Федеральное агентство по образованию

Государственное образовательное учреждение
высшего профессионального образования
«Сибирский государственный индустриальный университет»

Кафедра систем информатики и управления

ВВЕДЕНИЕ В UNIX

Учебно-методическое издание
по дисциплине «Операционные системы, среды, оболочки».
Специальности: 080801 – Прикладная информатика (в управлении);
230201 – Информационные системы и технологии

Новокузнецк
2006

УДК 681.3.06
ББК 32.973

Рецензент
Доктор технических наук, профессор,
заведующий кафедрой систем автоматизации
Кулаков С.М.

Введение в Unix: Уч.-метод. изд. / Сост.: М.В. Ляховец, С.П. Огнев;
СибГИУ. – Новокузнецк, 2006. - ____с., ил.

Рассмотрены практические аспекты использования средств операционных систем семейств Unix. Произведено введение в среду выполнения системы Unix в целом, способы обращения пользователей к ее различным частям. Представлены средства для практических каждодневных задач по сопровождению файлов - для копирования и сохранения файлов и для удаления ненужных файлов. Также обращено внимание на собственную среду пользователей и ее администрирование.

Учебно-методическое издание предназначено для студентов всех форм обучения специальностей 080801 – Прикладная информатика (в управлении); 230201 – Информационные системы и технологии.

Содержание

Основы работы в Unix

Начало и конец сеанса работ

Каждый пользователь имеет:

- имя пользователя (для установления взаимодействия пользователей и начисления расходов);
- пароль пользователя (для контроля входа в систему и защиты своих данных).

Пользователи могут быть объединены в группы (для работы над проектами и т.п.) для разделения общих ресурсов, тогда еще есть имя группы пользователей.

Один пользователь, называемый суперпользователь (`superuser`) является администратором системы (его имя - `root`). В частности, он регистрирует всех прочих пользователей.

При входе в систему необходимо на запрос

```
login:
```

ввести имя пользователя, например:

```
login: student
```

и нажать на клавишу `Enter`. На следующий запрос надо ввести пароль пользователя. Необходимо заметить, что при вводе пароля символы не отображаются, однако пароль вводится в память компьютера.

После входа в систему будет отображаться системное приглашение, которое, по умолчанию, состоит из знака $.

Можно сменить свой пароль в любое время с помощью команды `passwd`:

```
login: student
password:
$ passwd
Changing password for student
Old password:
New password:
Retype new password:
$
```

Остановить ввод в систему или выйти из системы (завершить работу) можно с помощью `Ctrl-d`.

Исправление ошибок при наборе текста команды

Исправление последней буквы путем набора:

- `backspace`
- `^H (Ctrl-H)`
- `delete`

Исправление последней строки путем набора:

- `^X`
- `^V`
- `@`

Некоторые простые команды

Вывести на экран текущую дату:

```
$ date
Mon Apr 24 14:42:25 UTC 2006
$
```

Чтобы узнать всех активных пользователей, работающих в системе:

```
$ who
root       tty0    Apr  24   14:38
student    ttyp0   Apr  24   14:53  (10.1.2.1)
student    ttyp1   Apr  24   14:48  (10.1.2.10)
. . .
$
```

Чтобы узнать под каким именем пользователя находитесь:

```
$ who am i
student    ttyp0   Apr  24   14:53  (10.1.2.1)
```

Формат команд

Команды разделяются либо концом строки, либо точкой с запятой:

```
$ who; date
student    ttyp0   Apr  24   14:53  (10.1.2.1)
Mon Apr 24 14:42:25 UTC 2006
```

Командная строка - последовательность слов, разделенных пробелами. Первое слово командной строки есть команда; остальные - параметры.

Типы параметров:

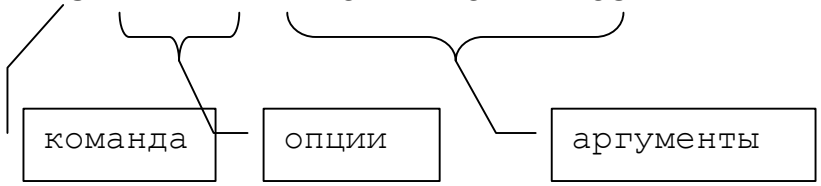
- имя файла = идентификатор (использует символы `a-z`, `A-Z`, `0-9`, `_`, `.`,`-`);
- опция (ключ) уточняет смысл команды; начинается обычно с минуса. Например `-al` (может быть со знаком `+` или без знака); смысл опции зависит от команды;
- выражение - описывает обычно строку символов или является строкой.

Порядок параметров в команде:

```
command options expression filename(s)
```

Примеры команд:

```
ls  -l -i  file1 file2 file3
```




```
rm old.news bod.news
rm -fr goodies.c baddies.o
grep -o "mary" people
```

Приостановка - продолжение вывода на экран

`^s` - временный останов вывода на экран;

`^q` - продолжает вывод на экран информации, которая была остановлена при помощи <`^s`>.

Останов выполнения команды

^C или клавиша BREAK

Каталоги и файлы

При регистрации пользователя ему назначается администратором собственный каталог пользователя (Home directory).

Файловая система

Файловая система является краеугольным камнем операционной системы UNIX. Она обеспечивает логический метод организации, восстановления и управления информацией. Файловая система имеет иерархическую структуру. Файл, который является основной единицей системы UNIX, может быть: обыкновенным файлом, справочником, специальным файлом или символическим каналом связи.

- **Обыкновенные файлы** являются набором символов. Они используются для хранения любой информации. Они могут содержать тексты для писем или отчетов, коды программ, которые вы написали, либо команды для запуска ваших программ. Однажды создав обыкновенный файл, пользователь может добавить нужный материал в него, удалить материал из него, либо удалить файл целиком.
- **Справочники** (каталоги, директории) являются супер-файлами, которые могут содержать файлы или другие справочники. Обычно файлы, содержащиеся в них, устанавливают отношения каким-либо способом. Пользователь может создать каталоги, добавить или удалить файлы из них или удалить каталоги. Все справочники, которые создаёт пользователь, будут размещены в домашнем справочнике пользователя. Этот справочник назначается системой во время входа в систему. Никто кроме привилегированных пользователей не может читать или записывать файлы в этот справочник без разрешения пользователя-владельца. Система UNIX также содержит несколько справочников для собственного использования. Структура этих справочников аналогична во всех системах UNIX. Этот справочник, включающий в себя несколько системных справочников, размещается непосредственно под справочником root (/), который является исходным в файловой структуре UNIX. Все справочники и файлы иерархически располагаются ниже.
- **Специальные файлы** соответствуют физическим устройствам, таким как терминал, дисковое устройство, магнитная лента или канал связи. Система читает и записывает из/в специальные файлы также как и в обыкновенные файлы. Однако запросы системы на чтение и запись не приводят в действие нормальный механизм доступа к файлу. Вместо этого они активизируют драйвер устройства, связанный с файлом, приводя, возможно, в действие головки диска или магнитной ленты.
- **Символические каналы связи** - это файлы, которые указывают на другие файлы.

Структура корневого каталога

Как правило, корневой каталог имеет следующую структуру (рисунок 1), но администратор системы может изменять эту структуру.

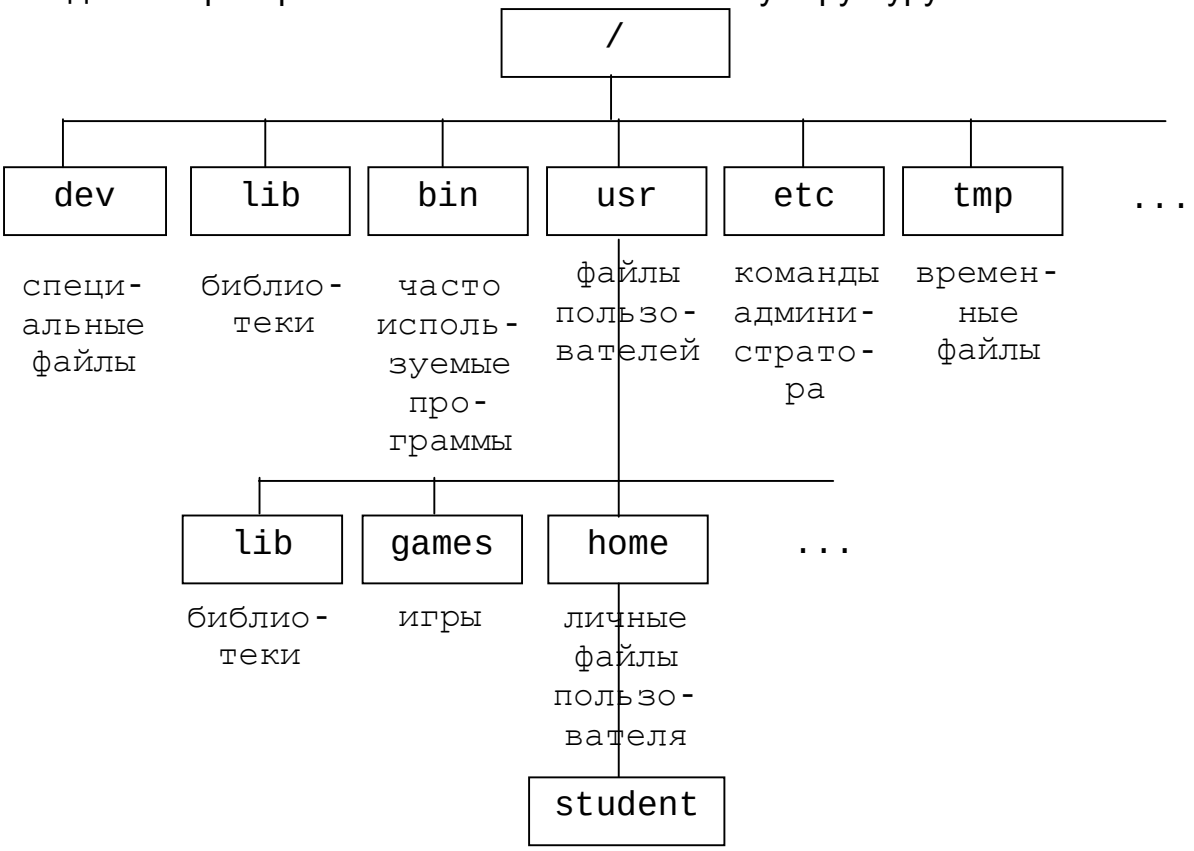
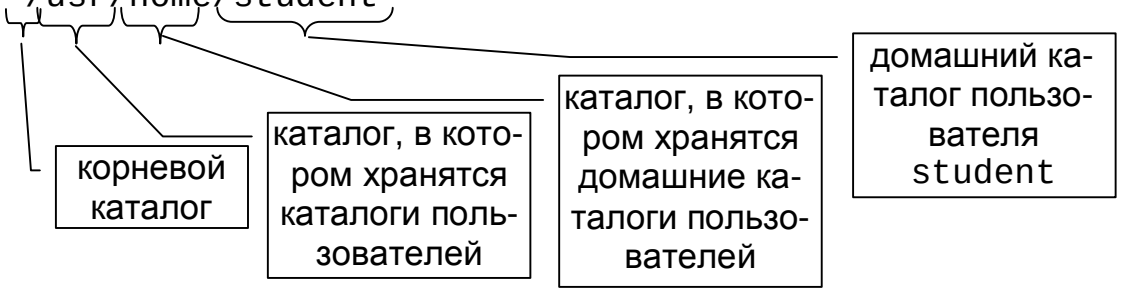


Рисунок 1 – Пример структуры корневого католога.

Печать рабочего каталога

Узнать полное имя текущего каталога можно с помощью команды pwd:

```
$ pwd
/usr/home/student
```



В данном случае, команда pwd сообщает полное имя текущего рабочего каталога. Каждый файл и справочник системы UNIX идентифицируется уникальным именем пути. Имя пути показывает местоположение файла или справочника и обеспечивает направление поиска его. Существует 2 типа имени пути: полное и родственное.

Полное имя пути

Полное имя пути (иногда называемое абсолютным именем пути) дает направление, которое начинается в справочнике root и показывает путь далее по уникальной последовательности справочников к конкретному справоч-

нику или файлу. Отличительной особенностью полного пути является то, что это имя начинается с символа косая черта (/). Последнее имя в полном имени пути может быть либо именем файла, либо именем справочника. Все другие имена пути должны быть справочниками.

Родственное имя пути

Родственное имя пути дает направления, которые начинаются в текущем рабочем справочнике и ведут вверх или вниз через серию справочников к конкретному файлу или справочнику. Двигаясь вниз из текущего справочника, можно получить доступ к своему файлу или справочнику. Двигаясь вверх из текущего каталога, можно пройти через родительские справочники к родителю всех системных справочников, т.е. к root.

Родственное имя пути начинается с имени справочника или файла.

Одна точка (.) означает текущий справочник, две точки (..) - справочник, непосредственно находящийся выше текущего справочника в иерархии файловой системы. Справочник, представленный двумя точками, называется родительским для справочника, обозначенного одной точкой.

Например, пользователь находится в справочнике student, который содержит справочники draft, letters и файл file1. Родственным именем пути для каждого из них является просто его имя.

Справочник draft, принадлежащий справочнику student, содержит файлы outline и table. Родственное имя пути от student к файлу outline:

```
draft/outline
```

Чтобы подняться к родителю вашего текущего справочника, можуj ввести две точки (..). Это означает, что если пользователь находится в справочнике draft, то .. является именем пути к student, и ../.. является именем пути к родительскому справочнику для student, т.е. к home.

Правила именования справочников и файлов

Пользователь может давать справочникам или файлам любые имена в соответствии со следующими правилами:

- допустимы все символы, за исключением /;
- некоторые имена лучше не использовать, такие как пробел, табуляция и следующие: ? " # $ ^ ( ) ; < > [ ] | \ * @ ' ~ &. Если вы воспользуетесь символами пробел или табуляция в имени файла или справочника, то вы должны заключить имя в двойные кавычки в командной строке;
- избегайте использования знаков + - или . в качестве первого символа в имени файла;
- система UNIX различает большие и маленькие буквы.

Печать содержимого каталога

Все справочники в файловой системе имеют информацию о содержащихся в них файлах и справочниках, такую как: имя, размер и дата последней модификации. Можно получить эту информацию о справочниках, задав команду ls. Команда ls перечисляет имена всех файлов и подсправочников в

указанном справочнике. Если справочник не указан, то команда ls напечатает информацию о файлах и справочниках в текущем справочнике. Чтобы напечатать имена файлов и подсправочников в справочнике, отличном от текущего без перехода из текущего справочника, вы должны указать имя справочника.

Синтаксис команды:

ls [опции] [имя_пути]

Имя пути может быть либо полным именем пути требуемого справочника, либо родственным.

$ ls - *печать текущего каталога;*

$ ls / - *печать каталога root;*

Часто используемые ключи команды ls

Команда ls может содержать ключи, которые перечисляют специфичные атрибуты файла или подсправочника. Рассмотрим часто используемые опции:

Распечатать содержание в расширенном формате

Ключ -l позволяет отобразить содержание справочника в расширенном формате. Этот формат включает в себя: режим, число связей, владелец, группа, размер в байтах и время последней модификации каждого файла. Например:

$ ls -l / - *печать каталога root полная*.

Выдаст на экран:

```
total 49
drwxr-xr-x   2 root  wheel  1024 Feb 27 16:56 bin
drwxr-xr-x   5 root  wheel   512 Feb 27 17:29 boot
dr-xr-xr-x   4 root  wheel   512 Apr 24 14:35 dev
drwxr-xr-x  17 root  wheel  2048 Apr 12 09:36 etc
drwxr-xr-x   3 root  wheel  1024 Feb 27 16:56 lib
drwxr-xr-x   2 root  wheel   512 Nov  5  2004 mnt
drwxrwxrwt   4 root  wheel   512 Apr 24 14:35 tmp
drwxr-xr-x  17 root  wheel   512 Feb 27 17:47 usr
drwxr-xr-x  22 root  wheel   512 Apr 24 14:35 var
...
```

Необходимо отметить, что здесь вывод команды ls показан не полностью. Рассмотрим вывод этой команды более подробно.

Первая выводная строка (total 49) показывает объем дискового пространства в байтах. Последующие строки дают представление о справочниках и файлах в указанном справочнике (в данном случае /). Первый символ в каждой строке (d, -, l, b или c) указывает тип файла:

- d = справочник
- - = обыкновенный файл
- l = символическая связь(канал)
- b = специальный блочный файл
- c = специальный символьный файл.

Следующие девять символов, которые являются либо буквами, либо дефисами, идентифицируют право на чтение и использование файла или справочника.

Далее следует цифра - счетчик связей. Для справочника этот счетчик показывает число справочников, расположенных под ним, плюс два (для самого справочника и справочника-родителя).

Следующим является регистрационное имя владельца файла (в данном случае `root`), и за ним - групповое имя файла или справочника (`wheel`).

Следующее число показывает длину файла или справочника в байтах. Месяц, день и время последней модификации файла - в предпоследней колонке. В последней колонке представлено имя справочника или файла.

Перечислить все файлы в справочнике

Имена некоторых файлов начинаются с точки (например, `.profile`). Когда имя файла начинается с точки, он не включается в список, распечатываемый командой `ls`. Чтобы распечатать такие файлы, необходимо использовать ключ `-a`.

Например, команда:

```
$ ls –a
```

Наряду с другими файлами текущего справочника покажет и файлы:

```
.               - ссылка на текущий каталог
..              - ссылка на родительский каталог
.profile        - файл, используемый системой в
                служебных целях
```

и другие.

Распечатать содержание в укороченном формате

Ключи `-C` и `-F` команды `ls` используются достаточно часто. Вместе эти ключи распечатывают подсправочники и файлы справочников и помечают исполняемые файлы символом *, справочники - символом /, символическую связь(каналы) - символом @.

Так, самостоятельно проанализируйте вывод команды

```
$ ls –F
```

выделив из полученного списка справочники, исполняемые файлы и обычные файлы.

Изменение рабочего каталога

Изменение рабочего каталога производится командой `cd`.

Синтаксис команды:

```
cd имя_пути_нового_справочника
```

Любое допустимое имя пути (полное или родственное) может использоваться в качестве аргумента команды `cd`. Указанный справочник, в случае успешного перехода, становится текущим справочником.

Например, чтобы перейти из справочника `student` в подчиненный `work`, введите команду `cd work`. Проверить новое местоположение можно, введя команду `pwd`. Экран терминала будет выглядеть следующим образом:

```
$ cd work
```

```
$ pwd
/usr/home/student/work
$
```

То есть этот справочник стал текущим.

В команде cd, наравне с родственным именем пути, можно использовать и полное имя. Например, чтобы перейти из справочника work обратно, необходимо использовать команду:

```
cd /usr/home/student
```

или

```
cd ..
```

Еще пример:

```
$ cd /etc
$ ls –l          - печать команд администратора
$ cd /usr
$ ls -l bin      - редко используемые команды
$ cd             - без параметров - возврат к собственному домашнему каталогу (/home/student).
```

Создание каталога пользователем

Домашний каталог пользователя, являющийся корнем его ветви файловой системы, создается администратором. Пользователь может создать нижележащие каталоги командой mkdir.

Синтаксис команды:

```
mkdir имя_нового_справочника(ов)
```

Например, подсправочник draft создается при помощи следующей команды, выдаваемой из справочника student (/home/student):

```
$ mkdir draft
```

Если необходимо создать несколько справочников одновременно, то можно перечислить имена вновь создаваемых справочников через пробел, так команда:

```
$ mkdir draft1 letters bin
```

создаст три справочника (draft1, letters и bin).

Если необходимо построить иерархический список справочников, то надо перейти к подсправочнику с помощью команды cd и в нем построить дополнительные подсправочники.

В качестве самостоятельного задания попытайтесь создать иерархическую структуру справочников, аналогичную представленной на рисунке 2.

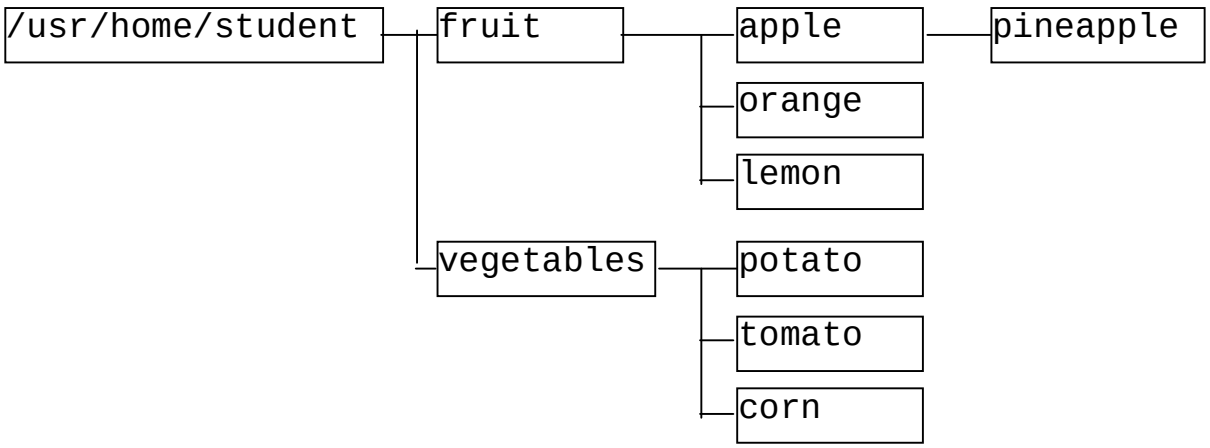


Рисунок 2 – Пример иерархической структуры справочников.

Печать содержимого файла

В системе UNIX существует три команды для распечатки содержимого файлов: cat, pg, pr:

- Команда cat выводит содержимое файла на экран терминала или в другой файл или новую команду.
- Команда pg особенно полезна, если необходимо прочитать содержимое большого файла, так как она отображает текст файла постранично.
- Команда pr форматирует указанные файлы и отображает на терминал или направляет вывод на печать.

Команда cat

Отобразить содержимое указанного файла или файлов на терминал (стандартный вывод, обычно, дисплей) можно с помощью команды cat.

Синтаксис команды:

cat [-u] *имя_файла* ...

Команда cat последовательно читает каждый файл (если необходимо вывести на экран содержимое нескольких файлов, то их перечисляют в командной строке через пробел) и записывает его в файл стандартного вывода. Если в качестве стандартного файла вывода задан не терминал, то осуществляется поблочная буферизация выводимых данных, иначе — построчная буферизация. Опция –u отменяет всякую буферизацию.

Пример:

```
$ cat /etc/motd   - посмотреть файл (message of today)
FreeBSD 5.3-RELEASE (GENERIC) #0: Fri Nov  5 04:19:18 UTC
2004

Welcome to FreeBSD!
```

и далее.

Этот файл, создаваемый администратором для текущих объявлений, обычно печатается автоматически при входе в систему (login).

Необходимо отметить, что рассматриваемая команда выводит содержимое указанных файлов на стандартный вывод, который обычно является дисплеем. Однако, пользователь всегда может воспользоваться переадреса-

цией ввода-вывода и перенаправить, например, вывод команды cat не на дисплей, а в файл. Рассмотрим подробнее возможности переадресации ввода-вывода.

Переадресация ввода-вывода

Существует три направления ввода-вывода:

- стандартный ввод (stdin, дескриптор 0),
- стандартный вывод (stdout, дескриптор 1),
- стандартный протокол (stderr, дескриптор 2).

Как правило, команды берут исходные данные из стандартного ввода и помещают результаты в стандартный вывод.

Стандартные ввод, вывод и протокол можно переназначить. Делается это с помощью следующих структур:

1. Для переназначения стандартного ввода (дескриптор файла 0):

```
< имя_файла
```

(теперь стандартный ввод будет браться из указанного файла, т.е. те данные, которые находятся там будут считаться, грубо говоря, набранными на клавиатуре);

2. Для переназначения стандартного вывода (дескриптор файла 1):

```
> имя_файла
```

(теперь результат выполнения команды будет записан в указанный файл; если он не существует, то будет создан);

3. Многострочный стандартный ввод:

```
<< строка
```

(ввод происходит со стандартного ввода, пока не встретится указанная <строка> (не включая ее) или конец файла; если <строка> является пустой цепочкой литер, то вводимый текст завершается пустой строкой),

4. Для стандартного вывода

```
>> имя файла
```

(если файл существует, то выводимая информация добавляется в конец этого файла, иначе файл создается),

5. Указание стандартного ввода с помощью дискриптора

```
<&  цифра
```

(в качестве стандартного ввода объявляется файл, ассоциированный с дескриптором <цифра>);

6. Аналогично для стандартного вывода

```
>&  цифра
```

(в качестве стандартного вывода объявляется файл, ассоциированный с дескриптором <цифра>);

7. Закрытие стандартного ввода

```
<&-
```

8. Закрытие стандартного вывода

```
>&-
```

Если любой из этих конструкций предшествует цифра, то с указанным файлом будет ассоциирован дескриптор, равный указанной цифре, вместо 0 и 1 по умолчанию. Например,

```
2>имя_файла
```

то при выполнении команды диагностическое сообщение будет направляться в указанный файл, следовательно

```
2>&1
```

совместит файлы стандартного вывода и диагностики при выполнении команды.

Таким образом, команда

```
$ cat /etc/motd > myfile1
```

не выведет содержимое файла `motd` на экран, а запишет его в файл с именем `myfile1`, причем содержимое последнего будет переписано.

В качестве самостоятельного задания попробуйте вывести на экран и в файл содержимое таких файлов, как: `.login`, `.profile` (находящихся в домашнем справочнике пользователя), `/usr/src/README`.

Также с помощью переадресации ввода-вывода возможно создать новый файл, например, команда:

```
$ > myfile2
```

создаст пустой файл с именем `myfile2`.

Команда pg

Команда `pg` позволяет распечатывать содержимое файла или файлов на терминал. Синтаксис команды:

`pg имя-файла(ов)`

После того, как `pg` отобразит страницу текста, она напечатает подсказку "двоеточие" (`:`), которая служит сигналом ввода последующей инструкции. Возможной инструкцией может быть запрос вывода следующей страницы содержимого файла, либо запрос на поиск указанного символа по образцу. В таблице 1 приведены допустимые инструкции команды `pg`.

Таблица 1 Инструкции команды `pg`

Команда	Функция
`h`	Помощь по инструкциям команды `pg`
`q` или `Q`	Прекращает режим команды `pg`
`Enter`	Отображает следующую страницу текста
`l`	Отображает следующую строку текста
`d` или `^d`	Отображает дополнительную половину страницы текста
`.` или `^l`	Вновь отображает текущую страницу текста
`f`	Пропускает одну страницу текста и распечатывает следующую за ней
`n`	Начинает распечатывать следующий указанный в командной строке файл
`p`	Отображает предыдущий указанный в командной строке файл
`s`	Отображает последнюю страницу текста текущего файла
`/pattern`	Осуществляет поиск вперед указанного шаблона символов
`?pattern`	Осуществляет поиск назад указанного шаблона символов

Некоторые команды могут быть введены с цифрой впереди. Например:

`+1` - *отображает следующую страницу;*

`-1` - *отображает предыдущую страницу;*

1 - *отображает первую страницу текста.*

Размер отображаемой страницы зависит от типа терминала. Например, терминал способен отображать 24 строки; значит страница текста определяется 23 строки плюс одна строка для двоеточия.

Например, чтобы отобразить содержимое файла README из справочника /usr/src/, введите команду:

pg /usr/src/README

Когда весь файл будет прочитан, в последней строке появится:

(EOF):

Подсказка : (двоеточие) предлагает вам ввести новую команду. Для окончания работы команды надо ввести команду q.

Команда pr

Команда pr используется для форматирования и печати содержимого файла. Она форматирует заголовки, количество страниц и печатает файл на экране терминала. Синтаксис команды:

pr [опция] [файл] ...

Если не указан ни одна из допустимых опций, то команда pr сформирует вывод в одну колонку, страница будет содержать 66 строк, и тексту будет предшествовать короткий заголовок. Заголовок состоит из 5 строк: две пустые строки, строка, содержащая дату, время, имя файла и номер страницы и далее две пустые строки. Возможные опции представлены в таблице 2. Опции применяются ко всем выводимым на терминал файлам, но их можно переустанавливать, задавая между файлами.

Таблица 2 Опции команды pr

Опция	Функция
–число	Выполнять вывод в указанное количество колонок.
+число	Пронумеровать страницы, начиная с указанного номера.
-h	Использовать следующий параметр в качестве заголовка страниц.
-w число	Установить ширину страницы равной указанному числу литер вместо 72 по умолчанию.
-f	Использовать для перехода на новую страницу литеры перевода формата вместо литер новой строки.
-l число	Установить длину страницы равной указанному числу строк вместо 66 по умолчанию.
-t	Не выводить пятистрочные «шапку» и «концевик», обычно сопровождающие каждую страницу.
-s символ	Отделять колонки одной указанной литерой вместо подходящего количества пробелов. Если с опущено, используется литера табуляции.
-m	Печатать все файлы одновременно, каждый в своей колонке.

Команда pr часто используется с командой lp для получения копии текста на бумаге в том виде, в каком он был введен в файл.

Для самостоятельного изучения работы команды pr выполните печать файла /usr/src/README и проанализируйте вывод команды.

Просмотр конца файла

Команда `tail` печатает конец указанного файла. Синтаксис команды:
`tail [-F | -f | -r] [-b число | -c число | -n число] [имя_файла ...]`

По умолчанию команда `tail` печатает 10 последних строк. Например:

```
$ tail /usr/src/README

usr.bin         User commands.

usr.sbin        System administration commands.


For information on synchronizing your source tree with one or more of
the FreeBSD Project's development branches, please see:

http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/synching.html
```

Явно можно задать количество (со знаком -) или номер строки, от которой печатать до конца (со знаком +):

```
$ tail -3 /usr/src/README     - три последние строки;
$ tail +6 /usr/src/README     –последние строки, начиная с 6-й
```

Определение типа файла

Для определения типа файла используется команда `file`. При использовании этой команды файлы, тип которых надо узнать, перечисляются через пробел в командной строке после команды, например:

```
$ file /bin/ls /usr/bin etc/passwd usr/include/stdio.h
/bin/ls: pure executable              - исполняемый;
/usr/bin: directory                   - каталог;
/etc/passwd: ascii text               - текст в коде ASCII;
/usr/include/stdio.h:C program text-  текст C-программы.
```

Копирование файлов

При работе с системой UNIX может появиться необходимость сделать копию файла. Команда `cp` полностью копирует содержимое одного файла в другой. Также она позволяет скопировать один или более файлов из одного справочника в другой, оставив оригинал файла на прежнем месте. Синтаксис команды.

`cp старый_файл новый_файл`

или

`cp файл ... оглавление`

Содержимое `старый_файл` копируется в `новый_файл`. Если `новый_файл` уже существует, его полномочия и владелец не изменяются. В противном случае полномочиями `новый_файл` становятся полномочия файла `старый_файл`.

Команда `cp` во второй форме копирует один или несколько файлов в заданное оглавление, сохраняя их исходные имена. Подготовленные к копированию файлы должны быть перечислены через пробел.

Команда `cp` отказывается копировать файл в себя.

Например, командой

```
$ cp /usr/src/README /usr/home/student
```

будет произведено копирование файла `README` в домашний справочник пользователя. А с помощью команды

```
$ cp /usr/src/README /usr/home/student/README.new
```

будет произведено копирование файла `README` в новый файл `README.new`, находящийся в домашнем справочнике пользователя.

В качестве самостоятельной работы произведите копирование файлов `/usr/home/student/README` и `/usr/home/student/README.new` в созданный Вами справочник `/usr/home/student/temp`.

При копировании и других операция, связанных групповыми операциями над файлами и справочниками, удобно для задания групп имен файлов использовать метасимволы. Рассмотрим принципы работы с метасимволами подробнее.

Использование метасимволов

Метасимволы служат для подстановки любых строк и символов в именах файлов:

- `*` - представляет произвольную строку (возможно, пустую);
- `?` - любой одиночный знак;
- `[C1 - C2]` - любая литера из диапазона `C1` - `C2` (в стандарте `ASCII`). Пара символов, разделенных знаком `-`, означает любой символ, который находится между ними, включая и их самих. Если первым символом после "[" идет "!", то указанные символы не должны входить в имя файла.

Например:

```
$ cat f*          выдаст все файлы каталога,
                  начинающиеся с "f";
$ cat *f*         выдаст все файлы, содержащие "f";
$ cat program.?   выдаст файлы данного каталога с
                  однобуквенными расширениями, скажем
                  "program.c" и "program.o", но не
                  выдаст "program.com";
$ cat [a-d]*      выдаст файлы, которые начинаются с
                  "a", "b", "c", "d". Аналогичный эффект
                  дадут и команды "cat [abcd]*" и
                  "cat [bdac]*".
```

Перенос и переименование файлов

Команда `mv` позволяет пользователю переименовать файл в том же справочнике или переместить его из одного справочника в другой. Когда перемещаете файл в другой справочник, то можете переименовать его или оставить прежнее имя. Синтаксис команды:

```
mv старый_файл новый_файл
```

или

mv файл ... оглавление

Команда mv пересылает (переименовывает) старый_файл в новый_файл. Если новый_файл уже существует, он предварительно исключается. Если право записи в файл2 отсутствует, mv выводит полномочия этого файла и спрашивает подтверждение переноса. Например, скопируем файл /usr/home/student/README.new в файл /usr/home/student/README.old, изменим права доступа файла README.new и переместим README.old обратно:

```
$ cp README.new README.old
$ chmod 444 README.new        - только чтение
$ mv README.old README.new
README.new: mode 444? y
```

Команда при подтверждении перемещения считывает одну строку из стандартного файла ввода. Если эта строка начинается с буквы y, то пересылка (переименование) выполняется; иначе mv заканчивает работу.

Команда mv во второй форме пересылает один или несколько файлов в заданное оглавление, сохраняя их имена. Некоторые версии этой команды позволяют переименовывать оглавления.

Команда mv отказывается пересылать файл в себя.

В качестве самостоятельной работы произведите перемещение файлов README и README.new из справочника /usr/home/student/temp в созданный Вами справочник /usr/home/student/temp1.

Удаление файлов и справочников

Удаление ненужных справочников и файлов возможно с помощью двух команд: rmdir (для удаления пустых справочников); rm (для удаления файлов и справочников).

Уничтожение пустого справочника

Для удаления пустых справочников используется команда rmdir. Синтаксис команды:

rmdir имя(имена) справочника(ов)

Для удаления нескольких справочников необходимо указать их имена в командной строке.

Командой rmdir нельзя удалить справочник, если пользователь не является его владельцем или он не пустой. Если одному пользователю необходимо удалить файл из справочника другого пользователя, то владелец должен дать право на запись для родительского справочника этого файла.

Если попытаться удалить справочник, в котором содержатся подсправочники и/или файлы, то команда rmdir напечатает сообщение:

имя-справочника not empty

Например, если попробовать удалить справочник /usr/home/student/temp1, который содержит два файла README и README.new, то на терминале появится сообщение:

```
$ rmdir temp1
```

```
rmdir: temp1 not empty
```

Таким образом, сначала необходимо удалить все содержимое справочника, а затем удалять его. Так справочник /usr/home/student/temp уже пуст, поэтому команда:

```
$ rmdir temp
```

удалит указанный справочник.

Удаление файлов

Команда rm исключает из справочника одну или несколько ссылок на файлы и удаляет файлы. Синтаксис команды:

```
rm [ опция ] файл ...
```

Если исключаемая ссылка была последней ссылкой на файл, файл уничтожается. Для исключения файла из справочника необходимо иметь право на запись в этот справочник, но не требуется права ни на чтение, ни на запись в сам файл.

Если право на запись в файл отсутствует, а стандартным файлом ввода является терминал, на него выводятся полномочия файла и считывается одна строка. Если эта строка начинается с символа y, файл уничтожается, иначе – сохраняется. Если задана опция -f, вопросы не задаются и сообщения не выводятся.

Если указанный файл является справочником, то кроме случая, когда задана опция -r, выводится сообщение об ошибке. Если задана опция -r, rm рекурсивно уничтожает содержимое всего указанного справочника и его самого.

Так, например, команда

```
$ rm -rf /usr/home/student/temp1
```

уничтожит все файлы, входящие в справочник temp1, а затем удалит и сам справочник, не выдавая на терминал ни каких сообщений.

Если задана опция -i, rm спрашивает про каждый файл требуется ли его исключить, и про каждое оглавление (если задана опция -r), требуется ли его просматривать. Например:

```
$ rm -i file1 file2
file1 : n                  (no - нет)
file2 : y                  (yes - да)
```

Пустая опция (знак минус) указывает, что все следующие параметры следует считать именами файлов. Это позволяет задавать имена файлов, начинающиеся со знака минус.

Владелец файла и защита файла

Каждый файл и каталог имеют владельца - обычно это пользователь, создавший их в первый раз. Владелец может затем назначить защиту файла со стороны трех классов пользователей:

- владелец (сам);
- группа - пользователи этой же группы, где владелец;
- остальные - все, имеющие доступ к системе.

Каждый файл имеет 3 вида разрешения на доступ:

- чтение (r) - можно читать (смотреть) содержимое файла или справочника (читать с ключом -l в ls);
- запись (w) - можно менять содержимое файла или справочника (создавать или удалять файлы в справочнике);
- выполнять (x) - использовать файл как команду UNIX и искать в справочнике.

Все комбинации трех видов разрешения доступа для трех классов пользователей (9 комбинаций) записываются в формате:

(если все права есть)

```
  rwx                 rwx                 rwx      или 777
   ¦                   ¦                   ¦
Владелец               ¦                   ¦
                     Группа                ¦
                                        Остальные
```

Отсутствие права доступа указывается минусом:

```
r--r--r-      или 444
```

Пример:

```
$ ls -l /bin
-r-xr-xr-x     1 bin  2005  Nov.26    12:00 ar
  ...
```

Эта команда показывает режимы доступа.

Установка и изменение режима доступа к файлу

Изменить существующие права можно с помощью команды chmod. Синтаксис команды chmod для установки режима:

```
chmod <режим> <файлы>
```

Пример:

```
$ chmod 644 f1 f2 f3
```

где 644 соответствует rw-r--r--. А сама команда устанавливает указанные права для трех файлов f1 f2 f3.

Также для изменения прав доступа к файлам и оглавлениям можно воспользоваться другой формой команды chmod. Синтаксис команды chmod для изменения режима:

```
chmod <изменения> <файлы>
```

В изменениях используются обозначения:

```
u – user          (установка для пользователя-владельца)
g – group         (установка для членов группы пользователя)
o – other         (установка для остальных пользователей)
a – all           (установка для всех групп)
r – read          (право на чтение)
w – write         (право на запись)
x – execute       (право на исполнение или поиск)
= - назначить     (назначить права вновь)
+ - добавить      (добавить новые права)
- - отнять        (исключить старые права)
```

Например, имеется какой-то справочник с тремя файлами f1 f2 f3:

```
$ ls -l
-r-------- ... f1
```

```
-r-------- ... f2
-r-------- ... f3
$ chmod  a = r, u + w  f1 f2 f3
```
или (эквивалентный вариант изменения прав доступа)
```
$ chmod  u = rw, go = r  f1 f2 f3
$ ls -l
-rw-r--r-- ... f1
-rw-r--r-- ... f2
-rw-r--r-- ... f3
$ chmod o-r f1 f2 f3
$ ls -l
-rw-r----- ... f1
-rw-r----- ... f2
-rw-r----- ... f3
```
Другие пользователи, не входящие в группу, потеряли право читать файлы.

Команды обработки информации

Подсчет строк, слов и символов

С помощью команды wc можно подсчитать число строк, слов и символов в указанном файле. Если указано более одного файла в командной строке, то программа wc осуществляет подсчет строк, слов и символов в каждом файле и затем выдает общее число. Словами считаются цепочки литер, разделенные литерами типа пробел, табуляция, новая строка. С помощью ключей указать либо подсчет только строк, или только слов, или символов.

Синтаксис команды:

wc [–lwc] [файл] ...

Если задана опция, начинающаяся с одной из букв lwc, то буквы l, w, c в ней задают подсчет строк, слов и литер соответственно. По умолчанию берется -lwc.

Система отвечает строкой в следующем формате:

l w c имя_файла

где l - число строк в файле; w - число слов в файле; c - число символов в файле.

Например, чтобы подсчитать число строк, слов и символов в файле /usr/home/student/temp/README, необходимо использовать команду:

```
$ wc README
      80     380    2749 README
```
Система отвечает, что в файле README 80 строк, 380 слов и 2749 символов.

Чтобы подсчитать число строк, слов и символов в нескольких файлах, используйте следующий формат:

wc файл1 файл2

Система отвечает следующим образом:

```
l       w       c       файл1
l       w       c       файл2
l       w       c       total
```

Число строк, слов и символов для файл1 и файл2 отображается на отдельных строках. На последней строке отображается общее число строк, слов и символов в двух файлах.

Например, подсчитаем число строк, слов и символов в файлах README и myfile1, который заранее создадим. Экран будет выглядеть следующим образом:

```
$ cal > myfile1
$ wc README myfile1
      80     380    2749 README
       8      39     145 myfile1
      88     419    2894 total
$
```

Поиск различий в файлах

Команда diff обнаруживает и сообщает обо всех различиях между двумя файлами и указывает, как изменить первый файл, чтобы он был дубликатом второго.

Синтаксис команды:

```
diff [ опция ] файл1 файл2
```

Команда diff выводит те строки файлов, которые нужно изменить для того, чтобы привести файлы в соответствие друг с другом. За исключением редких случаев, diff находит минимальный достаточный набор различий. Если файл1 или файл2 задан знаком минус, то используется файл стандартного ввода. В стандартном формате выводимые строки имеют вид

```
n1 a n3,n4
n1,n2 d n3
n1,n2 c n3,n4
```

Номера после буквы относятся к файлу2. Буквы означают: a – добавить строки/символы; d – удалить; c – заменить. Заменяя a на d, и читая наоборот (справа налево), можно также выяснить, как преобразовывать файл2 в файл1. Вслед за каждой такой строкой идут все затрагиваемые заданным в ней действием строки из первого файла с пометкой <, а затем — все затрагиваемые данным действием строки из второго файла с пометкой >.

Опции, доступные при выполнении команды diff, представлены в таблице 3.

Таблица 3 Опции команды diff

Опция	Функция
-e	Сгенерировать для редактора ed командный файл с запросами a, end, который будет воссоздавать файл2 из файла1.
-h	Выполнить задание быстрым, но не очень надежным методом. Этот метод годится, когда файлы имеют неограниченную длину, а отличия в них невелики и хорошо различаются.
-b	Проигнорировать пробелы и табуляции в конце строк; остальные цепочки пробелов и табуляций считать одинаковыми.

Примечание: за исключением опции -b, которую можно задавать вместе с любой другой опцией, опции команды diff взаимоисключают друг друга.

Рассмотрим применение команды diff на примере. Для чего произведем копирование файла README в файлы README1 и README2, отредактируем их с помощью текстового редактора ee, удалив некоторые строки, и сравним эти файлы:

```
$ cp /usr/src/README /usr/home/student/temp/README1
$ cp /usr/src/README /usr/home/student/temp/README2
$ cd /usr/home/student/temp
$ ee README1                    - удаляем некоторые строки
$ ee README2                    - удаляем некоторые строки
$ diff README1 README2
2a3
> $FreeBSD: src/README,v 1.22 2003/03/08 10:01:26 markm Exp $
36,37d36
< Source Roadmap:
< ---------------
79c78
<                     http://www.freebsd.org/doc/en_US.ISO8859-
1/books/handbook/synching.html
---
>
$
```

Поясним полученный вывод команды подробнее. В первой строке (2a3) указано, что вторая строка файла README1 и третья строка файла README2 отличаются; в следующей строке вывода команды указано, какую строку текста надо добавить в файл README1, чтобы файлы были одинаковыми (направление изменения указано символом >, а характер изменения - символом a в предыдущей строке вывода команды). Далее, три строки вывода команды, начиная с «36,37d36...», указывает: чтобы рассматриваемые файлы стали одинаковыми, надо либо удалить две указанные строки текста из файла README1, либо добавить в их же в файл README2, в указанные позиции (строки под номером 36 и 37). И последняя группа указаний, начинающаяся с строки «79c78», показывает, что существует различие в 78 строке файла README2 и в строке 79 файла README1, и указывает какие именно отличия.

Поиск в файле по шаблону

Пользователь может произвести поиск в файле указанного слова, фразы, группы символов с помощью команды grep. Поиск осуществляется по шаблону. Синтаксис команды:

grep [опции] шаблон файл(ы)

Команда grep сканирует заданные входные файлы (по умолчанию — файл стандартного ввода) в поисках строк, соответствующих некоторому образцу. Каждая найденная строка, как правило, копируется в файл стандартного вывода (то есть выводится на терминал). Образцы, используемые командой grep, являются ограниченными регулярными выражениями. Наиболее часто используемые опции приведены в таблице 4.

Таблица 4 Опции команды grep

Опция	Функция
-c	Напечатать только общее число найденных строк.
-e шаблон	То же самое, что и просто параметр шаблон; применяется, если выражение начинается со знака минус.
-i	При выполнении сравнений игнорировать размер букв. В некоторых версиях системы UNIX эта опция называется -y.
-l	Напечатать имена файлов, содержащих строки, сопоставляющиеся с образцом.
-n	Перед каждой строкой выводить ее номер в данном файле.
-s	Выполнить команду «молча». Никаких сообщений не выводится (кроме сообщений об ошибках).
-v	Напечатать все строки, которые не сопоставляются с данным образцом.

Если входных файлов несколько, то во всех случаях перед выводимыми строками выводится имя соответствующего файла. Следует соблюдать осторожность при использовании в регулярном выражении литер $ * ? [^ | ( ) и пробел, так как они имеют особый смысл и в языке-оболочке. Самое надежное – заключить весь параметр в одиночные кавычки (апострофы).

Например, команда

```
$ grep –n FreeBSD README
```

выведет на экран все строки, содержащие слово FreeBSD в файле README:

```
1:This is the top level of the FreeBSD source directory.  This file
3:$FreeBSD: src/README,v 1.22 2003/03/08 10:01:26 markm Exp $
11:building components (or all) of the FreeBSD source tree, the most
13:everything in the FreeBSD system from the source tree except the
78:the FreeBSD Project's development branches, please see:
```

А команда

```
$ grep FreeBSD README > myfile
```

сохранит те же строки в новый файл myfile, но уже без указания номеров строк.

Если же шаблон поиска будет содержать, например, знак пробела, то весь шаблон надо заключить в одинарные кавычки:

```
$ grep –c ‘source tree’ README
3
```

в данном примере на экран будет выведено число строк (в данном случае 3), в которых входит шаблон поиска.

Сортировка и слияние файлов

Система обеспечивает эффективное средство для сортировки и слияния файлов с помощью команды sort. Синтаксис команды:

```
sort [ опции ] [ +поз1 [ –поз2 ]]...[ опции ] [ имя ] ...
```

Команда sort считывает поименованные файлы, сортирует (упорядочивает) их и помещает результат в файл стандартного вывода. Имя — обо-

значает файл стандартного ввода. Если не задано ни одного имени входного файла, то сортируется файл стандартного ввода.

Подразумеваемым по умолчанию ключом сортировки является вся строка; подразумеваемым порядком сортировки является лексикографический порядок:

- строки, начинающиеся с цифры, будут отсортированы по цифрам и перечислены после строк, начинающихся с буквы;
- строки, начинающиеся с большой буквы, перечисляются до строк, начинающихся с маленькой буквы;
- строки, начинающиеся с таких символов, как "%", "*" сортируются на основе символьного представления ASCII.

Часто используемые опции, воздействующие на процесс сортировки глобальным образом, представлены в таблице 5. Можно задавать одну или несколько этих опций.

Таблица 5 Опции команды sort

Опция	Функция
-b	Игнорировать начальные пробелы и табуляции при сравнении полей.
-f	Преобразовывать при сравнениях прописные буквы в строчные.
-r	Изменить порядок сортировки на противоположный.
-t символ	Использовать в качестве разделителя полей (табуляции) символ
-m	Только слияние, входные файлы уже упорядочены.
-o имя	Следующий параметр является именем выходного файла, который будет использоваться вместо файла стандартного вывода. Этот файл может совпадать с одним из входных файлов.
-T имя	Следующий параметр является именем оглавления, в котором следует создавать временные файлы.

Конструкция +поз1 -поз2 делает ключом сортировки поле, начинающееся с позиции поз1 и оканчивающееся непосредственно перед позицией поз2. Оба параметра поз1 и поз2 имеют вид m.n; далее может следовать один или несколько флагов. Число m задает количество полей, которое нужно пропустить от начала строки, а n задает количество литер, которое требуется пропустить в дополнение к этому. Опущенное n означает 0; опущенное — поз2 означает конец строки.

Когда имеется несколько ключей сортировки, последующие ключи сравниваются, только если все предшествующие ключи равны. Строки, которые при выполнении операции сравнения оказались одинаковыми, упорядочиваются по всем значимым байтам.

Рассмотрим применение команды sort на практике. Допустим, имеется два файла group1 и group2, каждый из которых содержит перечень имен. Вы хотите отсортировать каждый список по алфавиту и затем объединить два списка в один. Вначале отобразим содержание файлов, выполнив команду cat для каждого файла. Экран будет выглядеть следующим образом:

```
$ cat group1
Фамилия Имя     Отчество
Петрова Ольга   Ивановна
Васина  Ирина   Васильевна
Антонов Иван    Михайлович
Сидоров Антон   Петрович
Иванов  Николай Федорович
$ cat group2
Фамилия Имя     Отчество
Орлов   Петр    Григорьевич
Яковлев Роман   Васильевич
Жданов  Олег    Алексеевич
Рыкова  Жанна   Николаевна
$
```

Теперь отсортируем и склеим эти два файла, выполнив команду `sort`:

```
$ sort group1 group2
```

Однако, результат выполнения этой команды будет только распечатан на экране терминала. Для того, чтобы отсортированный список был сохранен в файл для дальнейшего его использования, необходимо использовать опцию `-o`:

```
$ sort -o group12 group1 group2
$ cat group12
Фамилия Имя     Отчество
Фамилия Имя     Отчество
Антонов Иван    Михайлович
Васина  Ирина   Васильевна
Жданов  Олег    Алексеевич
Иванов  Николай Федорович
Орлов   Петр    Григорьевич
Петрова Ольга   Ивановна
Рыкова  Жанна   Николаевна
Сидоров Антон   Петрович
Яковлев Роман   Васильевич
$
```

В данном случае отсортированный список будет сохранен в файл с именем `group12`.

В случае если необходимо произвести сортировку, например, по имени, то команда будет выглядеть следующим образом:

```
$ grep –v ‘Фамилия’ group1 > group1.new        –избавимся от строки «Фамилия Имя     Отчество»
$ grep –v ‘Фамилия’ group2 > group2.new        –избавимся от строки «Фамилия Имя     Отчество»
$ echo ‘Фамилия Имя     Отчество’ > group12    –добавим соответствующую строку в результирующий файл
$ sort +1 group1.new group2.new >> group12     –сортируем по имени (по второму полю)
$ rm group1.new group2.new                     –удаляем временные файлы
$ cat group12
```

```
Фамилия Имя      Отчество
Сидоров Антон    Петрович
Рыкова  Жанна    Николаевна
Антонов Иван     Михайлович
Васина  Ирина    Васильевна
Иванов  Николай Федорович
Жданов  Олег     Алексеевич
Петрова Ольга    Ивановна
Орлов   Петр     Григорьевич
Яковлев Роман    Васильевич
```

Поиск и обработка файлов и каталогов

Поиск файлов и каталогов, удовлетворяющих определенным критериям, выполняется стандартными средствами операционной системы. Обычно для этих целей используется команда `find` с соответствующими опциями. Это довольно мощное средство, но имеет и свои ограничения. Синтаксис команды:

`find` *список-составных-имен выражение*

Команда `find` для каждого составного имени, заданного в *списке-составных-имен* (обычно, указывается справочник, из которого будет начата процедура поиска), рекурсивно обходит определяемую этим именем иерархию оглавлений в поисках файлов, удовлетворяющих заданному логическому выражению. Логическое выражение строится из первичных выражений, описания которых приводятся ниже (таблица 6). В этих описаниях параметр n используется для указания целого десятичного числа, причем +n означает «больше чем n», -n означает «меньше чем n», а =n означает «в точности n».

Таблица 6 Опции команды `find`

Опция	Функция
`-name` *шаблон*	Истина, если имя текущего файла совпадает с *шаблоном* поиска. При использовании метасимволов необходимо маскировать шаблоны.
`-type` *тип*	Истинна, если файл имеет тип, указанный параметром *тип*. Типы файлов задаются символами `b`, `c`, `d`, `f`, `l`, `p` и `s`, обозначающими специальное блочное устройство, специальное символьное устройство, каталог, обычный файл, символическую ссылку, именованный канал и сокет.
`-user` *имя*	Истина, если владельцем файла является пользователь *имя*, где *имя* — либо регистрационное имя пользователя, либо его числовой идентификатор.
`-group` *группа*	Истина, если файл принадлежит группе *группа*, где *группа* – либо имя группы, либо ее числовой идентификатор.
`-perm [-]` *права*	Если дефис не задан, то условие истинно, только если права доступа в точности соответствуют указанным (как в команде `chmod`). Если задан дефис, то условие истинно,

Опция	Функция
	если в правах доступа файла, как минимум, установлены те же биты, что и в указанных правах.
-links n	Истина, если на файл имеется n ссылок.
-size n [c]	Истина, если файл занимает n 512-литерных блоков или символов (если указан суффикс c). Перед размером можно указывать префикс + (не меньше), - (не больше) или = (в точности равен)
-atime n	Истина, если к файлу были обращения в течение последних n дней.
-mtime n	Истина, если файл изменился в течение последних n дней.
-ctime n	Истина, если файл был создан в течение последних n дней.
-newer файл	Истина, если файл – более новый, чем указанный в параметре файл.
-ls	Условие истинно всегда (выдает информацию о файле, аналогичную длинному листингу).
-print	Условие истинно всегда (выдает полное имя-файла в стандартный выходной поток)
-exec команда {} \;	Истина, если выполненная команда имеет код возврата 0. Команда заканчивается замаскированной точкой с запятой. В команде можно использовать конструкцию {}, заменяемую полным именем рассматриваемого файла.
-ok команда {} \;	Аналогично exec, но полученная после подстановки имени файла вместо {} команда выдается с вопросительным знаком и выполняется, если пользователь ввел символ y.
-print	Истина всегда; вызывает печать составного имени текущего файла.

В различных версиях ОС UNIX могут поддерживаться и другие компоненты выражений в команде find. Если командная строка сформирована неправильно, команда немедленно завершает работу.

Рассмотренные в таблице 6 первичные выражения (p) можно объединять при помощи следующих операций (в порядке уменьшения приоритета):

- (...) - Заключенная в скобки группа первичных выражений, связанных операциями (скобки имеют специальное значение для программы-оболочки, которое должно быть отменено).
- !p - Знак ! представляет собой унарную операцию отрицания.
- p p - Конкатенация первичных выражений. Если первичные выражения располагаются друг за другом, подразумевается, что они связаны логической операцией «и».
- p -o p - Альтернативные первичные выражения. Операция -o означает логическое «или».

Рассмотрим несколько примеров применения команды find.

Вначале – простой пример поиска файлов в каком-либо каталоге и вывод одержимого на консоль. Например, чтобы получить содержимое рабочего

каталога программы (в нашем примере это /usr/home/student/temp), достаточно выполнить команду

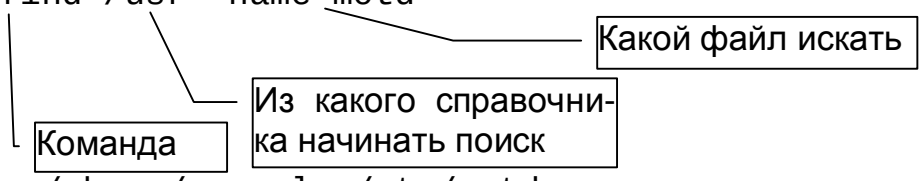
```
$ find `pwd` -print
```

На экране дисплея будет отображено содержимое текущего каталога:

```
/usr/home/student/temp
/usr/home/student/temp/README.new
/usr/home/student/temp/README
/usr/home/student/temp/myfile
/usr/home/student/temp/group1
/usr/home/student/temp/group2
/usr/home/student/temp/group12
/usr/home/student/temp/temp1
/usr/home/student/temp/group.new
```

С помощью следующей команды на консоли будут перечислены все файлы с именем motd, присутствующие в справочниках ниже по иерархии справочника /usr.

```
$ find /usr -name motd
```




```
/usr/share/examples/etc/motd
/usr/src/etc/motd
/usr/src/etc/rc.d/motd
/usr/src/release/picobsd/mfs_tree/etc/motd
$
```

В следующем примере с помощью команды find в указанном каталоге выполняется поиск файлов начинающихся с temp или READ:

```
$ find /usr/home/student/temp \( -name 'temp*' -o -name 'READ*' \)
/usr/home/student/temp
/usr/home/student/temp/README.new
/usr/home/student/temp/README
/usr/home/student/temp/temp1
$
```

Еще одной, часто используемой опцией является -size. Следующий пример демонстрирует поиск в каталоге usr/home/student/temp файлов, размер которых превышает 200 байтов, и файлов, размер которых меньше 200 байтов. Однако, сначала покажем какие файлы находятся в указанном каталоге:

```
$ ls -l
total 20
-rw-r--r--  1 student  wheel  2749 Apr 26 12:07 README
-rw-r--r--  1 student  wheel  2749 Apr 26 10:20 README.new
-rw-r--r--  1 student  wheel   119 Apr 26 16:34 group.new
-rw-r--r--  1 student  wheel   140 Apr 26 15:56 group1
-rw-r--r--  1 student  wheel   235 Apr 26 17:00 group12
-rw-r--r--  1 student  wheel   116 Apr 26 15:58 group2
```

```
-rw-r--r--  1 student  wheel   313 Apr 26 12:13 myfile
-rw-r--r--  1 student  wheel   203 Apr 26 17:00 temp1
$ find usr/home/student/temp -size +200c
/usr/home/student/temp
/usr/home/student/temp/README.new
/usr/home/student/temp/README
/usr/home/student/temp/myfile
/usr/home/student/temp/group12
/usr/home/student/temp/temp1
$ find usr/home/student/temp -size -200c
/usr/home/student/temp/group1
/usr/home/student/temp/group2
/usr/home/student/temp/group.new
$
```

Команда find обладает еще одной возможностью — выполнять команду или группу команд, передавая в качестве параметра результат поиска файлов. Рассмотрим следующий пример. Пусть в каталоге usr/home/student/temp необходимо удалить все файлы, имя которых начинается с temp. Для выполнения команды служит опция -exec. Команда должна заканчиваться пробелом и символами \;.

```
$ ls                          –узнаем, какие файлы есть
README          group.new       group12         myfile
README.new      group1          group2          temp1
$ find usr/home/student/temp -name 'temp*' -print -exec rm {} \;
/usr/home/student/temp
rm: /usr/home/student/temp: is a directory
/usr/home/student/temp/temp1
$ ls                          –узнаем, какие файлы остались
README          group.new       group12         myfile
README.new      group1          group2
$
```

Рассмотрим еще один пример с использованием опции –exec:

```
$ find `pwd` -name 'group*' -exec ls -l {} \;
-rw-r--r--     1   student     wheel     140   Apr   26   15:56
/usr/home/student/temp/group1
-rw-r--r--     1   student     wheel     116   Apr   26   15:58
/usr/home/student/temp/group2
-rw-r--r--     1   student     wheel     235   Apr   26   17:00
/usr/home/student/temp/group12
-rw-r--r--     1   student     wheel     119   Apr   26   16:34
/usr/home/student/temp/group.new
$
```

В данном примере на консоль были выведены атрибуты всех файлов, удовлетворяющих шаблону поиска.

Еще одной часто используемой опцией является опция –user, с помощью которой можно искать файлы определенных пользователей. Например, следующая команда удалит все файлы пользователя student1:

```
$ find / -user student1 –exec rm {} \;
```

Управление дисковым пространством и процессами

Управление дисковым пространством

Очень важной задачей для пользователя и системного администратора является контроль за используемым дисковым пространством. Даже при больших емкостях дисковых накопителей рано или поздно наступает момент, когда места на дисках не хватает. Современные программы потребляют, как правило, значительные ресурсы постоянной памяти, и сохраняется устойчивая тенденция к дальнейшему увеличению такого потребления. Даже небольшая система UNIX с малым числом пользователей порождает сотни файлов в ходе обычной работы. В процессе программирования можно создавать множество файлов для различных версий ваших программ. Поэтому каждый пользователь должен нести ответственность за расход дискового пространства.

Решение этой проблемы во многих случаях оказывается довольно простым — архивация и/или удаление неиспользуемых или редко используемых файлов и удаление "мусора". Для того чтобы отслеживать используемые ресурсы файловой системы, в операционных системах UNIX предусмотрен целый ряд команд. Контроль за использованием ресурсов жестких дисков можно выполнить с помощью системных команд `du` и `df`.

Справка об использовании дискового пространства

В системе UNIX существует команда, выдающая отчет об использовании дискового пространства заданными файлами, а также каждым справочником иерархии подсправочников каждого указанного справочника. Здесь под использованным дисковым пространством понимается пространство, занятое всей иерархией подсправочников указанного справочника. Синтаксис команды:

```
du [ опции ] [ имя ... ]
```

Команда `du` для каждого заданного имени файла или справочника выдает количество блоков, занимаемых на диске этим файлом или всеми файлами и (рекурсивно) всеми подсправочниками данного справочника. Если имена файлов не заданы, выдается информация для файлов из текущего справочника.

Команда `du` не показывает фактический размер файла в байтах, а отображает только количество выделенных блоков или байтов на диске. Размер блока зависит от настроек операционной системы и может варьироваться в широком диапазоне. Часто используют блоки размером в 1 Кбайт. Минимальное дисковое пространство, выделяемое для файла и каталога, обычно составляет 4 блока или (при размере блока в 1 Кбайт) 4 Кбайт. Обычно такую группу блоков называют кластером. Если, например, для файла выделено 8 Кбайт дискового пространства, то говорят, что файл занимает 2 кластера. Размер кластера может также варьироваться, хотя обычно он принимается равным 4 блокам. Именно поэтому для файлов, имеющих разные размеры, может быть выделено одинаковое дисковое пространство. Например, для файлов с размерами 3 байта и 3 Кбайт будет выделено на жестком диске одинаковое пространство в 4 Кбайт, то есть 1 кластер.

Запушенная без аргументов, команда du выдает отчет о дисковом пространстве для текущего каталога.

Перейдем к рассмотрению опций команды du. Некоторые наиболее часто используемые опции приведены в таблице 7.

Таблица 7 Опции команды du

Опция	Функция
-s	Выдается только суммарный итог для каждого аргумента.
–a	Информация выдается для каждого встретившегося файла, а не только для справочников, причем производится рекурсивный обход всех справочников, заданных в командной строке.
–c	Выдается общий итог по всем аргументам с конкретизацией по подсправочникам после того, как все аргументы будут обработаны. Этот параметр может быть использован для выяснения суммарного использования дискового пространства для всего списка заданных файлов и справочников.
-I шаблон	При рекурсивном выполнении пропускает справочники или файлы, чьи имена совпадают с шаблоном. Это шаблон может быть любым файловым шаблоном.
-h	Добавляет букву размера, например M, для двоичного мегабайта к каждому размеру.
-k	Выдает размеры в бинарных килобайтах (1024 байт).
-m	Выдает размеры в бинарных мегабайтах (1048576 байт).

Рассмотрим некоторые практические аспекты применения команды du. Команда du без параметров показывает размер дискового пространства, занимаемого файлом или каталогом в целом:

```
$ pwd
/usr/home/student
$ du temp                  –узнаем размер справочника temp
2       temp/temp1         –размер подсправочника temp1 (2 блока)
22      temp               –размер справочника temp (22 блока)
$
```

Информация выдается в блоках, размер которых определяется установками операционной системы. В приведенном примере информация о занимаемом дисковом пространстве подразумевает размер блока в 1 Кбайт.

Команда du выдает информацию в виде

```
[размер_используемого_дискового_пространства]    [полное_имя_файла]
```

Если команда du задана с опцией –c, то помимо информации по отдельным каталогам выдается суммарная величина используемого дискового пространства. Это можно увидеть в следующем примере:

```
$ du -c temp
2       temp/temp1
22      temp
22      total
```

$

Для сравнения можно выполнить команду is, которая показывает фактический размер файлов в байтах:

```
[yury@yuryhost yury]$ Is -1 $HOME/TEST1 total 12
-rw-rw-r--     1 yury    yury            11 Dec 14 10:35 ltmp
drwxrwxr-x     2 yury    yury          4096 Dec 14 16:26 USR1
-rw-rw-r—      1 yury    yury           120 Dec 14 16:27 ztestl
[yury@yuryhost yury]$ Is -1 $HOME/TEST1/USR1
total 8
-rw-rw-r—      1 yury    yury            28 Dec 14 16:26 testul
-rw-rw-r—      1 yury    yury            57 Dec 14 16:26 testu2
```

Как видно из примера, суммарный размер файлов в каталоге SHOME/TEST равен 216 байтам, в то время как для этих же файлов выделено 20 Кбайт дискового пространства.

В приведенных примерах в качестве параметров команды du используются имена каталогов. В этих случаях информация отображается в целом для каталога. Для того чтобы просмотреть размер файлов, необходимо ввести полное путевое имя файла. Следующие примеры демонстрируют применение команды du для работы с файлами:

```
[yury@yuryhost yury]$ du $HOME/TESTl/ltmp 4
/home/yury/TESTl/ltmp [yury@yuryhost yury]$ du -b $HOME/TESTl/ltmp
4096    /home/yury/TESTl/ltmp
[yury@yuryhost yury]$ du -B 512 $HOME/TESTl/ltmp 8
/home/yury/TESTl/ltmp
```

Команда du, примененная с опцией —exclude, позволяет исключить из рассмотрения отдельные файлы:

```
[yury@yuryhost yury]$ du -b —exclude 't*' $HOME/TEST1
1096      /home/yury/TESTl/USRl
L6384     /home/yury/TESTl
```

З этом примере из результата выполнения команды исключены данные по файлам, удовлетворяющим шаблону t*. Два таких файла находятся в ката-юге $HOME/TEST1/USR1:

```
yury@yuryhost yury]$ Is -1 $HOME/TEST1/USR1 otal 8
rw-rw-r—   1 yury    yury         28 Dec 14 16:26 testul
rw-rw-r—    1 yury    yury           57 Dec 14 16:26 testu2
```

!тобы расширить возможности команды du, можно применять ее в команд-ых скриптах. Рассмотрим простой пример, разработанный с помощью ин-

терпретатора shell и показывающий, как можно проанализировать использование дискового пространства в нескольких каталогах. Исходный текст скрипта (назовем его du_l) показан далее:

```
echo Size in bytes of directories for x in $*
 do du -b $x
 done
```

Здесь выполняется обработка нескольких каталогов, имена которых указываются в командной строке. Например, если нужно получить информацию о размере дискового пространства, используемого каталогами SHOME/TESTl и $HOME/TEST2, необходимо выполнить команду:

```
[yury@yuryhost BASH]$ ./du_l $HOME/TEST1 $HOME/TEST2
Size in bytes of directories                • ■     '
12288   /home/yury/TESTl/USRl ' 32768
/home/yury/TESTl 16384  /home/yury/TEST2
```

Далее я хочу предложить читателям рассмотреть пример командного скрипта, который может оказаться полезным при настройке и диагностике файловой системы. В этом примере анализируется процент использования дискового пространства в определенном каталоге. Результатом работы скрипта

является отношение фактического размера файла к выделенному под него дисковому пространству. Анализ проводится для всех файлов каталога. Программа написана на языке Perl (назовем ее du_percent.pl). Исходный текст скрипта представлен далее:

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

Sworkdir = shift or die "Usage: du_percent.pi [dir] [file]"; print "Percent of use
disk space by files in Sworkdir:\n";

chdir $workdir; opendi r(DIR, $workdir); Sentry =
readdir(DIR); closedir(DIR);

foreach $entry (@entry) { . next if ($entry eq ".");
    next if ($entry eq "..");
    $size_real = (stat($entry))[7];
    open(DU, "du -b '$workdir/$entry'|cut -fl |");
    $size_du = <DU>;
    chomp($size_du);
    close(DU);

    $per_usage = sprintf("%.2f", $size_real/$size_du * 100); if (-d $entry) {
    print "Dir:\t"; } else {
    print "File:\t"; }
   print "$entry: space $size_du bytes: real $size_real bytes: % of usage $per_usage%\n";
   }                                ' •
```

Программа достаточно сложная, поэтому я сделаю некоторые пояснения. В качестве аргумента программы из командной строки берется имя каталога. Далее переходим в искомый каталог и получаем список файлов. Это выполняется в строках

```
opendir(DIR, $workdir); Sentry = readdir(DIR)■ '
closedir(DIR) ;
```

Имена файлов после выполнения данного фрагмента кода будут находиться в переменной @entry. После этого в цикле foreach получаем информацию по каждому файлу, имя которого помещается в переменную $entry (не путать с массивом @entry!). Фактический размер файла получаем с помощью функции stat и сохраняем его в переменной $size_reai:

```
$size_real = (stat(Sentry))[7];
```

Размер дискового пространства, занимаемого файлом, получаем с помощью команды du и сохраняем.его в переменной $size_du:

```
open(IO, "du -b '$workdir/$entry'|cut -fl | ") ; $size_du = <DU>;
chomp($size_du); close(DU);
```

ше остается найти процентное соотношение $size_reai и $size_du и ывести результат на консоль:

```
usage = sprintf("%.2f", $size_real/$size_du * 100);

. t "gentry: space $size_du bytes: real $size_real bytes: % of usage $per_usage%\n";
```

Результат выполнения программы du_percent.pl может выглядеть так:

```
[yury@yuryhost PERL]$ ./du_percent.pi $HOME/TEST1
percent of use disk space by files in /home/yury/TESTl:
File:   ztestl: space 4096 bytes: real 120 bytes: % of usage 2.93%
File:   ltmp: space 4096 bytes: real 11 bytes: % of usage  0.27%
File:   t7: space 4096 bytes: real 3 bytes: % of usage 0.07%
Dir:    USR1: space 12288 bytes: real 4096 bytes: % of usage  33.33%
```

```
File:   981: space 4096 bytes: real 3 bytes: % of usage 0.07%

[yurySyuryhost PERL]$ ./du_percent.pi $HOME/TEST1/USR1 Percent of use disk space by files in
/home/yury/TESTl/USRl: File:   testul: space 4096 bytes: real 28 bytes: % of usage  0.68%
File:   testu2: space 4096 bytes: real 57 bytes: % of usage 1.39% [yury@yuryhost PERL]$
```

Информация о свободном пространстве на диске

df файловая-система

Команда **df** печатает количество свободных блоков в указанной ФС, например в **/dev/rp0a.** Если ФС не задана, печатается размер свободного пространства во всех смонтированных ФС.

Единицей измерения пространства в команде **df** являются физические блоки ФС. В некоторых версиях системы UNIX физические блоки состоят из 1024 байт и вмещают два (логических) блока, используемых в качестве единицы измерения пространства в команде **du**(l) и в команде **ls**(l) с опцией **—s**

∨ Рассмотрим теперь, как можно использовать системную команду df. Она выдает отчет о доступном и использованном дисковом пространстве на файловых системах. Синтаксис команды может быть представлен как

df *[опции]' [файл]*

При запуске без аргументов df выдает отчет по доступному и использованному пространству для всех смонтированных файловых систем (всех типов). В противном случае df для каждого файла, указанного в командной строке, выдает отчет по файловой системе, которая его содержит. Если аргумент *Файл* является дисковым файлом устройства, содержащим смонтированную файловую систему, то df показывает доступное пространство этой файловой системы, а *не* той, где содержится файл устройства.

Если используется команда df в стандарте POSIX, то по умолчанию все размеры выдаются в блоках по 512 байтов, но если задана опция -к, то используются блоки размером по 1024 байта. Формат вывода не стандартизован, за исключением случая применения опции -р. Если файл является не обычным файлом, а каталогом или FIFO, то результат не определен. В этом стандарте используются опции [-кр] [—], Переменная POSIXLY_CORRECT влияет на выбор размера блока. Если она не установлена и переменная BLOCKS IZE имеет значение, начинающееся со слова "HUMAN", ТО программа ведет себя, как при указании опции -h (если при этом не заданы опции -к и -т). Значения переменных LANG, LC_ALL, LC_CTYPE И LC_MESSAGES учитываются обычным образом.

Если используемая команда поддерживает стандарт GNU, то по умолчанию все размеры выдаются в блоках по 1024 байта (если размер блока не задан опцией), за исключением ситуации, когда установлена переменная POSIXLY_CORRECT. В этом случае размер блока соответствует POSIX-версии этой команды. В этом стандарте используются следующие опции:

[-ahHiklmPv]

[-t *тип__файловой_системы]*

[-x *тип файловой_системы]*

[--block-size=pa3Afep]

[--print-type]

[—no-sync]

[--sync]

[—help]

[--version]

Приведем описание основных опций команды:

- -к — используется размер блока в 1024 байта вместо размера по умолчанию 512 байтов;
- -р — вывод осуществляется в шесть колонок с заголовком "Filesystem , N-blocks Used Available Capacity Mounted on". Размер блока устанавливается по умолчанию (512 байтов) или 1024 байта при задании опции -к;

□ ------- завершает список опций;

П -a, —all — включает в список файловых систем те, которые имеют размер в 0 блоков и которые по умолчанию опускаются. Такие файловые системы обычно являются псевдофайловыми системами специального назначения, например, для automounter. Кроме того, только если задана эта опция, будут показаны файловые системы типа ignore или auto, которые поддерживаются некоторыми операционными системами;

О —blpck-size=paЗMep — выдает размер в блоках, размер в байтах которых задан;

П -ь, —human-readable — добавляет к каждому размеру букву, например, м для мегабайт;

— _H; —si — делает то же, что и опция -ь, но использует официальные единицы измерения СИ (где для расчетов берется 1000 вместо 1024 и, таким образом, м — это 1 000 000, а не 1 048 576);

Q _1? --inodes — вместо информации о блоках выдается информация об использовании индексных узлов в файловой системе. Индексный дескриптор (inode) содержит информацию о файле: владелец, права доступа, временные штампы и местоположение на диске;

р -k, —kilobytes — при выводе устанавливает размер блока в 1024 байта; О -1, --local — выводит только данные о локальных файловых системах;

Q _m) --megabytes — при выводе устанавливает размер блока в двоичный мегабайт (1 048 576 байтов). Заметим, что четыре опции -h, -н, -k, -m носят взаимоисключающий характер, и фактически будет работать только последняя, указанная в командной строке. Например, если заданы одновременно опции —si и -т, то в результате при выводе размер блока будет установлен в мегабайт (1 048 576 байтов);

□ —no-sync — указывает не делать системный вызов sync перед получени ем данных об использовании дискового пространства. При этом df мо жет запуститься значительно быстрее, но на некоторых файловых систе мах (например, в SunOS) результаты могут быть слегка устаревшими. Этот флаг включен по умолчанию;

□ -p, —portability — использует формат вывода по стандарту POSIX. Он такой же, как формат по умолчанию, только информация о каждой фай ловой системе всегда выводится на одной строке; имя смонтированного устройства никогда не помещается на отдельную строку. Это приводит к тому, что если длина имени смонтированного устройства больше, чем 20 символов (например, для некоторых сетевых файловых систем), то ко лонки не выравниваются;

□ —sync — выполняется системный вызов sync перед получением данных об использовании дискового пространства. На некоторых файловых сис темах (например, в SunOS) выполнение данного системного вызова дает более достоверные результаты, но зато с этим флагом выполнение df происходит значительно медленнее, особенно, когда файловых систем много или же они чрезвычайно активно используются;

О -t *тип файловой системы,* —type—тид *файловой_системы* — **показывает** только файловые системы с указанным типом. Можно задать несколько типов файловых систем, если использовать несколько опций -t. По умолчанию никакие типы файловых систем не опускаются;

□ -т, --print-type — выдает тип для каждой файловой системы. Тип оп ределяется системно-зависимым способом, например, посредством чте ния файла /etc/mtab;

□ -x *тип_файловой_системъ1, --exc1и&e-Ьуре=тип__файмовой_системы* — определяет, что не нужно показывать файловые системы с заданным типом. Можно задать несколько типов файловых систем, если использовать несколько опций -х. По умолчанию никакие типы файловых систем не опускаются;

□ -v — игнорируется для совместимости с версией df из System V.

Рассмотрим несколько примеров использования команды df:

```
[yury@yuryhost    PERL]$ df -k
Filesystem            lK-blocks     Used Available Use% Mounted on
/dev/hda5               7091968       2004488  4727224  30% /
```

```
/dev/hda3                 101107       14832     81054   16% /boot
lone                      256900        0    256900   0% /dev/shm
```

З этой команде используется опция -x, поэтому вывод на консоль выполняется в пересчете на однокилобайтные блоки.

З следующем примере задается опция -a, поэтому на дисплей выводится :писок всех файловых систем, включающий специальные файловые системы:

```
yurygyuryhost PERL]$ df -a
'ilesystem          lK-blocks       Used Available  Use% Mounted on
dev/hda5               7091968    2004488    4727224       30% /
.one                         0          0          0       /proc
sbdevfs                      0          0          0   /proc/bus/usb
dev/hda3                101107      14832      81054     16% /boot
one                          0          0          0      /dev/pts
one                     256900          0     256900   0% /dev/shm ■
```

■истемные администраторы часто используют команду df для периодически ыполняемого контроля дискового пространства и предотвращения нехватки еста на диске. В качестве примера такого применения я приведу програм-у dfjwarning.pl, разработанную на языке Perl, выполняющую периодиче-сий контроль свободного дискового пространства и выдающую предупреж-*ние, если размер используемого пространства превышает заданные >аницы. В качестве первого параметра программа принимает числовое зна-:ние (в процентах) используемого дискового пространства, при превыше-яи которого выдается предупреждение. Вторым параметром является пери-*l* проверки дискового пространства. Программа реализована в виде демона выполняется без вмешательства администратора.

Исходный текст программы представлен далее: #!/usr/bin/perl -w

```
use POSIX 'setsid'; use POSIX 'strtod';

$limit = shift or die "Usage: df_warning [limit] [interval]";
$interval = shift or die "Usage: df_warning [limit] [interval]";
chdir "/" ;
$pid = fork;
if ($pid) { exit; }
setsid;
while (1) {
  open(DF, "df -k |"); @lines = <DF>; ,
  close(DF);
  foreach $lines (@lines) { $tmp = Slines; @sptmp = split(" ",
    $tmp); next unless ($sptmp[4] =~ /"\d+/); $num =
    strtod(substr($sptmp[4], 0, 2)); if ($num > $limit) {
   print "Usage of disk space on $sptmp[0] exceeded $limit% !\n"; } }
   sleep $interval; }
```

Сделаю некоторые пояснения, касающиеся исходного текста программы. Программа принимает в качестве первого аргумента величину предела использования дискового пространства ($limit), в качестве второго аргумента — интервал времени для опроса состояния дискового пространства ($intervai). Для выполнения анализа необходимо получить вывод программы df, что выполняется с помощью операторов:

```
Open(DF, "df -k |"); ©lines = <DF>;
Close(DF);
```

Переменная @lines после выполнения этих строк кода будет содержать вывод команды df. Далее необходимо проанализировать каждый элемент массива eiines, предварительно разбив его на поля с помощью следующей команды:

```
@sptmp = split(" ", $tmp);
```

Таким образом, массив @sptmp содержит в качестве элементов поля одной строки вывода команды df. Нас интересует поле с индексом 4, поскольку именно в нем находится информация об используемом дисковом пространстве. Чтобы легче было представить эту картину, напомню, как выглядит вывод команды df с опцией -к:

```
Filesystem            lK-blocks     Used Available Use% Mounted on
/dev/hda5               7091968        2004488  4727224  30% /
'dev/hda3                101107       14832    81054  16% /boot
lone                     256900        0   256900  0% /dev/shm
```

7оле с индексом 4 — это колонка "use%", которая нам нужна! При этом гужно пропустить первую строку

Filesystem lK-blocks Used Available Use% Mounted on"

оскольку она не несет никакой информации. Отсечь эту строку можно с омощью команды

```
ext unless ($sptmp[4] =~ /^\d+/);
```

ля всех последующих строк выполним преобразование их значений в чи-говой формат с помощью команды

lum = strtod(substr($sptmp[4], 0, 2));

этом случае, например, слово зо% будет преобразовано в число 30. В ос-льном же текст программы понятен и в дополнительных пояснениях не ждается.

:зультат работы программы-демона будет выглядеть так:

```
ury@yuryhost PERL]$ ./df_warning.pl 15 60 age of disk space on
/dev/hda5 exceeded 15% ! age of disk space on /dev/hda3 exceeded 15% !
```

эимеры

L этом рассмотрение возможностей команд du и df можно закончить. я успешного администрирования важно также контролировать фактиче-*ш* размер файлов, чтобы можно было вовремя заметить аномалии в ра-

боте как отдельных приложений, так и операционной системы. Значительное увеличение размеров файлов за короткое время при неизменном характере работы операционной системы или приложений должно насторожить системного администратора. Отслеживать подобные изменения в файловой системе можно с помощью простых командных скриптов, написанных на Perl.

Посмотрим, как можно контролировать размер файлов какого-либо каталога. Эта информация, как правило, должна находиться под рукой и периодически обновляться. Далее приведен исходный текст программного скрипта, выполняющего подсчет объема дискового пространства, занимаемого каталогом:

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


$workdir = shift or die "Usage: du.pl [dir] [file]"; Sworkfile = shift or die
"Usage: du.pl [dir] [file]";


open(DIR, "find '$workdir' -name '$workfile' -print |"); Sentry = <DIR>; close-
dir(DIR);


foreach $entry (Sentry) {
 chomp($entry);
 next if ($entry eq ".");
 next if ($entry eq "..");
```

```
 next if (-d $entry);
 $size += (stat($entry))[7]; } print "Total size of files in $workdir = $size
bytes\n";
```

Эта программа (назовем ее du.pl) в качестве параметров принимает имя каталога и шаблон имени файла. Результат работы программы для каталога $HOME/TEST1 приведен далее:

```
[yuryOyuryhost PERL]$ ./du.pl $HOME/TEST1 '*' Total size of files in
/home/yury/TESTl = 216 bytes
[yury@yuryhost PERL]$ ./du.pl $HOME/TEST1 'z*[1] Total size of files in
/home/yury/TESTl =12 0 bytes
[yury@yuryhost PERL]$
```

Для контроля просмотрим, какой объем дискового пространства занимают файлы этого каталога:

```
■ [yury@yuryhost PERL]$     Is -1 $HOME/TEST1
  total 12
  -rw-rw-r—    1 yury      yury          11 Dec 14 10:35 ltmp
  drwxrwxr-x   2 yury      yury        4096 Dec 14 16:26 USR1
  -rw-rw-r--    1 yury     yury         12 0 Dec 14 16:27 ztestl
  [yury@yuryhost PERL]$ Is -1 $HOME/TEST1/USR1
  total 8
  -rw-rw-r—   1 yury      yury          28 Dec 14 16:26 testul
  -rw-rw-r—  . 1 yury      yury          57 Dec 14 16:26 testu2
```

В этой программе получаем "чистый" результат — без учета дискового пространства, выделяемого под каталог!

Немного усложнив программу, можно получить объем дискового пространства, занимаемого файлами, находящимися в разных каталогах. Исходный текст командного скрипта (назовем его du_multidir.pl) приведен далее:

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

@arg = @ARGV; foreach $arg (@arg) { open(DIR, "find '$arg' -
print |"); ■ @entry = <DIR>; closedir(DIR); foreach $entry
(©entry) {     ■   ■
   chomp($entry);
   next if ($entry eq ".");
   next if (Sentry eq "..");
   next if (-d $entry);
   $size += (stat($entry))[7];                                    '
   }
 print "Total size of files in $arg = $size bytes\n"; $total += $size; $size
 = 0;
}                "                    •
 print "Total = $total bytes\n";
```

о результате работы скрипта с содержимым каталогов SHOME/TEST1, 0OME/TEST2 и SHOME/TEST3 получим:

```
   ry@yuryhost PERL]$ ./du_multidir_vpl $HOME/TESTl $HOME/TEST2 5HOME/TEST3
Total size of files in /home/yury/TESTl = 216 bytes Total size of files in
/home/yury/TEST2 = 30 bytes Total size of files in /home/yury/TEST3 = 23
bytes Total = 269 bytes
```

Полученный результат легко проверить:

```
[yury@yuryhost PERL]$ Is -1 $HOME/TEST3
total 8
```

```
_rw-rw-r—   1 yury    yury         12 Dec 14 10:36 3tmp
-rw-rw-r—   1 yury    yury            0 Dec 14 12:29 test3_zero
-rw-rw-r—   1 yury    yury         11 Dec 13 17:03 test4tmp [yury@yuryhost PERL]$
Is -1 $HOME/TEST2
total 8
-rw-rw-r—     1 yury    yury          11 Dec 13 17:12 2tmp
-rw-rw-r—     1 yury    yury          19 Dec 13 17:09 qtmp2
-rw-rw-r--    1 yury    yury        0 Dec 14 12:29 test2_zero
[yury@yuryhost PERL]$ Is -1 $HOME/TEST1
total 12
-rw-rw-r--    1 yury    yury          11 Dec 14 10:35 ltmp
drwxrwxr-x    2 yury    yury         4096 Dec 14 16:26 USR1
-rw-rw-r—     1 yury    yury       ■   120 Dec 14 16:27 ztestl
[yuryOyuryhost PERL]$ Is -1 $HOME/TEST1/USR1
total 8
-rw-rw-r—     1 yury    yury         28 Dec 14 16:26 testul
~r*-rw-r—     1 yury    yury         57 Dec 14 16:26 testu2
```

На основе предыдущего примера можно разработать программу-демон, выполняющую контроль размеров файлов в непрерывном режиме и сигнали-3иРующую в случае превышения определенной границы. Эта программа Может оказаться полезной при сопровождении отдельных приложений, инTeНсивно использующих дисковое пространство. Исходный текст програм-Mbt (назовем ее du_multidird.pl) представлен далее. Программа отслеживает cУММарный размер (в байтах) дискового пространства, занимаемого файла-И, Размещенными в нескольких каталогах. В случае превышения опреде-
ленного лимита, определяемого переменной $MAX_SIZE, на консоль выводится сообщение. Контроль дискового пространства выполняется каждые 30 секунд. Как и большинство примеров из этой книги, программа является демонстрационной, но может,, послужить шаблоном для собственных разработок читателей. Итак, вернемся к исходному тексту:

```
#!/usr/bin/perl -w use POSIX 'setsid';

@arg = @ARGV; $MAX_SIZE = 300;

:hdir "/";
$pid = fork;
if ($pid) { exit; .}
setsid;

/hile (1) {
foreach $arg (@arg) { open(FD, "find '$arg' -print |");
 ©entry = <FD>; close(FD);
 foreach $entry (Sentry) { chomp($entry); next if
   ($entry eq "."); next if ($entry eq ". . ") ; next if
   (-d Sentry); $size += (stat(Sentry))[7]; }
 Stotal += Ssize; $size = 0;

 if (Stotal > $MAX_SIZE) { print "Total: $total\n";
  print "Disk space occupied more than $MAX_SIZE bytes!\n";
 }
 $total = 0; sleep 30; }
```

Программный скрипт принимает в качестве аргументов имена контролируемых каталогов, количество которых может быть переменной величиной. Результат работы программы представлен далее (анализируется дисковое пространство, занимаемое файлами каталогов $HOME/TEST1, $HOME/TEST2 и $HOME/TEST3). В данном случае размер занимаемого дискового пространства превысил установленный лимит в 300 байтов, поэтому каждые 30 секунд на консоль будет выводиться сообщение:

```
[yury@yuryhost PERL]$ ./du_multidird.pl $HOME/TEST1 SHOME/TEST2 SHOME/TEST3
[yury@yuryhost PERL]$ Total: 313
Disk space occupied more than 300 bytes!
```

Вместо вывода сообщения на консоль можно сделать доработку программного скрипта, например, так, чтобы системному администратору отправлялось сообщение по электронной почте. Прервать работу демона можно посылкой ему сигнала kill, при этом нужно определить идентификатор процесса:

```
[yury@yuryhost PERL]$ ps -ef|grep du_multidird.pl
yury     4280     1  0 22:19 ?       00:00:00 /usr/bin/perl -w ./du_multidird.pl
/home/yury/TESTl /home/yury/TEST2 /home/yury/TEST3
yury     4288  4056  0 22:20 pts/0    00:00:00 grep du_multidird.pl [yury@yuryhost PERL]$
kill -s 9 4280
```

Список литературы

1. Дегтярев Е.К. Введение в Unix. М.: - 1991, 156 с.
2. Кристиан К. Операционная система UNIX. - М.: Финансы и статистика, 1985, 320 стр.
3. Баурн С. Операционная система UNIX. – М.: Мир, 1986, 464 стр.
4. Магда Ю.С. Администрирование Unix. – СПб.: БХВ-Петербург, 2005. – 800 с.

Составители:
Ляховец Михаил Васильевич
Огнев Сергей Петрович

ВВЕДЕНИЕ В UNIX

Учебно-методическое издание
по дисциплине «Операционные системы, среды, оболочки».
Специальности: 080801 – Прикладная информатика (в управлении);
230201 – Информационные системы и технологии

Печатается в полном соответствии с авторским оригиналом

Подписано в печать «___»_____ 20___г.
Формат бумаги_____. Бумага писчая. Печать офсетная.
Усл. печ. л. _____. Уч.-изд. л. _____. Тираж _____ экз. Заказ _____.

Сибирский государственный индустриальный университет
654007, г. Новокузнецк, ул. Кирова, 42.
Типография СибГИУ