами
#            программирования на shell.
# Автор:     Моргунов Е.П.
# Дата:      25.03.2002
# ---------------------------------------------------------

# проверим, сколько раз запущена оболочка deco?
echo "Сколько раз запущена оболочка deco?"

# опишем команду
# ps -ax - выводит список выполняющихся процессов
# grep deco - выбирает среди них процессы, выполняющие
#             программу deco
# grep -v grep - отбрасывает строки, в которых выводится
#             информация о процессе, который выполняет саму
#             команду grep (попробуйте выполнить команду
#             ps -ax | grep deco из командной строки и
#             посмотрите, что получается, а затем добавьте
#             grep -v grep через символ "конвейера")
# awk - получает каждую строку, описывающую один процесс, со
#             стандартного ввода и выводит информацию в
#             "читабельном" виде

ps -ax  |  grep deco  |  grep -v grep  |
           awk '{ print "deco запущена на терминале " $2, \
                        "процесс номер " $1 }'

echo "Организация цикла и использование awk для " \
     "анализа типов файлов"

# теперь покажем организацию цикла с использованием ключевого
```

```
# слова for в заголовке цикла:
# file_name - имя переменной
# `ls` - это команда ls, но при ее выполнении происходит
#        следующее:
#        все имена файлов, которые она выдаст, подставляются
#        в виде списка в условие цикла for, а затем эти имена
#        поочередно присваиваются переменной file_name
#        (обратите внимание на наличие ключевого слова in).
# ВАЖНЕЙШЕЕ ПРИМЕЧАНИЕ. Обратите внимание на обратные кавычки
#                       вокруг команды ls. Именно они
#                       заставляют shell выполнить команду ls,
#                       а затем подставить результат
#                       выполнения команды ls в условие
#                       цикла. Если бы этих кавычек не было,
#                       то имя команды ls было бы воспринято
#                       как имя обычного файла, и команда ls
#                       даже не была бы выполнена. Обратные
#                       кавычки - это очень мощное и часто
#                       используемое средство при
#                       программировании на shell.
# ПРИМЕЧАНИЕ. Чтобы ввести символ "обратная кавычка"
#             в редакторе joe, нужно нажать данную клавишу
#             дважды.
for file_name in `ls`
{
  # команда echo выводит значение переменной file_name
  # на стандартный вывод;
  # параметр -n указывает на то, что не нужно выводить символ
  # новой строки;
  # обратите внимание, что при использовании переменной нужно
  # перед ее именем указывать символ $;
  # символ "|" - это "конвейер", позволяющий передать имя
  # очередного файла или каталога команде awk;
  # в качестве символа-разделителя для полей назначаем точку;
  # символ "\" позволяет продолжить команду на новой строке;
  # выполняем примитивный анализ имен файлов: здесь мы
  # разделяем имя файла на собственно имя и расширение,
  # а затем анализируем их
  echo -n $file_name |
  awk 'BEGIN { FS = "." } \
      { if ( NF == 1 ) \
          print $1 " - это каталог или файл без расширения"; \
        else if ( $1 == "" ) \
          print $1 "." $2 " - это служебный файл"; \
        else if ( $2 == "pl" ) \
          print $1 "." $2 " - это программа на языке Perl"; \
        else if ( $2 == "html" || $2 == "htm" ) \
          print $1 "." $2 " - это текст на языке HTML"; \
        else \
          print $1 "." $2 " - это неизвестный файл"; \
      }'
}
```

```
echo "Организация цикла и использование shell " \
     "для анализа типов файлов"

# а теперь покажем, как можно определить тип файла
# средствами shell
for file_name in `ls`
{
  # это условный оператор в квадратных скобках записывается
  # условие, в данном случае условие -d означает - каталог,
  # условие -x означает исполняемый файл;
  if [ -d $file_name ]; then
    echo "$file_name - это каталог"
  elif [ -x $file_name ]; then
    echo "$file_name - это исполняемый файл"
  fi
}

# команда exit возвращает операционной системе значение 0
exit 0;
```

Для исполнения этого командного файла можно поступить так:

sh my_command_file

Но можно предварительно назначить этому файлу права на исполнение с помощью команды **chmod**, а затем выполнить его так:

./my_command_file

Ниже приведен пример еще одного командного файла, который **монтирует флэш-устройство**:

```
#!/bin/sh
#  Author: Novikov Dmitriy (BIS-81)
#  Purpouse: Mounts devices with names da[0-9]s[0-9]
#              Enables Russian locale for mounted devices.
#  Notice: Works only for FAT32 file systems

mount_successful=0
DEFAULT_LANG="ru_RU.KOI8-R"

# if the LANG variable is not set in the system
# concider it to be set as DEFAULT_LANG

LANG_=${LANG:-$DEFAULT_LANG}

echo "Looking for apropriate devices in /dev:"
ls -1 /dev | grep 'da[0-9]s[0-9]' >devs

# if there were no devices found
```

```
if test -z `cat devs` ; then  # test -z returns 0 if length of
string is 0
  echo "Found: none"
  echo "Mntzilla is unable to detect your flash device."
else
  echo "Found: "`cat devs`
  echo
  # For all the devices found
  for device in `ls -n /dev | grep 'da[0-9]s[0-9]'`
  do  # try to mount device to /mnt; cd to it and list its
contents
      mount_msdosfs -L $LANG_ -D CP866 /dev/$device /mnt
$1>/dev/null
      if test $? -eq 0 ; then
        cd /mnt
        echo "Contents of attached flash:"
        echo
        ls
        mount_successful=1
        break
      else
        echo "Device $device can not be mounted."
      fi
  done
  # If mount was successful
  if [ $mount_successful -eq 1 ]; then
    echo
    echo "Your flash is mounted successfully."
    echo "Don't forget to unmount it with 'umount /mnt'."
  else  # if mount failed
    echo
    echo "Your flash is either already mounted"
    echo "or can not be mounted by mntzilla."
  fi
fi

#echo "Looking for apropriate devices in /dev:"
#ls -n /dev | grep da0 >devs
#devs=`cat devs`

#echo "Found:" $devs

#for device in devs
# do
#    mount_msdosfs -L ru_RU.KOI8-R -D CP866 /dev/$device /mnt
# done

# mount_msdosfs -L ru_RU.KOI8-R -D CP866 /dev/$devs[1] /mnt

# awk 'BEGIN {FS="\n"} \
#         { if (NR<=1) \
#            print "Nothing" \
#           else \
```

```
#             for (i=1; i<NR ; i++) \
#              print $i }'

#     mount_msdosfs -L ru_RU.KOI8-R -D CP866 /dev/$devs /mnt
```

8.3.3. Вызов справки по языку shell

Для вызова справки по языку shell используйте команду **man**:

```
man sh
```

Контрольные вопросы и задания

1. Используя awk, извлеките из файла строки, содержащие какое-либо слово. Совет: используйте выражение $0, которое обозначает всю строку в целом, и регулярное выражение.

2. Используя регулярные выражения awk, отобразите содержимое любого каталога, но так, чтобы при этом отображались только подкаталоги, входящие в каталог, а файлы – нет.

3. Используя регулярное выражение awk, выберите и выведите на экран из какого-нибудь файла все строки, которые содержат нули.

4. Используя регулярное выражение awk, отобразите суммарный размер файлов текущего каталога.

5. Как изменить разделитель полей в выходной записи при работе с awk?

6. Используя редактор sed и регулярные выражения, найдите в файле слова, в которых вторым символом является латинская буква «а», а за ней идет не более двух символов. В результате выполнения этой команды должны быть найдены, например, такие слова, как: ban, was, bad, ras. Можно использовать выражения /< и />, которые обозначают начало и конец слова соответственно.

7. В файле замените какое-нибудь слово в каждой строке на другое слово.

8. В программе на языке shell объявлена переменная, например, today_date и выполнена такая команда:

```
today_date = `date`
```

Что обозначают «обратные» кавычки?

9. Как можно организовать цикл в программе на языке shell?

10. Как выполнить проверку типа того или иного файла в языке shell?

11. Как и при изучении языка Perl, наша рекомендация – модифицировать приведенные примеры программ на awk, sed и shell и постепенно усложнять их, обращаясь при необходимости к электронным руководствам **man**.

9. Основы администрирования пользователей в ОС FreeBSD

С чего начинается администрирование операционной системы? Возможно, с создания пользовательских учетных записей и управления ими. Теперь у вас уже достаточно знаний, чтобы познакомиться с основами администрирования пользователей в ОС UNIX (на примере FreeBSD), т. е.:

– настраивать программную среду пользователей;
– удалять учетные записи пользователей системы;
– назначать владельцев для файлов и каталогов;
– познакомиться со структурой файла паролей.

9.1. Настройка среды пользователя

Сначала нужно сказать два слова о командном интерпретаторе. Когда мы вводим какую-либо команду в командной строке, именно командный интерпретатор, по-другому он называется **shell**, производит ее анализ и исполнение. В ОС UNIX существует несколько командных интерпретаторов. Самый старый из них (и самый, наверное, простой) – это Bourne Shell, который вызывается так: **/bin/sh**. Он есть во всех UNIX-системах. Пользователь системы может изменить свой **shell** на другой, но мы пока этого делать не будем. Настройка среды (программного окружения) пользователя зависит от выбранного командного интерпретатора, а он выбирается при создании учетной записи пользователя. Эта настройка для Bourne Shell производится в два этапа: первый этап выполняется при начальной загрузке системы, на этом этапе обрабатывается файл **/etc/profile**. В этом файле указываются настройки, которые являются общими для всех пользователей, которые используют **/bin/sh**. На втором этапе, т. е. при регистрации пользователя в системе (через **login**), используется файл **.profile**, который находится в домашнем каталоге пользователя. Кроме него есть еще файл **.shrc**, который вызывается из файла **.profile**. После обработки этих файлов программная среда пользователя будет сформирована.

Программная среда включает в себя ряд **переменных**. Рассмотрим основные из них:

BLOCKSIZE – размер единиц объема, в которых выражается информация, выводимая некоторыми командами (главные из них: **df**, **du**, **ls**). Часто этот размер выражается в килобайтах, для этого значение этой переменной присваивается равным «K»;

EDITOR – текстовый редактор, используемый по умолчанию. При установке системы это, как правило, редактор **vi**. Его можно заменить на любой другой редактор, установленный в системе, например, **joe**;

HOME – домашний каталог пользователя. В этом каталоге пользователь является полновластным хозяином, т. е. имеет все права доступа к файлам;

LANG – эта переменная используется теми программами, которые учитывают национальные особенности той страны, которая выбрана при установке ОС. Эта переменная нужна для выполнения локализации («русификации») системы;

MAIL – местонахождение файла, в котором сохраняется электронная почта, направленная пользователю (см. программу **mail**);

PAGER – программа, используемая по умолчанию такими программами, как **mail**, **man**, **ftp** и др. для постраничного просмотра информации на экране;

PATH – команда **PATH (путь)**: это список каталогов, в которых производится поиск исполняемых программ;

SHELL – полное имя командного интерпретатора, используемого данным пользователем;

TERM – вид терминала (т. е. настройки дисплея), используемый данным пользователем. Переменная также может использоваться для выполнения локализации («русификации») системы;

TZ – часовой пояс;

USER – входное имя пользователя, работающего в системе.

Просмотреть состояние программной среды можно с помощью команды **env.**

Посмотрите содержимое файлов **/etc/profile** и **.profile** из домашнего каталога. Там можно увидеть команду **export**. Она служит для помещения переменной и ее значения в программную среду. Перед вызовом этой команды сначала нужно присвоить переменной какое-либо значение, например: **BLOCKSIZE=K**. После выполнения команды **export** переменная становится доступной для всех программ (процессов), запущенных пользователем. На одной строке можно указать более одной команды, разделяя их точкой с запятой. Например:

```
BLOCKSIZE=K; export BLOCKSIZE
```

Еще одна команда – **umask**. Она позволяет задать права доступа по умолчанию к вновь создаваемым файлам и каталогам. О ней подробнее будет сказано позднее. Заодно упомянем еще две команды. Команда **id** выводит информацию о пользователе, его группе и группах, в которые он включен. А команда **logname** выводит имя (идентификатор) пользователя.

9.2. Удаление учетной записи пользователя

Создавать новые учетные записи пользователей вы уже научились, прорабатывая главу 2. Противоположной операцией является удаление учетных записей. Для выполнения этой операции служит программа **rmuser**. Она написана на языке shell и находится в каталоге **/usr/sbin.** Перед тем как приступить к изучению программы **rmuser**, создайте учетную

запись пользователя, на которой вы и потренируете свои навыки удаления учетных записей.

Зарегистрируйтесь в системе под именем пользователя **root** и запустите эту программу, набрав на клавиатуре:

```
rmuser
```

Далее следуйте указаниям, которые программа выводит на экран. Рассмотрим эти указания в порядке их поступления.

```
Please enter one or more usernames:
```

Введите имя пользователя, которого нужно удалить из системы, пусть это будет пользователь с именем **test**. На экран будет выведена вся учетная запись этого пользователя, например:

```
Matching password entry:

     test:*:1002:1002::0:0:Test user:/home/test:/bin/sh

Is this the entry you wish to remove?
```

Это та запись, которую вы хотите удалить? Вы должны нажать клавишу «y» и затем Enter для подтверждения удаления учетной записи.

Теперь вас спросят о необходимости удаления домашнего каталога этого пользователя:

```
Remove user's home directory (/home/test)?
```

Отвечайте утвердительно – «y».

```
Removing user (test): mailspool home passwd.
```

Все. Учетная запись пользователя удалена. Можете проверить это, посмотрев файл **/etc/master.passwd**.

9.3. Включение пользователей в группы

В каталоге **/etc** есть файл **group**. Посмотрите его содержимое (например, используя редактор **joe** или модуль просмотра текстовых файлов, вызываемый из файлового менеджера **deco** нажатием клавиши F3). Вы увидите, что в нем есть группа, имя которой совпадает с именем того пользователя, учетную запись которого вы создали при изучении второй главы нашего пособия. Если вы дали пользователю имя **stud**, то в файле **group**

будет группа **stud**. Конечно, это будет так, если вы следовали инструкциям, приведенным выше.

Каждая запись этого файла состоит из четырех полей:

имя группы – оно должно быть уникальным (нужно использовать латинские буквы, желательно в нижнем регистре);

пароль группы – в этом поле стоит звездочка (пароль на практике используется редко);

числовой идентификатор группы – он должен быть уникальным;

члены группы – список имен пользователей, входящих в эту группу (имена в списке разделяются запятыми, пробелы недопустимы).

При необходимости включить какого-либо пользователя в группу можно отредактировать этот файл непосредственно в текстовом редакторе, вписав имя пользователя в поле членов этой группы.

Обратите внимание на группу **wheel** – это самая важная группа. Ее члены имеют ряд привилегий, о которых будет сказано позднее.

9.4. Смена паролей

Для смены пароля пользователя служит программа **passwd**. Она находится в каталоге **/usr/bin**. Вы уже знакомились с ней ранее. Теперь же вы увидите, что пользователь **root** может сменить пароль у любого другого пользователя. Запустите эту программу, набрав на клавиатуре:

```
passwd user_name
```

где **user_name** – имя_пользователя, пароль которого вы хотите сменить. Смените пароль у какого-либо пользователя, созданного вами. Если вы меняете пароль суперпользователя **root**, то будьте очень внимательны, т. к. если вы забудете этот пароль, то восстановить его невозможно. В этом случае для входа в систему под именем **root** нужно будет выполнить действия, требующие более высокой квалификации, чем та, которой вы обладаете на настоящий момент.

Далее следуйте указаниям, которые выводит на экран эта программа. Пароль нужно ввести дважды, при этом он должен иметь не менее 6 символов в длину. Программа **adduser** не требовала, чтобы длина пароля была не менее 6 символов, но программа **passwd** это требует. Тут есть некоторое противоречие, но на данном этапе изучения ОС UNIX с этим можно смириться.

После изменения пароля попытайтесь вновь войти в систему под именем этого пользователя.

9.5. Обзор файла паролей

Как вы уже знаете, в каталоге **/etc** есть два файла паролей: **/etc/passwd** и **/etc/master.passwd**.

ВНИМАНИЕ. Ни в коем случае не редактируйте эти два файла с помощью обычного текстового редактора.

Структуру файла паролей можно узнать, посмотрев электронное руководство:

```
man 5 passwd
```

ПРИМЕЧАНИЕ. Цифра 5 означает номер раздела электронного руководства. Статья с именем **passwd** есть в двух разделах. Поэтому нужно указать конкретный номер. Когда вы в тексте какого-либо руководства видите в секции **SEE ALSO** имя программы или файла, рядом с которым в скобках стоит цифра, например, **passwd(5)**, то это как раз и означает номер раздела. В пятом разделе собраны описания структур различных системных файлов.

Опишем структуру файла паролей **/etc/master.passwd**. Каждая запись этого файла состоит из десяти полей, разделенных двоеточием. Приведем в качестве примера две записи (для пользователей **root** и **test**).

```
root:$1$kzg7j4kB$ZCRWWn.WsJwH02xxQZn241:0:0::0:0:Charlie &:
/home/root:/bin/sh

test:9xDINEMl.Ren2:1002:1002::0:0:Тестовый пользователь:
/home/test1:/bin/csh
```

Перечислим поля:

имя пользователя – оно должно быть уникальным. В имени нельзя использовать заглавные буквы и точки, а также оно не должно начинаться с символа «-»;

пароль – хранится в зашифрованном виде. Если это поле оставить пустым, то можно будет входить в систему без пароля (однако так обычно не делают). Есть специальные – административные – имена пользователей (**bin**, **daemon** и др.), в поле пароля которых стоит звездочка. Это означает, что войти в систему под именем такого пользователя нельзя (можете попробовать и убедиться в этом);

uid – уникальный числовой идентификатор пользователя. В принципе можно иметь пользователей с одинаковыми идентификаторами (и в файле паролей есть такой пример – это пользователи **root** и **toor**, имеющие идентификатор 0), однако лучше так не делать без особой необходимости;

gid – числовой идентификатор группы, в которую помещается пользователь при входе в систему;

класс пользователя – это наименование той конфигурации системных ресурсов, которая связана с именем данного пользователя. В нашем

примере это поле осталось пустым (два двоеточия подряд), значит, по умолчанию пользователь **test** будет отнесен к классу **default** (вспомните файл **/etc/login.conf**, упомянутый выше), а пользователь с идентификатором, равным 0, будет отнесен к классу **root**. Описание файла **/etc/login.conf** можно также посмотреть с помощью команды **man**:

```
man login.conf
```

время смены пароля – оно указывается в секундах, беря в качестве начала отсчета 1 января 1970 года (дата, имеющая в ОС UNIX особое значение). Если в этом поле стоит ноль, то пароль действует без ограничения времени;

время, после которого действие учетной записи для этого пользователя прекращается – аналогично времени смены пароля;

описание пользователя – здесь в свободной форме можно описать назначение данного пользователя;

домашний каталог пользователя – каталог, в который пользователь помещается сразу после входа в систему;

login shell – командный интерпретатор, который будет запущен для данного пользователя при его входе в систему.

ПРИМЕЧАНИЕ. Суперпользователю **root** после установки операционной системы назначен командный интерпретатор **/bin/csh**. Если вам больше нравится **/bin/sh** или какой-то другой, то его можно изменить с помощью программы **vipw**, о которой речь пойдет в следующем разделе.

Файл **/etc/passwd** состоит из записей, в которых сохранено только семь из десяти полей, составляющих запись файла **/etc/master.passwd**. Исключены следующие поля: пароль, время истечения срока действия пароля и время истечения срока действия самой учетной записи для данного пользователя. Покажем пример записей из этого файла:

```
root:*:0:0:Charlie &:/home/root:/bin/sh
test:*:1002:1002:Тестовый пользователь:/home/test1:/bin/csh
```

В системе существуют специальные пользователи (с числовыми идентификаторами меньшими 100), которые предназначены не для входа в систему с их именами, а для выполнения специальных системных функций.

9.6. Программа vipw

В системе есть удобная программа для редактирования файла паролей – **vipw**. С ее помощью можно изменить, например, домашний каталог пользователя, командный интерпретатор, описание пользователя. Нельзя изменять числовой идентификатор и пароль. Программа **vipw** для редакти-

рования файла паролей вызывает тот текстовый редактор, который определен в переменной программного окружения **EDITOR** (см. файл **.profile** в вашем домашнем каталоге). Если вы не назначили этой переменной иное значение, то будет использоваться редактор **vi.** Чтобы посмотреть программное окружение, используйте команду **env**. Отредактировав нужные вам поля в файле паролей, вы должны выполнить сохранение изменений стандартными средствами текстового редактора, который был вызван программой **vipw**, а затем завершить работу с редактором (выйти «из него»). Программа **vipw** выполнит некоторые дополнительные действия, а именно создание хешированных файлов паролей **pwd.db** и **spwd.db**. Эти файлы используются при аутентификации пользователя, пытающегося войти в систему. Поскольку указанные файлы являются хешированными (вспомните, что хеширование применяется для ускорения доступа к данным), то поиск имени пользователя в них происходит быстро даже в системах с множеством пользователей. Подробнее об этих файлах можно прочесть в электронном руководстве:

```
man vipw
```

В системе есть еще одна программа – **pw**. Она является самой мощной из всех программ, предназначенных для управления пользовательскими учетными записями. Ее рассмотрение выходит за рамки нашего курса. Познакомиться с ней вы можете самостоятельно (см. электронное руководство: **man pw**).

9.7. Назначение владельцев файлам и каталогам

Суперпользователь **root** имеет возможность сменить владельца и группу для любого файла и каталога. Для этого служат команды **chown** и **chgrp**. Прежде чем выполнять эти команды, создайте какой-нибудь файл (чтобы потом его можно было без сожаления удалить) и с помощью команды **ls -l** посмотрите его владельца и группу. А теперь с помощью команды **chown** смените владельца этого файла, например, на пользователя **stud** (или на того пользователя, которого вы создали на предшествующем этапе).

```
chown new_owner file_name
```

Здесь **new_owner** – это имя пользователя, которого вы назначаете новым владельцем файла с условным именем **file_name**. Если вы хотите сменить владельца для всех файлов и подкаталогов в каком-либо каталоге, то используйте параметр **-R**:

```
chown -R new_owner directory_name
```

Смена группы выполняется аналогично как для отдельного файла или каталога, так и для иерархии каталогов, только вместо команды **chown** используйте команду **chgrp**:

```
chgrp new_group file_name

chgrp -R new_group directory_name
```

Можно совместить смену владельца и группы с помощью команды **chown**. В этом случае имена нового владельца и новой группы разделяются двоеточием:

```
chown new_owner:new_group file_name
```

9.8. Программа su

Эта программа позволяет текущему пользователю имитировать вход в систему под именем другого пользователя. Например, вы можете войти в систему под именем **stud**, а затем пожелать получить права суперпользователя, тогда вам нужно ввести команду:

```
su
```

Однако если тот пользователь, под именем которого вы в настоящий момент зарегистрированы в системе, не является членом группы **wheel**, то вы получите сообщение:

```
su: you are not in the correct group to su root.
```

Вам нужно отредактировать файл **/etc/group**. В этом файле есть группа **wheel** (как правило, в самой первой строке). Вы должны вписать вашего пользователя в список членов группы. После этого нужно выйти из системы и снова войти в нее под этим же именем пользователя (при новом входе будет перечитана информация из файла **/etc/group**). Теперь можно повторить команду **su**. У вас потребуют пароль суперпользователя (напоминаем, что его имя – **root**). После ввода пароля вы – суперпользователь. Посмотрите программное окружение с помощью команды **env**, постарайтесь внимательно изучить то, что будет выведено на экран (это нужно для последующего сравнения). Чтобы снова стать «самим собой» используйте уже известную вам команду **exit**.

Теперь выполните еще один опыт: введите команду **su** с параметром (параметром будет простой знак «-»):

```
su -
```

Посмотрите программное окружение с помощью команды **env** и сравните его с тем окружением, которое вы получили, вводя команду **su** без параметра.

С помощью команды **su** вы можете стать не только суперпользователем, но и другим пользователем. При этом если вы уже являетесь суперпользователем, то пароль с вас требовать не будут. Чтобы «притвориться» пользователем **stud**, введите команду:

```
su - stud
```

Программа **su** часто применяется в командных файлах (в частности, в тех файлах, которые выполняются при старте системы). Это бывает нужно тогда, когда какая-либо программа должна быть запущена на исполнение от имени какого-то **конкретного** пользователя. Например, система управления базами данных (СУБД) PostgreSQL (это самая мощная из бесплатных СУБД с открытыми исходными текстами) должна запускаться при старте операционной системы от имени пользователя **postgres**, а вот от имени другого пользователя (даже от имени суперпользователя **root**) она запускаться не станет: в ней есть соответствующая проверка.

Контрольные вопросы и задания

1. Создайте учетные записи для двух новых пользователей. Введите одного из них в группу **wheel**. Вспомните, какие полномочия дает пользователю включение его в эту группу.

2. Удалите одного из вновь созданных пользователей.

3. Как настроить программную среду пользователя? Какие файлы и основные переменные среды вы знаете?

4. С помощью какой команды можно просмотреть состояние программной среды?

5. Смените пароль у вашего нового пользователя.

6. Зарегистрировавшись в системе как обычный пользователь, попробуйте «притвориться» суперпользователем **root**. Что для этого необходимо? К какой группе должна принадлежать ваша учетная запись, чтобы вы могли запустить команду **su** без указания имени пользователя в качестве параметра?

7. В чем отличие между файлами **passwd** и **master.passwd**? В каком виде хранится пароль в файле паролей?

8. В каком файле можно просмотреть все учетные записи? Просмотрите этот файл и убедитесь, что ваша учетная запись существует.

9. Смените владельца для одного из ваших файлов. Смените группу для этого же файла. Создайте каталог для упражнений и скопируйте в него несколько файлов, создайте в нем несколько подкаталогов, а затем смените владельца всех файлов и подкаталогов в этом тренировочном каталоге с помощью одной команды. Какой параметр в этом случае нужно использовать?

10. Основы администрирования файловых систем ОС FreeBSD

Продолжая осваивать основы администрирования ОС UNIX, переходим к управлению файловыми системами. Для грамотного управления ими необходимо:

– изучить назначение и разновидности файлов устройств;

– изучить способы монтирования и размонтирования файловых систем;

– научиться выполнять профилактические работы с файловыми системами.

Для выполнения заданий этой главы вам потребуется знание пароля суперпользователя **root**.

10.1. Файлы устройств

Выполните команду **df** или **mount** (все равно, какую из них). В первой колонке выведенной информации вы увидите значения типа **/dev/ad0s2e**. Это имена так называемых **файлов устройств**. Как вы уже знаете, в ОС UNIX обращения ко всем устройствам выглядят, как обращения к обычным файлам. За счет такого подхода достигается единообразие при работе в системе и упрощается жизнь пользователя системы. Рассмотрим принципы обозначения файлов устройств для жестких дисков. Если ваш компьютер не самый современный, то, скорее всего, в нем установлен диск (или диски) типа IDE. В этом случае в имени файла устройства первыми символами будут **ad**. Далее идет цифра, которая означает номер такого IDE-диска. Нумерация начинается с нуля. Как известно, в персональном компьютере может устанавливаться до четырех IDE-дисков, которые подключаются к контроллерам жестких дисков. Таких контроллеров два, они часто называются Primary IDE и Secondary IDE (т. е. первый и второй). Таким образом, к каждому из них могут подключаться по два диска. Причем, диски тоже имеют порядок при подключении к контроллеру: Master (первый диск) и Slave (второй диск). Для того, чтобы назначить диску роль Master или Slave, необходимо поставить в соответствующее положение специальные переключатели на диске. Возвращаясь к нумерации дисков, получаем такую картину:

ad0 – первый диск на первом контроллере;

ad1 – второй диск на первом контроллере;

ad2 – первый диск на втором контроллере;

ad3 – второй диск на втором контроллере.

Если ваш компьютер более современный, и на материнской плате предусмотрены разъемы типа SATA, тогда к каждому такому разъему может быть подключен только один жесткий диск. В этом случае нумерация дисков отличается от приведенной выше:

ad0 – диск на контроллере SATA0;

ad2 – диск на контроллере SATA1;

ad4 – диск на контроллере SATA4;

ad6 – диск на контроллере SATA5.

В имени файла устройства есть еще один фрагмент, который имеет обозначение типа **s2**, что означает **slice** (т. е. раздел) номер 2. Как известно, на жестком диске может располагаться до четырех операционных систем. Случаи, когда применяются специальные начальные загрузчики, позволяющие иметь на диске более четырех ОС, мы не рассматриваем. Каждая ОС располагается в своем так называемом **разделе**. Так вот, **s2** – это второй раздел на диске. Нумерация разделов начинается с единицы. Кроме того, ОС FreeBSD позволяет создать внутри своего раздела еще и отдельные файловые системы (в типичной конфигурации ОС FreeBSD это **/**, **/home**, **/usr**, **/var**), которым соответствуют буквы **a**, **d**, **e**, **f**. Таким образом, имя устройства, например, **/dev/ad2s3d** означает следующее: это первый диск на втором IDE-контроллере, третий раздел на этом диске, вторая файловая система (возможно, **/home**). Для диска типа SATA это обозначение указывало бы на то, что диск подключен к разъему SATA1 (упрощенно говоря, второму по счету среди таких разъемов).

Особенностью файлов устройств является то, что они не имеют размера файла в его традиционном понимании. Вместо этого они имеют так называемые **старший** и **младший** номера устройств. Выполните приведенную далее команду, но вместо обозначения **ad0s2** подставьте то, которое соответствует вашей установке ОС FreeBSD (выяснить номер диска и раздела вы можете при помощи команд **df** или **mount**):

```
ls -l /dev/ad0s2*
```

ПРИМЕЧАНИЕ. В дальнейших примерах также вместо обозначения **ad0s2** подставляйте то, которое соответствует вашей установке ОС FreeBSD.

Вы увидите эти самые номера в пятой и шестой колонках. Что эти номера обозначают, на данном этапе знать не обязательно. Любопытных отсылаем к электронному руководству:

```
man mknod
```

В более старых версиях ОС FreeBSD для создания файлов устройств использовалась специальная программа **/dev/MAKEDEV**. В версии FreeBSD 8.2 такая программа отсутствует, а вместо нее предусмотрена специальная файловая система **devfs**.

10.2. Монтирование и размонтирование файловых систем

При загрузке операционной системы выполняются операции по монтированию ряда файловых систем (**/**, **/home**, **/usr**, **/var**). Как уже отмечалось

в одной из предыдущих глав, **монтирование** означает включение файловой системы в общую иерархию каталогов. После выполнения этой операции смонтированная файловая система становится доступной для работы с ней.

Для монтирования используется команда **mount**. Чтобы начать ее изучение, сначала выполните команду **mount** без параметров. В первой колонке ее вывода указаны имена файлов устройств, а во второй – каталоги, на которые выполнено монтирование файловых систем. Например, чтобы смонтировать файловую систему, содержащую каталоги пользователей, нужно выполнить такую команду (пожалуйста, выполните ее не сейчас, а после выполнения следующей команды – **umount**):

```
mount /dev/ad0s2d /home
```

Для размонтирования файловой системы используется команда **umount**. Обратите внимание: именно **umount**, а не **unmount**. Чтобы размонтировать эту файловую систему, введите команду:

```
umount /home
```

Если вы попытаетесь ее выполнить, то можете получить сообщение об ошибке в том случае, если вы находитесь в каталоге этой файловой системы или открыли один из ее файлов. Не забывайте, что вы могли открыть один из файлов в этой файловой системе, зарегистрировавшись на другом виртуальном терминале (см. главу 2). Если же вам все-таки удалось ее размонтировать, то опять смонтируйте ее командой, рассмотренной выше.

На вашем компьютере, скорее всего, установлена ОС Windows. Она может использовать два типа файловых систем – FAT32 или NTFS. Выяснить, какова конфигурация разделов ОС Windows на вашем компьютере, вы можете с помощью утилиты администрирования дисков, которая включена в состав ОС Windows.

Чтобы получить доступ к разделу, отформатированному как FAT32, сделайте следующее (обратите внимание, что после номера раздела – **s1** – не указывается никакая буква):

```
mount_msdosfs /dev/ad0s1 /mnt
```

Каталог **/mnt** используется для временного монтирования файловых систем. Если вы укажете вместо него какой-нибудь другой каталог, содержащий файлы, то на период подключения к нему файловой системы эти файлы будут недоступны, но с ними ничего не произойдет. Конечно, не стоит указывать в качестве такой точки монтирования каталоги, например, **/etc** или **/dev**. После монтирования файловой системы ОС Windows вы можете перейти в каталог **/mnt** и получить возможность перемещения по

файловой системе Windows. Вы не можете, однако, запускать программы, написанные для Windows, но можете копировать и удалять файлы, создавать каталоги. Конечно, будьте осторожны, чтобы случайно не удалить системные файлы Windows. Если вы снова выполните команду **mount** без параметров или команду **df**, то увидите пополнение в списке смонтированных файловых систем. Чтобы размонтировать эту файловую систему, введите команду:

```
umount /mnt
```

Если у вас это не получилось, то, возможно, вы не «ушли» из файловой системы Windows (нужно выйти из каталога **/mnt** перед размонтированием).

Наверное, вы обратили внимание на то, что русскоязычные имена файлов и каталогов в ОС Windows не отображаются корректно. Чтобы они все-таки имели надлежащий вид, необходимо команду монтирования дополнить специальным параметром:

```
mount_msdosfs –L ru_RU.KOI8-R /dev/ad0s1 /mnt
```

Выполнив эту модифицированную команду, вы получите сообщение об ошибке, в котором говорится, что не найдена библиотека **libiconv**. Установите недостающую библиотеку из дистрибутивного комплекта CD дисков FreeBSD с помощью утилиты **sysinstall** аналогично тому, как вы устанавливали браузер **lynx** при изучении материала главы 1. Добавим только, что искать эту библиотеку нужно среди пакетов категории **Format conversion utilities**, которая находится в подсистеме **Packages** (пакеты).

Если раздел жесткого диска, в котором установлена ОС Windows (или который был создан уже после ее установки дополнительно), отформатирован как NTFS, то команда монтирования будет такой (параметр **-C koi8-r** позволяет правильно отображать русскоязычные имена файлов и каталогов):

```
mount_ntfs –C koi8-r /dev/ad0s1 /mnt
```

К сожалению, разделы NTFS монтируются в режиме «только для чтения», таким образом, вы можете скопировать файлы из раздела Windows в раздел FreeBSD, но не наоборот.

Необходимо сделать еще одно важное замечание, которое касается так называемого дополнительного, или расширенного (extended), раздела. Если вы устанавливаете две операционных системы Windows на ваш компьютер (например, Windows XP и Windows 2003), то вторая из них будет, скорее всего, установлена в дополнительный (расширенный) раздел. Получить представление о том, какова конфигурация разделов в ОС Windows на вашем компьютере, вы можете с помощью утилиты администрирования

дисков, которая включена в состав ОС Windows. Для монтирования этого раздела необходимо в команде **mount** указывать номер слайса 5 (термин **слайс** уже был введен ранее):

```
mount_ntfs –C koi8-r /dev/ad0s5 /mnt
```

ПРИМЕЧАНИЕ. Если в вашем компьютере два жестких диска и при этом ОС Windows и FreeBSD установлены на разные диски, то вы должны соответственно корректировать приведенные команды, например, вместо ad0 у вас может оказаться ad1 или ad2.

Теперь мы покажем, как смонтировать дискету, отформатированную для DOS/Windows. Снимите на ней защиту от записи и выполните команду:

```
mount_msdosfs /dev/fd0 /mnt
```

Здесь **/dev/fd0** – это файл устройства для дисковода гибких дисков. Скопируйте что-нибудь на дискету или с нее. Также посмотрите на список смонтированных файловых систем, используя команды **mount** и **df**.

Чтобы размонтировать эту файловую систему, выйдите из каталога **/mnt** и введите команду:

```
umount /mnt
```

Дискету также можно смонтировать и на другой каталог, а не только на **/mnt**. Создайте, например, каталог **/home/stud/floppy** и смонтируйте дискету на него.

ВАЖНОЕ ПРИМЕЧАНИЕ. Нельзя вынимать дискету из дисковода, не размонтировав ее предварительно. Если вы так сделаете, то операционная система может «зависнуть».

Коснемся и столь популярного в настоящее время устройства флэш-памяти. Когда вы вставите его в USB-разъем, на экран будет выведена информация об устройстве. Эта информация выводится поверх командной строки. Для того чтобы освободить командную строку, просто нажмите клавишу Enter. Устройство флэш-памяти может быть смонтировано в ОС FreeBSD командой

```
mount_msdosfs –L ru_RU.KOI8-R /dev/da0s1 /mnt
```

Если вам не требуется поддержки русскоязычных имен файлов, то можно опустить параметр **-L ru_RU-KOI8-R**.

После завершения работы с подключенным устройством флэш-памяти выйдите мз каталога **/mnt** и размонтируйте его:

```
umount /mnt
```

Теперь можно удалить устройство из USB-разъема. На экран так же, как и при вставке устройства флэш-памяти в USB-разъем, будет выведено информационное сообщение, которое можно убрать, нажав клавишу Enter.

О том, как смонтировать CD-диск, было сказано в главе 2. Диски формата DVD монтируются аналогично, если ваш дисковод поддерживает работу с ними. В главе 1 была приведена краткая форма команды, которая работает потому, что в файле **/etc/fstab** (см. следующий раздел) есть строка, описывающая файл устройства для CD дисков. Полная команда выглядит таким образом:

```
mount_cd9660 /dev/acd0 /cdrom
```

10.3. Файл fstab

В каталоге **/etc** есть файл **fstab**. Посмотрите его содержимое. Вы увидите, что в нем есть несколько строк, каждая из которых описывает одну файловую систему. При этом указывается имя файла устройства и каталог, на который выполняется монтирование этой файловой системы. Указывается также тип файловой системы. Раздел **swap** – это раздел подкачки, аналогичный файлу подкачки Windows. Раздел **swap** недоступен пользователю непосредственно. Колонка **Options** указывает режимы доступа к файловой системе, например, **rw** – означает чтение/запись. Подробно о файле **fstab** можно узнать, посмотрев электронное руководство:

```
man fstab
```

Все файловые системы, перечисленные в этом файле (за исключением **/cdrom**) , монтируются в процессе начальной загрузки системы. Файловая система **/cdrom** является исключением, т. к. для нее указан параметр **noauto**.

10.4. Проверка файловой системы (программа fsck)

Сейчас вы должны загрузить систему в однопользовательском режиме. Для этого проделайте следующее. В начале загрузки, когда вам дается 9-секундный тайм-аут и предлагается выбрать режим загрузки операционной системы, нажмите клавишу «4» для выбора режима **Single user mode**. Система начнет загружаться и, наконец, предложит вам указать полный путь к командному интерпретатору или нажать Enter, если вас устраивает **/bin/sh**:

```
Enter full pathname of shell or RETURN for /bin/sh:
```

Нажмите Enter, т. к. **/bin/sh** вас вполне устроит. Вы оказались в однопользовательском режиме с правами пользователя **root**. Причем, заметьте, что у вас не спросили пароль. Это поведение системы можно изменить, заставив ее запрашивать пароль пользователя **root** и в этом режиме.

Выполните команды: **ps -ax**, **df**, **mount**, попробуйте перейти на другой терминал (например, на второй: клавиши Ctrl-Alt-F2). Что вы видите? Внимательно изучите все, что выводят эти команды на экран.

Теперь настало время сказать о программе **fsck**. Она служит для проверки файловых систем. Она должна выполняться только на размонтированных файловых системах. Давайте выполним ее, проверив поочередно все файловые системы, которые созданы в нашей системе (если не помните, то введите команду: **cat /etc/fstab**), например:

```
fsck /dev/ad0s2e
```

После аварийного завершения работы с ОС FreeBSD проверки с помощью программы **fsck** выполняются автоматически, на больших дисках это может занимать некоторое время.

После проверки всех файловых систем их можно смонтировать, введя команду

```
mount -a
```

Будут смонтированы все файловые системы, перечисленные в файле **/etc/fstab**. Теперь можно перевести вашу ОС с многопользовательский режим работы, для чего просто введите команду **exit**. Произойдет «дозагрузка» тех программ, которые обычно загружаются при обычной загрузке без вашего вмешательства.

Контрольные вопросы и задания

1. Для чего нужны файлы устройств?

2. Поясните назначение каждого элемента в обозначении: /dev/ad0s2e.

3. Для чего нужна файловая система **/dev**?

4. Смонтируйте и размонтируйте дискету.

5. Смонтируйте и размонтируйте файловую систему Windows, учитывая, что в ней есть файлы и каталоги, имеющие названия на русском языке.

6. Смонтируйте диск в дисководе CD-ROM.

7. Как вы думаете, почему нельзя вынимать дискету из дисковода, предварительно не размонтировав ее?

8. Для чего нужен файл **/etc/fstab**?

9. Объясните назначение программы **fsck.**

10. Загрузите систему в однопользовательском режиме. Почему проверку файловых систем предпочтительнее проводить в однопользовательском режиме? Выполните проверку всех файловых систем, а затем переведите систему в многопользовательский режим.

11. Управление процессами в ОС FreeBSD

Управление процессами в многозадачной многопользовательской среде является одной из забот администратора системы. В этой главе вы познакомитесь с возможностями, которые ОС UNIX предоставляет пользователю для управления процессами (на примере FreeBSD). Вы узнаете:

– каким образом можно изменить приоритет процесса;

– как можно принудительно завершить процесс;

– что такое фоновые процессы;

– как сделать так, чтобы определенные процессы периодически выполнялись в системе без непосредственного участия администратора.

11.1. Управление процессами

Для просмотра процессов, выполняющихся в системе, используется, как вы уже знаете, команда **ps**. Выполните ее без параметров и с параметрами **-ax**. Вы увидите, что во втором случае выводится большее число строк. Это объясняется тем, что в дополнение к процессам, запущенным на выполнение текущим пользователем (или от его имени), выводится информация обо всех остальных процессах: как системных, так и принадлежащих другим пользователям системы. Поля в колонках имеют следующее назначение:

PID – уникальный номер процесса (process identifier);

TT – управляющий терминал, с которым связан процесс (т. е. терминал, на который процесс выводит информацию по умолчанию);

STAT – состояние процесса (выполняется, «спит» и т. д.), которое обозначается различными символами (см. **man ps**);

TIME – процессорное время, использованное процессом;

COMMAND – строка команды, которая запустила процесс на выполнение.

Есть еще одна команда для просмотра процессов, выполняющихся в системе – **top**. Выполните ее. Поля в некоторых колонках имеют следующее назначение:

PID – уникальный номер процесса (process identifier);

USERNAME – имя пользователя, являющегося владельцем процесса;

PRI – приоритет процесса (чем это значение меньше, тем приоритет выше; значение может быть и отрицательным)

NICE – та составляющая приоритета процесса, на которую может влиять пользователь, являющийся владельцем процесса;

SIZE – размер образа процесса;

COMMAND – строка команды, которая запустила процесс на выполнение.

Для выхода из программы **top** нажмите клавишу q.

Для того чтобы назначить более высокий (или более низкий) приоритет процессу, нужно при запуске программы использовать команду **nice**:

```
nice --10 joe my_file.txt
```

С помощью электронного руководства **man** выясните, для чего указывается два символа «-» в вышеприведенной команде, а также выясните, может ли обычный пользователь повышать приоритетность своих процессов.

Если процесс уже выполняется, то изменить его приоритетность можно с помощью команды **renice:**

```
renice +5 -p 38454
```

Здесь параметр -p 38454 обозначает номер процесса, для которого изменяется значение признака **nice**. Также используя электронное руководство **man**, выясните, какие ограничения по изменению приоритетности процессов налагаются на обычных пользователей в отличие от пользователя **root.**

Иногда бывает необходимо принудительно завершить какой-либо процесс. Делается это путем посылки нужному процессу так называемого **сигнала**. Для этого используется команда **kill** следующим образом:

```
kill 4512
```

где 4512 – номер процесса (это, конечно, условный номер, а реальный номер процесса, который вы хотите «убить», можно узнать с помощью команды **ps**). Иногда такая команда не помогает, и процесс «не хочет» завершаться. Тогда нужно добавить параметр **-9**. В этом случае ему посылается сигнал 9 (а не 1, как это происходит по умолчанию, когда номер сигнала не указан). Девятый сигнал должен заставить процесс завершиться. Например:

```
kill -9 4512
```

Попробуйте зайти в систему с правами обычного пользователя (не **root**) и завершить процесс, принадлежащий другому пользователю. Получили сообщение об ошибке? Конечно, **root** может уничтожать и чужие процессы, так что нужно соблюдать осторожность в многопользовательской среде.

11.2. Фоновые процессы

Как вы уже знаете, список выполняющихся процессов можно просмотреть с помощью команды **ps -ax**. Теперь скажем о так называемых **фоновых** процессах. Для примера воспользуемся командой для поиска файлов (выражение **2>/dev/null** позволяет перенаправить сообщения об ошибках на фиктивное устройство, т. е. попросту подавить эти сообщения):

```
find / -name profile -print 2>/dev/null > file_names
```

Модифицируем команду, добавив в командной строке символ «&»:

```
find / -name profile -print 2>/dev/null > file_names &
```

Обратите внимание, что командная строка «освободилась» для ввода новой команды. А теперь выполните команду **ps -ax**. Найдите в ее выводе строку, в которой указана команда **find** и прочитайте номер ее процесса в первой колонке. Теперь представьте, что поиск файлов очень затянулся, и вы хотите его прервать. Нажатие клавишей Ctrl-C не поможет. Поступить нужно так:

```
kill номер_процесса
```

Здесь вместо слов **номер_процесса** укажите тот номер, который вы прочитали в первой колонке. Выполнение программы должно прекратиться. Бывают случаи, когда это не помогает. Тогда в команду **kill** добавляют параметр **-9**:

```
kill -9 номер_процесса
```

11.3. Смена системной даты и времени

Давайте научимся изменять системную дату и время. Это будет полезно, в частности, и при изучении следующего раздела – периодических процессов.

Для смены системной даты и времени используется одна команда: **date**. Чтобы назначить время, например, 14 часов 28 минут, не меняя даты, введите в командной строке:

```
date 1428
```

Чтобы назначить время и дату, например, 14 часов 28 минут 5 ноября 2007 года, введите:

date 200711051428

В ответ будет выведено:

```
понедельник, 5 ноября 2007 г. 14:28:00 (KRAT)
```

Если назначить дату из периода «летнего» времени, то пометка (KRAT) изменится на (KRAST). Например:

date 200709051428

В ответ будет выведено:

```
среда, 5 сентября 2007 г. 14:28:00 (KRAST)
```

Верните дату и время в правильное (текущее) состояние. Эта команда умеет довольно много. Изучить ее возможности можно с помощью электронного руководства **man date**.

11.4. Периодические процессы

Часто в системе возникает необходимость выполнения каких-либо действий **периодически**. Для реализации такой возможности служит программа-демон по имени **cron**, которую вы можете обнаружить в вашей системе с помощью команды

ps –ax | grep cron | grep –v grep

Напомним, что команда **grep cron** выбирает из всего списка процессов только тот, который нам нужен, а команда **grep -v grep** отбрасывает строки, в которых выводится информация о процессе, выполняющем саму команду **grep**.

Подсистема **cron** имеет два уровня: пользовательский и общесистемный. Мы расскажем о втором уровне. В каталоге **/etc** есть файл **crontab**. Его структура описана в электронном руководстве. Для просмотра введите

man 5 crontab

Напоминаем, что цифра 5 обозначает номер секции электронного руководства, из которой следует взять информацию. Для **crontab** есть еще информация в первой секции. Возьмем фрагмент файла **/etc/crontab** и внимательно его изучим (в реальный файл добавлены наши комментарии).

```
# в этой строке указаны моменты времени, в которые должны
# выполняться те или иные команды
```

```
# минута  час   день   месяц    день   имя          команда
#              месяца          недели  пользователя
# minute  hour  mday   month    wday   who          command

# команда /usr/libexec/atrun выполняется каждые пять минут
# от имени пользователя root
*/5  *    *    *    *    root /usr/libexec/atrun

# Rotate log files every hour, if necessary
# команда newsyslog выполняется каждый час в ноль-ноль минут
# от имени пользователя root
0    *    *    *    *    root newsyslog

# Perform daily/weekly/monthly maintenance
# ... каждый день в 3 часа 1 минуту
1    3    *    *    *    root periodic daily
# ... каждое воскресенье (это 6-й день недели) в 4 часа
# 15 минут
15   4    *    *    6    root periodic weekly
# ... первого числа каждого месяца в 5 часов 30 минут
# ПРИМЕЧАНИЕ. Программа periodic - это скрипт на языке shell,
#             который находится в каталоге /usr/sbin.
30   5    1    *    *    root periodic monthly
# Adjust the time zone if the CMOS clock keeps local time,
# as opposed to UTC time. See adjkerntz(8) for details.
# ... каждый день с 0 часов до 5 часов в 1 и 31 минуту
# (т.е. в 00:01, 00:31, 01:01, 01:31 ... 05:01, 05:31)
1,31 0-5  *    *    *    root adjkerntz -a
```

Для примера введите такую строку в файл **/etc/crontab** (будем каждую минуту посылать по электронной почте пользователю **stud** на локальной машине файл **/etc/profile**). Обратите внимание, что поля в этой строке разделены символами табуляции.

```
*/1  *    *    *    *    root cat /etc/profile | mail -s "Pro-
file" stud
```

Значения первых пяти полей: минуты — число от 0 до 59, часы — число от 0 до 23, день месяца — число от 1 до 31, номер месяца в году — число от 1 до 12, день недели — число от 0 до 7.

Для каждого конкретного параметра можно задать несколько значений через запятую. Например, если в поле "часы" написать 1,4,22, то задание будет запущено в 1 час ночи, в 4 часа утра и в 22 часа. Можно задать интервал — 4-9 будет означать, что программу нужно запускать каждый час в период с 4 до 9 часов включительно. Символ '*' означает "все возможные значения". Например, указание '*' в поле "часы" будет означать "запускать каждый час". Символ '/' служит для указания дополнительной

периодичности задания. Например, '*/3' в поле "часы" означает "каждые три часа".

Ниже приведены примеры фрагментов файлов **/etc/crontab**:

1. Простейший сценарий cron, который будет автоматически запускаться каждые три часа во вторник и в пятницу /home/vasya/script.pl

```
0    */3  *    *    2,5  /home/vasya/script.pl
```

2. Выполнять раз в полчаса в 0 и 30 минут каждого часа

```
*/30 *    *    *    *    $HOME/bin/every_half_hour
```

3. Выполнять четыре раза в час в 0,15,30 и 45 минут каждого часа

```
*/15 *    *    *    *    $HOME/bin/every_half_hour
```

4. Выполнять через час в 0,2,4,6,8,10,12,14,16,18,20 и 22 часов

```
0    */2  *    *    *    $HOME/bin/every_other_hour
```

5. Выполнять еженедельно по воскресеньям в 5:20

```
20 5  *    *    7    $HOME/bin/weekly
```

6. Выполнять ежемесячно 1 числа в 6:30

```
30   6    1    *    *    $HOME/bin/monthly
```

Контрольные вопросы и задания

1. Выполните команду

```
ps -ax
```

и объясните назначение каждой колонки в ее выводе.

2. Для чего служит команда **nice**? Запустите какой-либо процесс с определенным приоритетом. Понизьте приоритет одного из процессов, запущенных вами.

3. Уничтожьте один из процессов, запущенных вами. Что нужно делать, если процесс «не хочет» завершаться?

4. Запустите поиск какого-нибудь файла в фоновом режиме, а вывод программы **find** перенаправьте в файл.

5. Что такое периодические процессы? В каком файле производятся назначения таких процессов?

6. Назначьте отправку сообщения по электронной почте какому-либо пользователю каждый понедельник в 14 часов. Измените системную дату и время, чтобы искусственно приблизить этот момент времени и убедиться в правильной работе вашей команды. Затем верните системную дату и время в правильное состояние.

12. Запуск и останов операционной системы и регистрация системных событий

В этой главе вы кратко познакомитесь с системой регистрации событий – **syslog**, а также с порядком начальной загрузки и останова системы.

12.1. Система syslog и файлы регистрации (файлы-журналы)

В операционной системе UNIX есть возможность регистрировать события, происходящие в процессе ее работы. Такими событиями могут быть, например, регистрация пользователей в системе (т. е. вход в систему), перезагрузка операционной системы, различные ошибки, возникающие при работе различных подсистем и т. п. Все эти события обрабатывается с помощью системы **syslog**. Вы можете увидеть программу-демон **syslogd** с помощью команды

```
ps –ax | grep syslogd | grep –v grep
```

Обычно **syslogd** запускается при начальной загрузке ОС. В каталоге **/etc** есть файл конфигурации этой системы – **syslog.conf**. Он состоит из строк, разделенных на две части: первая часть указывает тип события, а вторая – куда направить сообщение об этом событии. Тип события состоит также из двух частей: первая часть – это наименование подсистемы ОС UNIX (например, **mail** – подсистема электронной почты, **security** – подсистема безопасности), а вторая часть – уровень серьезности события (например, **info** – информационное сообщение, **err** – сообщение об ошибках). Например, пара **mail.info** говорит о том, что нужно регистрировать все сообщения с уровнем серьезности **info** и выше (т. е. **err** тоже), исходящие от подсистемы электронной почты. Можно использовать символы *, которые обозначают ВСЕ подсистемы или ВСЕ типы событий. Например, **mail.*** означает все сообщения, генерируемые подсистемой электронной почты. Еще пример: ***.err** – все сообщения с уровнем серьезности **err** и выше (например, **crit** – критические), генерируемые всеми подсистемами. В первом поле каждой строки может быть указано несколько типов сообщений, разделенных точкой с запятой. Второе поле указывает место для «складирования» сгенерированных сообщений. Например, **/var/log/maillog**. Сообщения можно не только помещать в файлы на локальном компьютере, но также перенаправлять на другие компьютеры. Как правило, файлы-журналы с сообщениями хранятся в каталоге **/var/log**. Вы можете их посмотреть с помощью обычного редактора. Только не редактируйте их. Подробное описание системы **syslog** смотрите в электронном руководстве: **man syslogd**, а описание файла конфигурации – **man syslog.conf**.

12.2. Запуск и останов системы

Процесс начальной загрузки ОС FreeBSD детально описан в электронных руководствах (см. **man boot** и **man loader)**. Отметим лишь, что этот процесс проходит в несколько стадий. В системе есть несколько так называемых загрузчиков, начиная с первого и заканчивая третьим. Все они находятся в каталоге **/boot**. Вдобавок еще существует и так называемый **boot manager**, который выводит меню при запуске компьютера и позволяет выбрать операционную систему для загрузки. Лучший способ изучить процесс начальной загрузки – сделать так: начать с просмотра электронного руководства **man boot**, затем – **man loader**. В конце каждого из этих руководств есть разделы **FILES** и **SEE ALSO**, в которых указаны файлы и электронные руководства, имеющие отношение к рассматриваемому вопросу. Вспомните, что запись, например, **boot(8)** означает, что искать электронное руководство для **boot** нужно в восьмой секции, т. е. команду нужно давать такую:

```
man 8 boot
```

Хотя в том случае, когда одноименных руководств в различных секциях нет, можно номер секции не указывать. А вообще лучше всего изучить систему электронных руководств с помощью команды

```
man man
```

ПРИМЕЧАНИЕ. При изучении загрузчиков нужно соблюдать осторожность, а именно: не копировать файлы загрузчиков в системные каталоги, не пытаться редактировать их, не редактировать их конфигурационные файлы без полного понимания того, что вы делаете. Эта глава носит ознакомительный характер, она не предполагает «активных» действий с загрузчиками.

Вернемся к процессу загрузки. После того, как загрузчик третьей ступени загрузит ядро операционной системы, начинается обработка конфигурационных файлов в каталоге **/etc**. Это файлы, имена которых начинаются с символов **rc**. Сделаем небольшое отступление. Выполните команду **ps -ax** и найдите в самом начале того списка, который она выведет, строку, в которой в поле COMMAND указано: **/sbin/init**. Этот процесс имеет номер 1 и является самым главным процессом в системе. Так вот этот самый **init** и продолжает работу, начатую группой загрузчиков. Он запускает файл **/etc/rc** (если вы заглянете в файл **/etc/rc**, то в нем сможете прочесть то, что только что прочли здесь). Кстати говоря, обязательно посмотрите **man init**. Конечно, запомнить все это сразу вряд ли возможно, но вы, по крайней мере, будете представлять процесс начальной загрузки хотя бы в самых общих чертах.

Теперь уже скрипт **/etc/rc** продолжает работу. Он считывает многочисленные конфигурационные файлы (например, **/etc/rc.conf** и ему подоб-

ные) и выполняет запуск необходимых системных программ-демонов, которые работают в течение всего времени работы ОС. Конечно, без хорошего знания языка shell понять процесс начальной загрузки ОС в деталях невозможно.

Ну и, наконец, запускается программа **/usr/libexec/getty**, которая и организует знакомые вам приглашения для входа в систему – **login:** Посмотрите еще раз файл **/etc/ttys** – запуск **/usr/libexec/getty** регулируется оттуда.

В каталоге **/etc** есть очень важный файл **inetd.conf**. Он командует запуском программ-демонов, отвечающих за работу в компьютерной сети. Посмотрите содержимое этого файла (не забывайте о том, что можно использовать **man inetd.conf**). В этом файле вы увидите имена команд UNIX: **ftp**, **telnet**, **shell** и др. Эти строки указывают на то, что указанные функции ОС могут быть активизированы при необходимости. Эта необходимость возникает, например, когда необходимо предоставить возможность входа на ваш компьютер с другого компьютера по протоколу **ftp**. В таком случае программа-демон **inetd** запускает программу-демон **ftpd** на вашем компьютере для того, чтобы обработать этот вход по протоколу **ftp** с другой машины. Однако сразу после установки операционной системы все команды в файле **inetd.conf** закомментированы (т. е. в начале каждой строки стоит символ #), таким образом, доступ к компьютеру по сети практически невозможен. Чтобы «открыть» вход на ваш компьютер по **ftp**, раскомментируйте строку для **ftp**, чтобы «открыть» **telnet** – строку для **telnet** и т. д. После внесения изменений в файл **inetd.conf** необходимо сделать так, чтобы эти изменения стали известны системе. Самый очевидный способ решения данной задачи – перезагрузить систему. Второй, более интеллектуальный и быстрый способ, – это послать сигнал с кодом 1 процессу, который выполняет программу **inetd**:

```
kill -1 номер_процесса_inetd
```

Для останова системы используется не только известная комбинация клавишей Ctrl-Alt-Delete, но и команда **shutdown**. Для того чтобы немедленно остановить систему, введите

```
shutdown -p now
```

И напоследок еще пара команд, имеющих отношение к процессу запуска системы. Первая из них **dmesg**. О ней нечего говорить – просто выполните ее, и вам все станет ясно (вспомните про способности клавиши Scroll Lock, когда вам нужно будет просмотреть все, что эта команда выведет на экран).

Еще команда – **finger**. Она покажет вам всех пользователей, подключенных к вашей ОС. Также выполните ее и посмотрите, что она выдает. Не забывайте про электронное руководство **man.**

Команда **who** похожа на нее. Выполните эту команду и сравните с предыдущей. У команды **who** есть параметр: **am i**, позволяющий вам определить, под каким именем вы работаете в системе. Введите:

```
who am i
```

А можно и без пробелов – вот так:

```
whoami
```

Контрольные вопросы и задания

1. Для чего нужна система **syslog**?

2. Посмотрите содержимое файла **/etc/syslog.conf** и постарайтесь разобраться в его содержимом с помощью электронного руководства **man**. Попробуйте разобраться, какая информация записывается в файл **/var/log/security**, какая – в **/var/log/maillog**, а какая – в **/var/log/messages**?

3. Какие файлы находятся в каталоге **/var/log**? Каково их назначение?

4. Назовите стадии процесса начальной загрузки ОС FreeBSD.

5. Посмотрите содержимое файлов **/etc/rc.conf** и **/etc/rc**. Попытайтесь разобраться в последовательности запуска тех или иных программ и выполнения тех или иных команд в файле **/etc/rc**. Попытайтесь увидеть, как связаны между собой файлы **/etc/rc.conf** и **/etc/rc**.

6. Каким образом можно запретить или, наоборот, открыть доступ на компьютер по протоколам **ftp** и **telnet**?

7. Объясните команды:

```
shutdown -p now
```

и

```
shutdown +5
```

13. Сетевые возможности ОС FreeBSD

Изучив материал этой главы, вы узнаете, каким образом вы можете взаимодействовать с пользователями, работающими на других компьютерах сети. Вы познакомитесь с программой электронной почты, а также с программами **ftp** и **telnet.**

13.1. Подготовка к работе в сети

В нашем учебном пособии мы исходим из того, что вы самостоятельно изучаете предложенный учебный материал в том порядке, в котором он здесь изложен. Следовательно, вы самостоятельно настраиваете вашу систему FreeBSD шаг за шагом. Для изучения сетевых возможностей FreeBSD необходимо сделать ряд дополнительных настроек системы.

Сначала добавьте в файл **/etc/rc.conf** следующую строку:

```
inetd_enable=”YES”
```

Затем в файле **/etc/inetd.conf** разрешите доступ к вашему компьютеру по протоколам ftp и telnet. Для этого раскомментируйте следующие строки:

```
ftp     stream tcp  nowait  root  /usr/libexec/ftpd    ftpd -l
telnet  stream tcp  nowait  root  /usr/libexec/telnetd telnetd
```

Теперь нужно перезагрузить компьютер, чтобы изменения вступили в силу.

ПРИМЕЧАНИЕ. Если бы мы не вносили изменения в файл **/etc/rc.conf**, то можно было бы обойтись и без перезагрузки, послав сигнал 1 процессу **inetd**, чтобы он вновь считал конфигурацию из файла **/etc/inetd.conf**.

13.2. Отправка сообщений электронной почты

Отправить сообщение по электронной почте можно двумя способами. При использовании первого из них сначала нужно подготовить текст письма (например, в редакторе **joe**) и сохраните его в файле. Затем производится отправка письма с помощью команды

```
mail -s "Привет от ..." stud@your.machine.ru < файл_письма
```

В этой команде «**Привет от ...**» является заголовком вашего письма; **stud@your.machine.ru** – адрес электронной почты, в котором **stud** означает имя адресата, а **your.machine.ru** – имя компьютера, на котором этот ад-

ресат работает. Символ < означает, что текст письма передается почтовой программе **mail** с помощью переадресации стандартного ввода. Поскольку в домашних условиях у вас может не быть возможности работы в локальной сети, то вместо имени компьютера **your.machine.ru** введите слово **localhost**, которое означает, что сообщение электронной почты будет адресовано пользователю на этом же компьютере. Предположим, что вами создана учетная запись пользователя **stud**, а текст письма сохранен в текущем каталоге в файле **my_letter.txt**, тогда команда будет выглядеть так:

```
mail -s "Мое первое письмо" stud@localhost < my_letter.txt
```

Рассмотрим второй способ отправки сообщения по электронной почте. Введите команду

```
mail stud@localhost
```

Вам будет предложено ввести заголовок письма (Subject). Затем вводите текст, он может располагаться на нескольких строках. Признаком завершения письма является символ «.» (точка), введенный с самого начала строки, и нажатие клавиши Enter. Завершить письмо можно также и другим способом – нажатием комбинации клавишей Ctrl-D. Ваше письмо будет отправлено.

Для просмотра электронных сообщений введите команду **mail** без параметров. Если на ваше имя уже поступили какие-либо почтовые сообщения, то список заголовков вы увидите на экране. Для просмотра сообщений нужно вводить их номера и нажимать клавишу Enter. Для просмотра конкретного сообщения вводите его номер. Можно просто нажимать клавишу Enter, тогда сообщения будут просматриваться с первого до последнего по порядку. Для просмотра заголовков поступивших почтовых сообщений введите команду **h** с номером сообщения. Если сообщения с указанными номерами имеются, то программа **mail** выведет их список. Для получения краткой подсказки по работе с почтовой программой введите символ «?» и нажмите Enter. Получить более подробные инструкции можно по команде **man mail**.

Можно отправить письмо, уже находясь «внутри» почтовой программы. Для этого введите:

```
m адрес_электронной_почты
```

Дальнейшие действия вам уже знакомы.

13.3. Копирование файлов на ваш компьютер с другого компьютера

Задания этого раздела также можно выполнять и на одном компьютере, при этом вместо имени компьютера **your.machine.ru** вводите слово **localhost**.

Запустите программу **ftp (file transfer protocol)**, введя команду

```
ftp your.machine.ru
```

В этой команде **your.machine.ru** означает имя компьютера, с которого вы будете копировать файлы (или просто **localhost** в домашних условиях).

Вам будет предложено ввести имя пользователя (вы увидите приглашение **Name:**). Введите имя **stud**, а затем пароль, который вы назначили этому пользователю. Если вы не ошиблись при вводе, то получите сообщение:

```
User stud logged in.
```

Вы подключились к другому компьютеру. Если даже вы выполняете это задание на одном компьютере, все равно все операции выглядят так, как если бы вы работали в компьютерной сети. Так что вы получите достоверное представление о работе в сети. Теперь можно просмотреть содержимое каталога на удаленной машине с помощью команды **ls**, узнать имя текущего каталога с помощью команды **pwd**. Для перехода из одного каталога в другой используйте команду

```
cd имя_каталога
```

Для того чтобы скопировать какой-либо файл с другого компьютера на ваш компьютер, введите команду

```
get имя_файла
```

ПРИМЕЧАНИЕ. При работе на одном компьютере нужно учитывать, что файлы, которые вы копируете командой **get**, будут помещаться в тот каталог, из которого вы запускали программу **ftp**.

Если имя файла длинное, то можно ввести лишь первые символы имени и нажать клавишу Tab – программа **ftp** дополнит имя файла сама. Однако в том случае, когда имеется несколько файлов с такой комбинацией символов в начале их имен, вам будет показан список таких файлов. Тогда нужно ввести еще дополнительно символы и опять нажать клавишу Tab. При перемещении по дереву каталогов с помощью команды **cd** можно

также вводить лишь часть имени каталога и дополнять его с помощью клавиши Tab. В программе **ftp** можно с помощью клавишей «стрелка вверх» и «стрелка вниз» просматривать команды, введенные вами ранее, чтобы избежать повторного их набора. По команде **help** вы получите полный список всех команд, которые понимает эта программа. Для получения краткой подсказки по конкретной команде введите **help имя_команды.** Клавиша Tab работает аналогично и в отношении команд. Например, для завершения работы с программой служит команда **close**, но вы можете ввести лишь символы **cl** и нажать Tab. Однако даже этого можно не делать: программе **ftp** достаточно столько символов от имени команды, чтобы ее можно было однозначно истолковать. Так что хватит и просто двух символов **cl** и нажатия клавиши Enter. Для окончательного выхода из программы **ftp** введите еще команду **quit.**

Для входа на ftp-серверы применяется, как правило, так называемый **анонимный** вход. В этом случае в качестве имени пользователя нужно ввести **ftp**, а качестве пароля – **ваш адрес электронной почты**. Вообще-то, на самом деле пароль при анонимном входе на ftp-сервер вообще не требуется, просто ввод вашего адреса электронной почты является правилом хорошего тона. Если вы просто нажмете клавишу Enter в ответ на приглашение ввести пароль, все будет нормально – вас впустят на ftp-сервер.

13.4. Вход на другой компьютер при помощи программы telnet

Если программа **ftp** предназначена для копирования файлов с одного компьютера на другой, то программа **telnet** служит для входа на другой компьютер и выполнения там каких-либо программ. Введите команду

```
telnet -8 имя_компьютера
```

В этой команде **имя_компьютера** означает то же, что и в предыдущем случае при работе с программой **ftp**, а параметр **-8** позволяет вам переключать латинские и русские буквы после входа на другой компьютер. После ввода команды вы увидите приглашение, аналогичное тому, которое предлагается вам на вашем компьютере, т. е. **User:** и **Pasword:.** Если вы работаете в локальной сети, то введите такое имя пользователя, которое присутствует на компьютере, к которому вы подключаетесь. Если же вы работаете в домашних условиях, то введите имя **stud**, а затем пароль, который вы назначили этому пользователю. Если вы не ошиблись при вводе, то теперь вы зарегистрированы в системе на чужом компьютере. Вы можете работать на нем так же, как и на своем, т. е. использовать все те команды, которые вы уже изучили. Для начала можно запустить Demos Commander, введя команду **deco.** Если выполнить команду **who** на том компьютере, ку-

да вы только вошли, то вы увидите в списке пользователей и вашего пользователя **stud** (в скобках указано имя компьютера, с которого был совершен вход по протоколу telnet, в приведенном примере вход был выполнен с этого же компьютера).

```
root             ttyv0    27 ноя 17:57
stud             ttyv1    27 ноя 18:00
stud             ttyp0    27 ноя 18:48 (localhost)
```

Обратите внимание, что для пользователя, подключившегося по протоколу telnet, во второй колонке указан тип терминала ttyp0. Это так называемый псевдотерминал, в отличие от виртуального терминала ttyv*.

Для выхода из сеанса работы с программой **telnet** нужно выполнить команду **exit** или нажать клавиши Ctrl-D, как и для выхода из системы на своем компьютере. При выходе вы получите сообщение:

```
Connection closed by foreign host.
```

Контрольные вопросы и задания

1. Создайте файл-письмо, отправьте это письмо с помощью почтовой программы **mail**. Убедитесь, что письмо отправлено. Прочитайте отправленное письмо. Как снова просмотреть список писем? Как прочитать какое-нибудь письмо?

2. Если вы работаете в компьютерной аудитории, то зайдите на соседний компьютер по протоколу **telnet** и выполните там какую-нибудь команду, например, **ls.**

3. Если вы работаете в компьютерной аудитории, то зайдите на ftp-сервер и скопируйте на свой компьютер в свой домашний каталог 2–3 файла.

14. Графическая подсистема X Window

В этой главе вы познакомитесь с графической подсистемой **X Window**. Вы научитесь устанавливать ее реализацию, разработанную консорциумом **X.org**, установите графическую настольную среду **KDE**. Эта среда содержит множество программ: от браузера до текстового редактора, от инструментов для разработки программного обеспечения до компьютерных игр.

Каким бы мощным ни был интерфейс командной строки, многие пользователи, тем не менее, хотели бы работать в графической среде. Операционная система FreeBSD может предоставить и такую возможность. В ее состав входит графическая подсистема **X Window**, которая, как правило, называется **X11** и в данной версии ОС FreeBSD представлена реализацией **X.org**.

Мы покажем, как установить и настроить эту подсистему.

1. Зарегистрируйтесь в системе как пользователь **root.**

2. Проверьте наличие в вашей системе необходимых предварительных настроек, а именно:

– должна быть проведена «русификация» системы, как было описано в главе 2;

– в файле **/etc/rc.conf** должны быть следующие строки:

```
moused_enable=”YES”
mousechar_start=3
```

3. Если при установке операционной системы вы установили не все ее компоненты, то сейчас вам необходимо установить подсистему **X.org**. Для этого вставьте установочный диск в дисковод и запустите программу **sysinstall**. Затем выберите в главном меню этой программы пункт **Configure**, в следующем меню – пункт **Packages**, а затем укажите устанавливаемый модуль – **xorg-7.5**.

Необходимио в файл /etc/rc.conf добавить следующие строки:

```
hald_enable="YES"
dbus_enable="YES"
```

Перезагрузите компьютер.

4. Перейдите в домашний каталог пользователя **root**. Создайте конфигурационный файл подсистемы **X11**, для чего выполните команду

```
Xorg -configure
```

Эта команда выполняется не в интерактивном режиме. Она тестирует оборудование вашего компьютера, а затем выводит информацию на

экран и создает конфигурационный файл **xorg.conf.new** в текущем каталоге.

5. Сделайте пробный запуск графической подсистемы. При этом запуске ничего выдающегося не происходит: монитор переводится в графический режим, экран закрашивается в серый цвет, и появляется указатель «мыши». При перемещении «мыши» ее указатель должен также перемещаться по экрану. Итак, делаем пробный запуск:

```
Xorg -config xorg.conf.new -retro
```

Если что-то будет работать некорректно, можно обратиться за справкой к файлу-журналу **/var/log/Xorg.0.log**.

8. Если вас все устраивает, то необходимо скопировать конфигурационный файл **xorg.conf.new** в системный каталог **/etc/X11** и переименовать его там в **xorg.conf**. Мы предлагаем использовать первый из указанных каталогов. Для копирования файла можно либо использовать файловый менеджер **deco**, либо выполнить команду

```
cp xorg.conf.new /etc/X11/xorg.conf
```

Таким образом, в домашнем каталоге пользователя **root** сохранится файл под именем **xorg.conf.new**, а в системном каталоге его копия будет иметь имя **xorg.conf**.

9. Настало время выполнить более ответственный тест. Выполните команду

startx

Она запускает графическую подсистему, а в рамках этой подсистемы по умолчанию запускается программа **twm**, которая относится к категории **оконных менеджеров**. На экране создается три окна, одно из которых имеет заголовок «Login». Вы можете попробовать вводить команды операционной системы, например, **ls, pwd** и др. Для выхода из графической подсистемы введите команду **exit** в окне с заголовком «Login».

10. Как вы увидели, оконный менеджер **twm** является довольно скромной программой. Поэтому мы предлагаем заменить его на более мощную настольную графическую среду (desktop environment) **KDE**. Для ее установки опять воспользуйтесь установочным диском FreeBSD. В разделе **Packages** выберите пакет **kde4-4.4.5**. Вы увидите, что при этом еще целый ряд пакетов будут отмечены символом D. Это означает зависимый пакет (dependency). Все эти пакеты будут установлены, как единое целое. Если ваш компьютер не очень мощный, то для установки **KDE** может потребоваться значительное время (до 30–40 минут).

11. Для того чтобы вместо оконного менеджера **twm** запускалась графическая среда **KDE**, создайте в домашнем каталоге пользователя **root**

файл с именем **.xinitrc** (обратите внимание на точку в начале имени файла) и введите в него всего одну строку «exec /usr/local/kde4/bin/startkde» (в файле кавычек быть не должно). Это можно сделать как в текстовом редакторе, так и путем выполнения команды (обратите внимание, что параметр **startkde** пишется без пробела)

```
echo "/usr/local/kde4/bin/startkde" >> ~/.xinitrc
```

Знак «~» перед именем файла означает домашний каталог пользователя. Если этот каталог является текущим, то команду можно упростить:

```
echo "/usr/local/kde4/bin/startkde" >> .xinitrc
```

12. Снова выполните команду **startx**. Если вы нигде не ошиблись, то должна запуститься среда **KDE**. При первом ее запуске вам будет предложено выполнить начальные настройки. Вы можете просто выбирать те, что предлагаются по умолчанию, поскольку впоследствии вы сможете эти настройки изменить уже в рамках **KDE**.

13. Еще раз обратимся к файлу **xorg.conf.new**. Теперь нам необходимо подключить русскоязычные шрифты. Для этого в секции «**Files**» добавьте следующую строку (обязательно ПЕРЕД другими строками с именем параметра **FontPath**):

```
FontPath "/usr/X11R6/lib/X11/fonts/cyrillic/"
```

Для того чтобы иметь возможность переключения клавиатуры с англоязычной раскладки на русскоязычную. В среде KDE выполните следующую последовательность переходов по меню: **Settings -> System Settings -> Input Devices -> Keyboard -> Layouts -> Add layout-> Layout -> Russia** Затем нужно добавить раскладку клавиатуры для России: **Shortcuts for switching layputs -> Main shortcuts -> Ctrl+Shift.**

Теперь преключение между русской и английской раскладками клавиатуры будет осуществляться с помощью клавишей Ctrl-Shift.

15. Вы можете запускать графическую среду **KDE** и при регистрации в системе с правами других пользователей, например, **stud**. Для этого зарегистрируйтесь в системе под именем конкретного пользователя и создайте файл **.xinitrc** в его домашнем каталоге. Затем запускайте графическую среду командой **startx**.

Чтобы временно перейти с графического терминала на алфавитно-цифровой, используйте комбинацию клавишей Ctrl-Alt-F*, для возвращения на графический терминал – клавиши Alt-F9.

ПРИМЕЧАНИЕ. Весь процесс установки графической подсистемы, описанный в нашем учебном пособии, в более подробной форме представлен в системной документации: **/usr/local/share/doc/freebsd/ru_RU.KOI8-R/books/handbook/book.html**.

Заключение

Итак, вы познакомились с основами работы в среде операционной системы FreeBSD. Естественно, вы пока еще не можете назвать себя квалифицированным специалистом по администрированию ОС UNIX. Для достижения этой высокой цели вам нужно много трудиться и неустанно идти вперед. А для этого следует обратиться к тем книгам, которые приведены в списке рекомендуемой литературы. Не стоит ограничиваться только этим списком: ведь постоянно выходят новые книги. Мы надеемся, что после изучения нашего практического курса вам будет легче ориентироваться в книжном мире (или, точнее, книжном море), посвященном ОС UNIX.

Для того чтобы вы представили, какой объем работы ожидает вас, приведем лишь несколько примеров того, что знает квалифицированный администратор ОС UNIX.

В ОС FreeBSD (как и во всех ОС UNIX) есть возможность сконфигурировать ядро системы с учетом ваших потребностей и реальных возможностей аппаратуры вашего компьютера. Ядро собирается непосредственно из исходных текстов, находящихся в каталоге **/sys** (на самом деле это ссылка на каталог **/usr/src/sys**). Это является очень мощным средством в умелых руках.

Не менее важные и интересные вопросы: UNIX в локальных и глобальных сетях; UNIX как сервер приложений; системное программирование в UNIX; программирование на языке **shell**; сервер доменных имен **DNS**; сетевая файловая система **NFS;** UNIX как сервер электронной почты и т. д. и т. п.

Желаем вам, уважаемый читатель, больших успехов в дальнейшем изучении операционной системы UNIX!

Рекомендуемая литература

1. Эбен, М. FreeBSD. Энциклопедия пользователя [Текст] : пер. с англ. / Майкл Эбен, Брайан Таймэн. – К. : ДиаСофтЮП, 2005. – 864 с.

2. Немет, Э. UNIX: руководство системного администратора : для профессионалов [Текст] : пер. с англ. / Э. Немет, Г. Снайдер, С. Сибасс, Т. Р. Хейн. – СПб. : Питер ; К. : BHV, 2008. – 928 с.

7. Тейнсли, Д. Linux и UNIX: программирование в shell. Руководство разработчика [Текст] : пер. с англ. / Д. Тейнсли. – К. : BHV, 2001. – 464 с.

Учебное издание

МОРГУНОВА Ольга Николаевна
ТЫНЧЕНКО Валерия Валериевна

Операционная система FreeBSD
Вводный курс

Учебное пособие

Печатается в авторской редакции
Компьютерная верстка *О. Н. Моргуновой*

Подписано в печать 2011. Формат 60×84/16. Бумага офсетная.
Печать плоская. Усл. печ. л. 7,44. Уч.-изд. л. 4,90. Тираж 50 экз.
Заказ № ___.

Отпечатано в отделе копировально-множительной техники
Сиб. гос. аэрокосмич. ун-та.
660014, г. Красноярск, просп. им. газ. «Красноярский рабочий», 31.