1. ЛАБОРАТОРНАЯ РАБОТА № 1. ПРАКТИЧЕСКОЕ ЗНАКОМСТВО С ОПЕРАЦИОННОЙ СИСТЕМОЙ *UNIX*

ЦЕЛЬ РАБОТЫ

Ознакомиться с операционной системой *Unix*, получить практические навыки работы в наиболее распространенном командном интерпретаторе *bash*, изучить принципы организации файловой системы *Unix* и базовых команд управления файлами.

ЗАДАНИЕ

Осуществить в локальной сети с помощью программы *PuTTY* через протокол *ssh* доступ к удаленному компьютеру (необходимые данные для доступа указывает преподаватель) под управлением ОС *Linux* в консольном режиме. Ознакомиться с перечнем основных команд, используемых пользователями ОС *Linux* при работе в системе.

1.1. ОСНОВЫ РАБОТЫ В ОПЕРАЦИОННОЙ СИСТЕМЕ UNIX

1.1.1. Интерфейс командной строки в системах Unix

В *Unix*-подобных ОС базовый уровень общения с пользователем заключается во вводе с клавиатуры команд и просмотре выводимой текстовой информации на дисплее (такой способ общения часто называют «интерфейсом командной строки»). Понятия «клавиатура» и «дисплей» в данном случае во многом условны и не обязательно означают реальные устройства компьютера, на котором работает пользователь. Например, «дисплеем» может быть окно графической среды пользователя или интерфейс программы удаленного доступа, использующей протоколы *telnet* или *ssh*. Но при этом смысл от этого не изменяется: базовый интерфейс пользователя предполагает ввод команд и вывод текстовой информации. Для того, чтобы подчеркнуть «виртуальность» устройств ввода и вывода текста, их вместе называют терминалом.

Unix – многопользовательская ОС, следовательно, каждый компьютер под управлением этой ОС может иметь множество терминалов, что позволяет одновременно работать многим пользователям. Терминалы могут быть непосредственно подключены к компьютеру или существовать где-либо в сети (локальной или глобальной). Сетевые терминалы обычно представляют собой

компьютеры с собственной ОС, на которых запущена программа удаленного доступа, подобная стандартной программе *telnet*, работающей с использованием сетевых протоколов на основе *TCP*.

В настоящее время вместо *telnet* обычно используют программы, работающие по защищенному шифрованием протоколу *ssh*.

Интерфейс командной строки – не единственный способ общения с ОС *Unix* (например, существуют графические интерфейсы пользователя: *KDE*, *GNOME*, *Xfse* и т. д., различные файловые оболочки и т. п.), но именно в *Unix* умение работать с командной оболочкой очень важно. Во многом это обусловлено огромным набором базовых команд, их чрезвычайной гибкостью и возможностью совместного использования для автоматизации обработки данных.

Регистрация

Во всех *Unix*-подобных ОС, установленных на конкретном компьютере, имеется некоторая база данных пользователей, имеющих право использования ресурсов этого компьютера. Пользователей, не включенных в этот список, система к работе не допустит. База данных ведется администратором компьютера и содержит для каждого пользователя следующую информацию:

- регистрационное имя;
- зашифрованный пароль;
- идентификатор пользователя (*User ID – UID*);
- список групп, в которые включен пользователь;
- путь к командной оболочке;
- путь к домашнему каталогу;
- другая дополнительная информация.

Регистрационное имя и пароль необходимы для процедуры регистрации пользователя. Идентификатор *UID* востребован внутренними функциями системы и непосредственно используется редко. Механизм групп позволяет объединять пользователей по определенным полномочиям на доступ к файлам и программам. Путь к командной оболочке нужен для ее запуска после процедуры регистрации. И, наконец, домашний каталог – это обычно место в файловой системе, целиком принадлежащее данному пользователю и, конечно, администратору.

Начало работы

Чтобы начать работу с *Unix*, нужно получить доступ к терминалу и зарегистрироваться в системе. В случае удаленной работы подключение

терминала можно осуществить запуском программы поддерживающей протокол *ssh* и соединением с тем компьютером, на котором предполагается вести работу. Когда компьютер готов зарегистрировать пользователя, на экране отображается приглашение к вводу его имени:

```
login: _
```

В ответ на это приглашение нужно ввести регистрационное имя, согласованное с администратором или владельцем компьютера (понятно, что на собственном компьютере пользователь сам вправе выбирать имена пользователей). Имя пользователя рекомендуется составлять из строчных латинских букв и цифр. Некоторые имена (например, *root*, *ftp* и т. п.) могут быть зарезервированы для системных целей и не могут быть отданы обычному пользователю. Особый частный случай в большинстве систем – имя *root*, которое принадлежит администратору или владельцу компьютера. Это имя дает практически неограниченные права по управлению системой. Пользователя с правами *root* часто называют привилегированным.

После ввода имени и нажатия клавиши *Enter* («*Return*», «*Ret*», «*CR*») система выведет на экран запрос на ввод пароля. Например:

```
login: alex
password: _
```

Здесь и далее в аналогичных примерах вводимую пользователем информацию будем выделять другим шрифтом, что позволит отличать ее от выводимых системой символов.

Ввод пароля также завершается нажатием клавиши *Enter*. Вводимый пароль не отображается на экране. Если пользователь с указанным именем существует и его пароль введен правильно, система сделает домашний каталог пользователя текущим и запустит командную оболочку, связанную с данным пользователем. Оболочка обычно выполняет некоторый начальный набор команд, который может вывести приветственное сообщение, указать на наличие или отсутствие новой почты, выполнить начальный набор команд (все эти действия зависят от особенностей настройки конкретной ОС). И, наконец, на экране появится приглашение к вводу команды:

```
$ _
```

Приглашение не обязательно имеет такой вид, как показано выше, и зависит от конкретной командной оболочки и ее конфигурации.

Командные оболочки ОС *Unix*

В системах *Unix* используются различные командные оболочки (*command shells*), называемые также командными процессорами или интерпретаторами команд. Среди них наиболее известны и распространены:

- *sh* (*Bourne shell*) – оболочка Борна (испытана временем, но не слишком удобна в работе);
- *csh* (*C-shell*) – оболочка *C* (несколько более удобна по сравнению с sh, но несовместима с ней по командному языку);
- *ksh* (*Korn shell*) – оболочка Корна (включает мощный командный язык, основанный на языке *sh*, и развитые средства интерактивной работы);
- *bash* (*Bourne-Again Shell*) – «снова» оболочка «Борна» (удобна для интерактивной работы, создана на основе *sh* и во многом с ней совместима).

Тип оболочки, как правило, можно определить по последнему символу приглашения: знак доллара («*$*») указывает на *sh*-совместимую оболочку (*sh, bash, ksh*), а знак амперсанда («*&*») соответствует оболочке *csh*. Однако у привилегированного пользователя независимо от используемого командного процессора последним символом приглашения обычно бывает знак решетки («#»).

Основными функциями командных оболочек являются:

- организация диалога с пользователем (ввод команд);
- выполнение внутренних команд;
- запуск внешних программ;
- исполнение командных файлов.

Возможности командных языков в системе *Unix* являются гораздо более полными, чем в системе *MS-DOS*, и вполне могут быть названы полноценными языками программирования. Командные языки в разных оболочках различаются, а стандартным принято считать командный язык оболочки *bash*.

Команды *Unix* и запуск программ

Общий синтаксис команд в *Unix*-подобных ОС выглядит следующим образом:

```
имя_команды [ключи ...] [параметры ...]
```

Первый элемент обозначает конкретную команду, аргументы (ключи и параметры) могут сообщать дополнительную информацию. Ключи обычно начинаются со знака «минус». Например, команда

```
ls -l -a /home
```

состоит из:

- имени команды «*ls*», выводящей список файлов в заданном каталоге;
- ключа (модификатора) «*l*», указывающего, что нужно вывести подробный листинг;
- ключа «*a*», указывающего, что нужно выводить все файлы, включая служебные («дот файлы»);
- параметра «*/home*», задающего путь к каталогу.

В командах ОС *Unix*, их ключах и параметрах регистр букв (строчные или заглавные) различается. Для большей части команд характерна запись строчными буквами. Ключи во многих случаях могут объединяться в одну группу. Например, команда

```
ls -la /home
```

полностью эквивалентна рассмотренной выше.

Команды разделяются на внутренние, которые выполняются командным процессором, и внешние. Внутренних команд обычно немного, а их состав и синтаксис могут зависеть от используемой командной оболочки. При использовании *bash* полный список и краткий синтаксис внутренних команд можно получить, набрав после приглашения команду «*help*».

Внешние команды представляют собой запуск программ, независимых от оболочки. Для запуска программы простым указанием ее имени необходимо, чтобы путь к этой программе был указан в переменной среды *PATH* (аналог одноименной переменной среды в *MS-DOS*). Если программа не найдена в каталогах, перечисленных в *PATH*, перед именем программы должен быть явно указан путь, даже если программа находится в текущем каталоге (хотя в современных *Unix* системах это уже не требуется). Например, запуск программы *hello* из текущего каталога может выглядеть так:

```
$ ./hello
```

В этом примере знак доллара в начале строки представляет собой приглашение к вводу, формируемое системой, а остаток строки – информацию, введенную пользователем.

Пути в переменной среды *PATH* отделяются друг от друга знаками двоеточия без окаймляющих пробелов. Если вывести на экран листинги всех каталогов, входящих в *PATH*, можно таким образом получить полный список внешних команд системы, с которой осуществляется работа. Следует отметить, что значение любой переменной среды

можно получить, указав в требуемом контексте ее имя с предшествующим знаком доллара. Например, в команде

```
$ echo $PATH
```

выражение *$PATH* будет заменено командным интерпретатором на содержимое переменной *PATH*. Учитывая, что действие команды *echo* заключается в выводе в стандартный поток вывода своего аргумента, то на экран попадет именно содержимое переменной среды *PATH*.

Изменение пароля

Первая команда, которую следует выполнить при первом сеансе работы в системе – команда изменения собственного пароля: *passwd*. Эта команда вызывается без параметров. После ее запуска на экране появится приглашение ввести старый пароль (если пароля не было, этот шаг может быть пропущен). После правильного ввода старого пароля будет предложено ввести новый пароль, а затем ввести его еще раз для исключения случайной ошибки. Пароли при вводе отображаться не будут. Ниже представлен примерный протокол работы команды *passwd*:

```
$ passwd
Changing password for alex
Old password:
New password:
Re-type new password:
$ _
```

Для выбора паролей существуют определенные правила. Основное требование состоит в том, что пароль не должен быть угадываемым. Не надо писать свое имя в обратном порядке, не следует составлять пароль из одинаковых или ряда соседних на клавиатуре букв и т. п. Пароль должен содержать минимум шесть-семь символов и включать необычные сочетания букв, цифр, дефисов и подчеркиваний. В современных системах предпочтительно использование длинных паролей из 12-ти и более символов. Многие реализации команды *passwd* пытаются определить пригодность нового пароля и выводят предупреждающие сообщения, если пароль неудачен. Типичным случаем, когда *passwd* может проявить «недовольство» – ввод пароля только из строчных букв.

Не следует думать, что вышеуказанные правила чрезмерны. Даже личный домашний компьютер при входе в сеть *Internet* через модемное соединение может стать видимым другим людям, которые могут

попытаться поменять пароль или узнать пароль соединения с провайдером. Поэтому простой пароль может существенно упростить задачу злоумышленников и стоить «нерадивому» пользователю очень дорого.

Получение справочной информации

Системы *Unix*, как правило, поставляются с огромным количеством справочной информации в электронном виде. Справочная информация разбита на разделы по тематике. Нумерация разделов в разных системах может быть разной. Пожалуй, самая часто используемая информация содержится в разделе 1, где рассматриваются команды и прикладные программы, доступные рядовым пользователям системы. В пределах раздела справочные материалы организованы по так называемым «страницам» (*manual page*). Каждая такая страница содержит документацию по конкретной команде, функции, интерфейсу, протоколу и т. п. и в реальности может быть многостраничным документом. Для получения справочной информации можно использовать команду

```
man [раздел] [ключ]
```

Для получения справки по использованию команды или программы аргумент «ключ» должен быть именем соответствующей команды или программы. Параметр «раздел» может представлять собой цифру (или букву) номера раздела справочных руководств, в котором находится нужная страница документации. Отметим, что номер раздела указывать необязательно, т. к. при его отсутствии будет найден первый подходящий раздел, где встречена нужная тема. Чтобы получить справку об использовании самой команды *man*, проще всего ввести

```
$ man man
```

Существуют и другие полезные команды для работы со справочными руководствами. В частности, команда

```
apropos [ключ]
```

позволяет найти и вывести перечень тех страниц руководств, которые содержат в строке краткого пояснения заданное ключевое слово ключ. Справочная информация *man* доступна только для внешних команд. Для получения подсказки по внутренним командам оболочки необходимо использовать команду *help*, например

```
$ help cd
```

Простейшие команды для работы с файловой системой

Команда изменения текущего каталога:

cd [имя каталога]

Если команда *cd* вызвана без аргументов, текущим каталогом станет домашний каталог пользователя. Чтобы вывести на экран полное имя текущего каталога, нужно использовать команду *pwd* без аргументов. Команда

ls [имя_каталога]

позволяет получить листинг указанного каталога. Если имякаталога не указано, то будет выведен листинг текущего каталога. У команды *ls* есть несколько полезных ключей

l – вывести полную информацию о каждом файле;

a – вывести листинг всех файлов, включая такие, имена которых начинаются с символа точки.

Команды *mkdir* и *rmdir* позволяют, соответственно, создать или удалить указанный каталог

mkdir [имя каталога]

rmdir [имя каталога]

Команда просмотра файлов *less* позволяет просматривать файлы произвольного размера и перемещаться по их содержимому с помощью клавиш управления курсором (для выхода используется клавиша «*q*»)

less [имя файла]

Команда копирования файлов

cp [источник] [приемник]

Команда перемещения или переименования файлов

mv [источник] [приемник]

Команда удаления файлов

rm [имя файла]

С командами *cp* и *rm* может использоваться ключ «*r*», позволяющий копировать, перемещать или удалять каталоги со всем их содержимым рекурсивно.

Для полной информации о перечисленных командах, их аргументах и вариантах их использования можно обратиться к страницам руководства пользователя (команда *man*).

Стандартные потоки ввода-вывода

С каждой программой, запускаемой из командной строки *Unix*, связаны три стандартных потока данных:

- стандартный поток ввода (*stdin*);
- стандартный поток вывода (*stdout*);
- стандартный поток ошибок (*stderr*).

Программы, требующие входных данных, обычно читают информацию из стандартного потока ввода. Например, команда *wc* подсчитывает количество строк, слов и символов во входных данных. Если запустить эту команду без аргументов, то *wc* будет ожидать входных данных с терминала (чтобы закончить ввод данных, нужно нажать комбинацию клавиш *Ctrl-D*):

```
$ wc
two words
<Ctrl-D>
      1       2      10
```

В данном примере программа *wc* прочитала введенный пользователем текст из стандартного потока ввода (куда пользователь ввел текст «*two words*»). По умолчанию этот поток соединен с терминалом (с клавиатурой) пользователя, но допускается его перенаправление. Чтобы связать данные стандартного входного потока с произвольным файлом, можно использовать операцию перенаправления «<», например:

```
$ wc < /etc/passwd
     28      37    1052
```

В данном случае команда *wc* уже не требует ввода с клавиатуры, т. к. она уже получила входные данные из файла */etc/passwd*. Заметим, что данная команда может иметь практическое применение – первая цифра означает количество строк в файле */etc/passwd*, что соответствует количеству пользователей, зарегистрированных в системе.

Стандартный поток вывода – это поток, куда программы записывают выходные данные. В предыдущем примере команда *wc* выводила результат (три числа) именно в этот поток. Так же работают и большинство других неинтерактивных команд (включая *echo*, *pwd* и *ls*, рассмотренные выше). Подобно стандартному потоку ввода выходной поток изначально связан с терминалом и также допускает перенаправление. Для связывания стандартного потока вывода с файлом используется операция «>», например:

```
$ ls > filelist.txt
```

В этом примере команда *ls*, вместо того, чтобы вывести список файлов на экран, записала его в файл с именем «*filelist.txt*». При этом, если файл с таким именем не существовал, он будет создан, в противном случае его старое содержимое будет потеряно. Существует и другая возможность перенаправления вывода, когда новые выходные данные будут дописаны в конец существующего файла. Для этого используется операция «>>». В следующем примере текущие дата и время будут дописаны в конец файла с именем «*dates.txt*»:

```
$ date >> dates.txt
```

Сообщения об ошибках выводятся в стандартный поток ошибок. Например, пусть выполняется попытка получить список файлов в каталоге без соответствующих прав доступа:

```
$ ls -l /home/ftp/bin/
ls: /home/ftp/bin/: Access denied
```

В данном случае команда *ls* вывела сообщение в поток стандартной ошибки. Чтобы перенаправить его в указанный файл, можно использовать операции «2>» и «2>>» (по аналогии с «>» и «>>», только цифра 2 говорит о том, что нужно перенаправить поток ошибок), например:

```
$ ls -l /home/ftp/bin/ 2> last-error.txt
```

Операции перенаправления ввода-вывода можно комбинировать, например:

```
$ wc < /etc/passwd 2>> errors.txt > result.txt
```

Существует другой полезный способ перенаправления ввода-вывода – конвейеры команд. Операция «|» (знак вертикальной черты) позволяет перенаправить стандартный поток вывода одной команды на стандартный входной поток другой команды:

```
$ ls -l /etc | less
```

В этом примере команда *ls* выводит длинный список файлов в каталоге */etc*, эти данные попадают на вход программы *less*, которая позволяет пролистывать текст с помощью клавиш управления курсором. Так осуществляется «объединение» двух независимых команд в один «конвейер».

Рассмотрим более сложный пример формирования конвейера команд. Пусть нам требуется получить в файле «*bash-users.txt*» отсортированный список пользователей в системе, пользующихся

командной оболочкой *bash*. Этого можно было бы добиться использованием нескольких команд, сохраняя промежуточные данные во временных файлах (комментарии к командам оболочки приведенные после знака #)

```
$ grep 'bash' /etc/passwd > list1.tmp
# Поиск по заданному шаблону «bash» в файле /etc/passwd
$ sort < list1.tmp > list2.tmp
# Сортировка по алфавиту данных из файла list1.tmp и запись в list2.tmp
$ cut -f1 -d: < list2.tmp > bash-users.txt
#Выделение первых полей строк по разделителю :
# и запись в файл bash-users.txt
$ rm list1.tmp list2.tmp
# Удаление временных файлов
```

Конвейеризация команд позволяет обойтись одной составной командой без использования промежуточных файлов

```
$ grep 'bash' /etc/passwd | sort | cut -f1 -d: > bash-users.txt
```

Заметим, что команды типа *sort* или *cut* часто называют фильтрами. Фильтры получают данные из стандартного входного потока, преобразовывают их и выводят в стандартный поток вывода.

Завершение работы с *Unix*

Каждый сеанс работы с ОС *Unix* должен заканчиваться вводом команды *logout*. Также можно использовать комбинацию клавиш *Ctrl-D*, которая позволяет выполнить команду завершения работы с командной оболочкой, после чего система переходит в режим ожидания регистрации следующего пользователя. Если сеанс работы производился с удаленной машины с использованием протоколов *telnet* или *ssh*, то завершение работы командной оболочки вызывает разрыв соединения.

1.1.2. Основы интерактивной работы в оболочке bash

Оболочка (*shell*) или командный интерпретатор в *Unix*-системах обеспечивает два набора функций:

- интерпретация командного языка и исполнение команд, введенных пользователем или подготовленных заранее в текстовом файле;

- интерактивное взаимодействие с пользователем, т. е. предоставление пользователю возможности редактирования и ввода команд.

Ниже рассмотрены особенности работы второй группы из набора функций, т. е. интерактивные возможности командной оболочки *bash*, которая является стандартной для систем *GNU/Linux*, и может быть установлена в других *Unix*-подобных системах.

Оболочка *bash* предоставляет пользователю развитые средства интерактивной работы. В частности, она поддерживает редактирование командной строки, повтор символов, макросы, «карман» (буфер), а также историю команд (т. е. возможность повторить ранее введенную команду) и настраиваемое автоматическое дополнение.

Следует отметить, что умение пользоваться интерактивными возможностями оболочки значительно повышает эффективность работы в *Unix*-системе (особенно в сочетании с хорошим знанием командного языка). Более того, работа непосредственно в командной оболочке часто оказывается значительно более продуктивной по сравнению с использованием файловых менеджеров, таких как *Norton Commander*, *Far Manager* или *Windows Explorer*. Обратная сторона преимуществ работы в оболочке *Unix* заключается в длительном начальном периоде изучения.

Далее рассмотрим лишь некоторые наиболее используемые приемы интерактивной работы. Для более полного описания возможностей оболочки следует пользоваться руководством по использованию *bash* (команда *man bash*).

Редактирование командной строки

Классические оболочки *Unix* позволяли вводить команды как последовательность символов, завершая ввод нажатием клавиши *Enter*. Современные версии командных оболочек, такие как *bash*, включают развитые средства редактирования.

Для многих функций редактирования используются комбинации клавиш с модификаторами *CTRL* и *META*. Модификатор *CTRL* имеется на клавиатуре *IBM*-совместимых компьютеров, а в качестве *META* чаще всего используется клавиша *ALT*. Работоспособность модификатора *META* зависит от настройки терминала, графической среды или программы удаленного доступа. Если с помощью клавиши *ALT* не удается добиться желаемого результата, можно использовать альтернативный способ ввода *META*-комбинаций. Для этого перед символом нужно нажать (и отпустить) клавишу *Esc*. Таким образом, например, комбинацию клавиш *META-d* можно заменить

последовательностью нажатий *Esc*, *d*. Для ввода комбинаций наподобие *META-_* (знак подчеркивания) или *META->* (знак «больше») необходимо нажимать и удерживать клавишу *Shift*.

В табл. 1.1–1.3 приведены основные команды для работы в командной строке. Одному действию соответствует, как правило, несколько разных комбинаций клавиш, т. к. их работоспособность может зависеть от типа терминала. Поэтому, если не работает какая-либо из клавиш (например, *Home*), вместо нее может быть использована альтернативная комбинация (например, *CTRL-a*). Также следует отметить, что многие из комбинаций клавиш имеют аналогичное или похожее назначение и в других программах, распространенных в *Unix*.

Таблица 1.1

Команды перемещения по командной строке

Комбинация клавиш	Описание действия
Вправо *CTRL-f*	Перемещение на один символ вправо
Влево *CTRL-b*	Перемещение на один символ влево
META-вправо *META-f*	Перемещение на одно слово вправо
META-влево *META-b*	Перемещение на одно слово влево
Home CTRL-a	Перемещение в начало строки
End CTRL-e	Перемещение в конец строки

Таблица 1.2

Удаление и вставка фрагментов команд

Комбинация клавиш	Описание действия
Backspase CTRL-h	Удалить символ слева от курсора
Del CTRL-d	Удалить символ в позиции курсора
CTRL-u	Вырезать часть строки слева от курсора
CTRL-k	Вырезать часть строки справа от курсора
META-Backspace CTRL-w	Вырезать слово слева от курсора
META-d	Вырезать слово справа от курсора
CTRL-y	Вставить последний вырезанный текст в позицию курсора
CTRL-/ CTRL-_	Отменить последнюю операцию редактирования

Таблица 1.3

Прочие комбинации клавиш

Комбинация клавиш	Описание действия
Enter	Выполнить текущую команду (положение

	курсора не имеет значения)
CTRL-L	Очистить экран и поместить текущую команду в верхней строке экрана
CTRL-d	Выйти из оболочки *bash*, аналогично вводу команды *logout* (только при условии чтокомандная строка пуста)

Использование истории команд

Оболочка *bash* поддерживает историю команд, т. е. запоминает введенные ранее команды. Это позволяет вернуться к любой ранее введенной команде, а также использовать отдельные фрагменты команд из истории для ускорения ввода новых команд. История сохраняется при выходе из оболочки в файле с именем *.bash_history* в домашнем каталоге пользователя и загружается вновь при следующем запуске *bash*. Таким образом, история команд не пропадает в перерывах между сеансами работы. Впрочем, существует ограничение на количество запоминаемых команд (например, 1000), и при превышении этого ограничения самые ранние команды будут автоматически удаляться.

Чтобы просмотреть историю команд, можно использовать команду *history*. Если после имени этой команды указан числовой аргумент, то будет выведено соответствующее число последних введенных команд. Например

```
$ history 5
 4995  mkdir tmp/work
 4996  cd tmp/work
 4997  cp ~/work/log.txt.
 4998  joe log.txt
 4999  history 5
```

Как видно из вывода команды *history*, каждой команде поставлен в соответствии ее порядковый номер в истории. Чтобы выполнить одну из команд истории, можно ввести в командной строке заданный номер, предварив его восклицательным знаком. Например:

```
$ !4996
cd tmp/work
```

Очевидно, что вызов команд с использованием их номера непрактичен. Удобнее использовать похожий синтаксис, указывая вместо номера первые несколько символов команды. В этом случае

будет произведен поиск команды совпадающими с первыми символами, начиная с конца истории, т. е. с недавно вводимых команд. Пример

```
$ !cd
cd tmp/work
```

Однако такой способ также имеет недостатки при практическом использовании из-за возможности легко ошибиться и выполнить неверную команду. Вместо этого чаще используют интерактивные операции навигации и поиска в истории. Наиболее употребительные комбинации клавиш, связанные с историей команд, приведены в табл. 1.4.

Таблица 1.4

Некоторые комбинации клавиш для навигации по истории команд

Комбинация клавиш	Описание действия
Вверх *CTRL-p*	Перейти к предыдущей команде
Вниз *CTRL-n*	Перейти к следующей команде
META-<	Перейти в начало истории команд
META->	Перейти в конец истории команд (т. е. к текущей команде)
CTRL-r	Осуществить обратный инкрементальный поиск в истории команд (см. описание ниже)
META-.	Вставить последнее слово предыдущей команды в текущую позицию курсора
CTRL-o	Аналогично *Enter*, но после выполнения команды показать следующую строку истории

Самый простой способ использования истории заключается в переходе на команду, подобную той, что требуется ввести, ее редактировании и нажатии клавиши *Enter*. Если же при этом вместо *Enter* нажать комбинацию *CTRL-o*, то это позволит повторить ввод серии последовательных команд, сохраненных в истории.

Отдельного внимания заслуживает возможность инкрементального поиска в истории (комбинация клавиш *CTRL-r*). Это, пожалуй, наиболее мощный способ использования истории команд. После нажатия комбинации клавиш *CTRL-r* обычное приглашение к вводу команд исчезает и появляется индикатор режима инкрементального поиска

```
(reverse-i-search)`': _
```

В этом режиме можно вводить символ за символом любую часть команды из истории, и в процессе ввода постоянно видеть наиболее позднюю из совпадающих команд. Например, если происходит поиск команды, содержащей подстроку «*web*», то после нажатия *CTRL-r* вводим сначала букву «*w*»

```
(reverse-i-search) 'w': cd tmp/work
```

увидим, что поиск пока не дал нужного результата, и уточняем поиск, вводя следующую букву, «*e*»

```
(reverse-i-search) 'we': ./update-web.sh
```

Теперь видно, что найденная команда уже содержит фрагмент «*web*» и для ее нахождения было достаточно ввести лишь два символа. Если же найденная команда оказалась не той, что искали, можно использовать *CTRL-r* для перехода на более ранние команды, также содержащие строку поиска. Продолжая предыдущий пример, повторно нажимаем *CTRL-r*. При этом будет найдена другая, более ранняя команда, например

```
(reverse-i-search) 'we': cd work/web/homepage/
```

Теперь можно выйти из режима поиска несколькими способами. Чтобы перейти на найденную команду в истории, достаточно нажать *Esc* или комбинацию клавиш *CTRL-j*. Чтобы отменить поиск и вернуться в исходное состояние, можно нажать *CTRL-g*. И наконец, нажатие *Enter* приведет к немедленному исполнению найденной команды.

Использование автоматического дополнения в командной строке

Автоматическое дополнение (*completion*) позволяет значительно ускорить ввод команд, имен файлов, имен переменных и имен машин в командной строке. Например, пусть в системе установлена программа *bunzip2* и нет ни одной другой программы или команды, начинающейся буквами «*bun*». В таком случае в *bash* достаточно набрать в начале командной строки эти три буквы и нажать клавишу *Tab*. При этом остальные символы, формирующие имя команды, будут вставлены автоматически. В оболочке *bash* поддерживается несколько типов дополнения и множество комбинаций клавиш для их активизации. Рассмотрим лишь две наиболее полезные возможности выполнять автоматическое дополнение (табл. 1.5).

Таблица 1.5

Возможности автоматического дополнения в командной строке

Комбинация клавиш	Описание действия
Tab	Дополнение наиболее подходящим окончанием
META-Tab	Дополнение на основе фраз из истории команд (поскольку роль модификатора *META* часто исполняет клавиша *ALT*, а комбинация *ALT-Tab* обычно используется графической средой для вызова этой команды рекомендуется использовать последовательность нажатий *Esc*, *Tab*).

Дополнение с помощью *Tab* может работать по-разному в зависимости от использования контекста. Табл. 1.6 в упрощенном виде показывает правила выбора типа дополнения.

Таблица 1.6

Возможности автоматического дополнения в командной строке

Контекст	Тип дополнения
Начало строки	Дополнение имени команды (поиск среди имен встроенных команд оболочки и программ в переменной среды *PATH*)
После символа *$*	Дополнение имени переменной (поиск среди имен установленных переменных среды)
После символа *@*	Дополнение имени машины (поиск среди имен машин в файле */etc/hosts*)
После символа ~	Дополнение имени пользователя (поиск среди имен известных системе пользователей)
После шаблона имени файла	Замена шаблона, только если найден лишь один подходящий файл (в данном случае производится не дополнение, а замена шаблона на подходящее имя файла)
В остальных случаях	Дополнение имени файла (поиск среди имен файлов и каталогов)

Дополнение с помощью *META-Tab* всегда ищет дополнения в истории команд, выбирая фразы, начинающиеся с символов, стоящих перед текущей позицией курсора. Если однозначного варианта не найдено, независимо от типа дополнения дописывается только часть, совпадающая во всех вариантах, и, в зависимости от конфигурации оболочки, может быть выведен список подходящих дополнений. Если список вариантов не выводится автоматически, его обычно можно вывести повторным нажатием *Tab* или *META-Tab*.

1.1.3. Файловая система

Особенности формирования файлового пространства

Файловое пространство *Unix*–систем представляет собой иерархию файлов, которая имеет единый общий корень – так называемый корневой каталог, обозначаемый знаком прямой косой черты «/». Чтобы однозначно идентифицировать любой файл, можно указать путь к этому файлу от корневого или текущего каталога. Все элементы пути отделяются друг от друга символом прямой косой черты. Если первый символ строки также косая черта, то путь берет начало в корневом каталоге, в противном случае – в текущем. Путь с единственным именем обозначает файл в текущем каталоге. Примеры

- *docs.ps* – файл с именем *docs.ps* в текущем каталоге;
- */usr/doc/FAQ/README* – файл с именем *README* в каталоге */usr/doc/FAQ;*
- *work/thesis.tex* – файл *thesis.tex* в подкаталоге *work* текущего каталога.

Понятие текущего каталога несколько отличается от такового в системе *MS-DOS* или *Windows*. В *Unix* у каждого процесса собственный текущий каталог. Корневой каталог файлового дерева *Unix* обычно содержит следующие подкаталоги (в разных системах эта структура может отличаться)

- */bin* – минимальный набор исполняемых файлов, необходимый для работоспособности системы;
- */etc* – файлы конфигурации системы;
- */dev* – файлы устройств;
- */home* – домашние каталоги пользователей;
- */lib* – основные системные библиотеки и модули;
- */root* – каталог администратора системы root;
- */proc* – файлы-образы выполняющихся процессов;
- */sbin* – минимальный набор утилит администратора;
- */tmp* – каталог для временных файлов;
- */usr* – основной объем файлов системы: установленные программы, библиотеки, исходные тексты ядра, файлы данных и прочее;
- */var* – каталог для изменяющейся информации (учетных данных, почтовых ящиков, очередей принтера, отформатированных страниц документации, логов и др.).

Следует отметить, что символ косой черты не является частью имен каталогов, а лишь указывает, что данные элементы находятся в

корневом каталоге. В каждом каталоге также существует два особых «подкаталога» с именами «две точки» и «точка». Первый из них служит указателем на однозначно определенный родительский каталог (вышестоящий), а второй – на данный текущий каталог. Например, путь *«../readme»* указывает на файл «*readme*», который находится в родительском каталоге (на ступень выше), а путь *«./readme.now»* укажет на файл *«readme.now»*, который находится в текущем каталоге.

Большая часть файлового дерева *Unix* обычно сосредоточена в каталоге */usr*. Как правило, там можно найти следующие подкаталоги

- */usr/bin* – исполняемые файлы;
- */usr/doc* – документация в различных форматах;
- */usr/etc* – файлы конфигурации программного обеспечения, установленного дополнительно;
- */usr/include* – включаемые файлы для программ, например на языке *C*;
- */usr/info* – документация пользователя в гипертекстовом формате *info*;
- */usr/lib* – разделяемые библиотеки;
- */usr/local* – локальное программное обеспечение, файлы данных и библиотеки (этот каталог в некоторых системах может не использоваться);
- */usr/man* – руководства пользователя (*manual pages*);
- */usr/sbin* – утилиты администратора;
- */usr/share* – данные, совместно используемые различными прикладными программами;
- */usr/src* – исходные тексты различных компонент системы, включая ядро.

Описанная в данном случае структура каталогов относится к ОС *Red Hat Enterprise Linux* 5.2.

Формирование имен файлов

В связи с тем, что зачастую для одного языка существует несколько кодировок (например, для русского языка существуют следующие кодировки: *CP866*, *CP1251*, *KOI-8R* и т. д., хотя в последнее время с распространением *UTF8* ситуация постепенно улучшается), то рекомендуется, чтобы имя файла или каталога составлялось из следующих символов

- прописные и строчные латинские буквы;
- цифры;
- символ подчеркивания;

- символ точки;
- знак минуса (не должен быть первым символом имени);
- знак плюса (использовать не рекомендуется).

В каждой конкретной ОС в именах файлов могут быть допустимы и другие символы, но их использование может привести к некорректности работы некоторых программ и, кроме того, может затруднить перенос файлов между разными ОС. Не рекомендуется использовать названия файлов из локальных таблиц кодировок (например, имена файлов на русском языке), т. к. очень часто для одного языка существует несколько кодировок.

Максимальная длина имени файла варьируется в разных системах и зависит скорее от используемой файловой системы, чем от самой ОС. Обычно можно использовать достаточно длинные имена файлов (до 255 символов). Максимальный размер файла в файловой системе также зависит от ее типа. Для современных файловых систем размер файла более 4 Гбайт не является проблемой.

Как отмечалось выше, прописные и строчные буквы в системе *Unix* различаются, т. е. имена «*filename*», «*FILENAME*» и «*FileName*» являются разными. При этом файлы, отличающиеся только регистром букв, могут находиться в одном каталоге.

В отличие от системы *MS-DOS*, знак точки является обычным символом, допустимым в любом месте имени файла, а такого понятия, как расширение имени файла, строго говоря, в системе *Unix* нет. Тем не менее, последние части имен файлов, отделенные от остальной части имени точками, часто указывают на тип файла. В качестве примера имя файла «*my-photo.tiff.gz*» может означать, что файл представляет собой изображение в формате *TIFF*, сжатое программой сжатия *gzip*.

Точка, являющаяся первым символом имени файла или каталога, имеет особое значение: такие имена по умолчанию не выводятся в листинге содержимого каталогов (хотя к ним можно свободно обращаться), для получения полного списка файлов вместо *ls***,** нужно ввести *ls -a*. Другими словами, чтобы сделать файл «скрытым», нужно начать его имя с точки. Этим часто пользуются для именования служебных файлов, на которые не имеет смысла обращать особое внимание.

Просмотр и интерпретация прав доступа к файлам

ОС семейства *Unix* – традиционно многопользовательские системы. Чтобы начать работать, пользователь должен «войти» в систему, введя со свободного терминала свое регистрационное имя и пароль. Человек, зарегистрированный в учетных файлах системы и,

следовательно, имеющий учетное имя, называется зарегистрированным пользователем системы. Регистрацию новых пользователей обычно выполняет администратор системы. Основными минимальными данными, требуемыми для регистрации пользователя в системе, являются

- имя пользователя;
- название группы, к которой относится пользователь;
- пароль.

В *Unix* базовые права доступа к файлам включают три составляющие

- разрешение чтения (обозначается буквой «*r*», от слова *Read*);
- разрешение записи (буква «*w*», от слова *Write*);
- разрешение выполнения (буква «*x*», от слова *eXecute*).

Разрешение на чтение позволяет пользователю читать содержимое файлов, а в случае каталогов – просматривать перечень имен файлов в каталоге (используя, например, команду *ls*).

Разрешение на запись позволяет пользователю писать в файл, т. е. изменять его содержимое. Для каталогов это дает право создавать в каталоге новые файлы и каталоги или удалять файлы в этом каталоге.

Наконец, разрешение на выполнение позволяет пользователю запускать файлы на исполнение (как программы в машинном коде, так и командные файлы). Если на файле стоит атрибут выполнения, то независимо от его имени он считается программой, которую можно запустить (в отличие от *DOS* или *Windows*, в *Unix* возможность исполнения файла не зависит от «расширения» имени файла, такого как *.exe*). Разрешение на выполнение применительно к каталогам означает возможность перехода в этот каталог (например, командой *cd*). Поэтому для каталогов право выполнения часто называют правом поиска. Отметим, что для каталогов биты чтения и выполнения (*r* и *x*) чаще всего используются в паре, т. е. либо присутствуют оба, либо отсутствуют.

В атрибутах доступа к файлам, перечисленные типы прав доступа могут быть предоставлены для трех классов пользователей:

- владельца (у каждого файла в *Unix* есть один владелец);
- группы (с каждым файлом связана группа пользователей этого файла);
- всех остальных пользователей.

Набор прав доступа для конкретных файлов можно просмотреть с помощью команды *ls -l*. Например:

```
$ ls -l tmp/
```

```
drwxrwxr-x  10 john    users      1024 Aug 30  2002 newdir
-rw-r-----  1 john    users    173727 Jan 13 23:48 archive-0113.zip
```

В этом примере видно, что владельцем файлов является пользователь *john*, а группой владельцев является группа *users*. Набор букв и прочерков в левой части определяет тип файла (первый символ) и права доступа к файлу (остальные девять символов). В приведенном примере первая запись относится к каталогу (первая буква *d*) и демонстрирует права доступа *rwxrwxr-x*. Вторая запись относится к обычному файлу (прочерк на месте первого символа) и показывает права *rw-r-----*. Девять символов прав доступа определяют возможность чтения (*r*), записи (*w*) и выполнения (*x*) для владельца файла (первые три символа), группы владельца (следующие три символа) и всех остальных (последние три символа). Прочерки означают отсутствие соответствующих прав. Следовательно, в приведенном примере

- *john* и все пользователи группы *users* могут просматривать и изменять содержимое каталога *newdir*, а также переходить в него, а остальные пользователи могут читать и переходить в этот каталог, но не могут создавать или удалять в нем новые файлы;
- *john* может читать и изменять файл *archive-0113.zip*, пользователи группы *users* могут только читать содержимое этого файла, а все остальные не имеют к нему никаких прав доступа.

Кроме символьного представления прав доступа часто используется цифровая форма. В цифровом представлении права доступа составляются из трех восьмеричных цифр, каждая из которых определяет набор из трех битов полномочий *rwx*. Чтобы перевести права доступа из символьного представления в числовое, следует:

- представить набор прав в двоичном виде (например, 110100000 для набора прав *rw-r-----*);
- перевести полученное двоичное число в восьмеричную систему счисления (например, восьмеричным представлением двоичного числа 110100000 будет 640).

Права доступа так же можно числовой форме путем суммирования восьмеричных значений отдельных битов прав доступа

- 400 – владелец имеет право на чтение;
- 200 – владелец имеет право на запись;
- 100 – владелец имеет право на выполнение;
- 040 – группа имеет право на чтение;
- 020 – группа имеет право на запись;
- 010 – группа имеет право на выполнение;
- 004 – остальные имеют право на чтение;

- 002 – остальные имеют право на запись;
- 001 – остальные имеют право на выполнение.

Можно заметить, что для прав доступа *rw-r-----* получим: 400 + 200 + 040 = 640.

Типы файлов

В ОС *Unix* имеются следующие основные типы файлов

- обычные файлы (*regular files*);
- каталоги (*directories*);
- символьные ссылки (*symbolic links*);
- файлы физических устройств (*device files*);
- именованные каналы (*named pipes*);
- доменные гнезда (*sockets*).

Обычные файлы используются наиболее широко и представляют собой именованные наборы данных с возможностью произвольного доступа.

Каталоги – специальный тип файлов, позволяющий группировать вместе другие файлы и каталоги. Содержимое каталога представляет собой список находящихся в нем файлов.

Файлы устройств в *Unix* являются средством общения прикладных программ с драйверами оборудования компьютера. Для того, чтобы передать данные драйверу какого-либо устройства, прикладная программа должна произвести запись в соответствующий специальный файл. По аналогии, операция чтения из файла устройства означает получение данных от его драйвера. Обычно каталог с файлами имеет имя «*/dev*».

Символьные ссылки подобны «ярлыкам» в *Windows*. Они позволяют создавать альтернативные имена файлов, причем могут указывать на файлы в других каталогах. При открытии программой символьной ссылки фактически открывается файл, на который она указывает. Символьные ссылки могут указывать как на обычные файлы, так и на каталоги и файлы других типов.

Именование каналы еще называют буфером *FIFO* (*First In First Out*). Через файлы такого типа два независимых процесса могут обмениваться данными: все, что записано в файл одним процессом, может быть прочитано другим.

Гнезда – это абстрактные конечные точки сетевого соединения. Записывая данные в этот файл, процесс отсылает их в сеть. При этом процессы, установившие связь через пару гнезд, могут быть запущены как на разных компьютерах, так и на одном.

Монтирование сторонних файловых систем

Как было представлено выше, к файловой системе *Unix* могут быть подключены сторонние файловые системы, например файловые системы других ОС или файловые системы, расположенные на внешних носителях (флоппи-дисках, *CD-ROM* и др.). Чтобы сторонняя файловая система была доступна ОС *Unix*, необходимо осуществить операцию ее монтирования. Фактически, монтирование – это указание того, куда системе следует адресовываться при обращении к объектам сторонней файловой системы. Для системы это указание называется точкой монтирования.

Теоретически можно указать системе произвольное место точки монтирования, но на практике для монтирования сторонних файловых систем существует каталог */mnt*. В нем необходимо создать подкаталог, который будет служить точкой монтирования. Например файловую систему *Windows*, физически расположенную на первом логическом разделе того же винчестера (на диске *C*), можно сделать доступной для *Unix*, примонтировав ее с помощью команды *mount* (предварительно нужно создать каталог */mnt/windows*)

mount -t vfat /dev/hda1 /mnt/windows (пример для Linux),

mount -t msdos /dev/ad0s1 /mnt/windows (пример для FreeBSD),

где ключ *t* и аргумент *vfat (msdos)* означают тип монтируемой сторонней файловой системы (в данном случае *FAT*), */dev/hda1* – первый раздел первого жесткого диска, к которому система обращается с помощью файла устройства; */mnt/windows* – каталог, представляющий собой точку монтирования.

Системный файл регистрации пользователей

В *Unix*-системах регистрация пользователей ведется в файле */etc/passwd*. Содержимое этого файла представляет собой последовательность текстовых строк. Каждая строка соответствует одному зарегистрированному в системе пользователю и содержит семь полей, разделенных символами двоеточия. Эти поля таковы

- регистрационное имя пользователя;
- зашифрованный пароль;
- значение *UID*;
- значение *GID* основной группы;
- комментарий (может содержать расширенную информацию о пользователе, например, имя, должность, телефоны и т. п.);
- домашний каталог;

- командная оболочка пользователя.

Файл */etc/passwd* должен быть доступен для чтения всем пользователям, т. к. к нему должны обращаться многие программы, запускаемые от имени рядового пользователя (например, чтобы узнать соответствие *UID* регистрационному имени). Но доступность для чтения всех зашифрованных паролей серьезно уменьшает безопасность системы, потому что современные вычислительные мощности позволяют сравнительно быстро подбирать пароли (в особенности, неудачно выбранные некоторыми пользователями). Поэтому часто используется схема теневых паролей (*shadow passwords*), при которой поле пароля в */etc/passwd* игнорируется, а реальный пароль берется из другого файла (например, */etc/shadow*), доступного для чтения только привилегированному пользователю. Файл теневых паролей часто содержит и другую важную информацию: срок, в течение которого допускается использование неизменного пароля, дата последнего изменения пароля и т. п. При использовании теневых паролей второе поле в */etc/passwd* обычно содержит символ звездочки или любой другой произвольный символ. Пустым поле пароля в */etc/passwd* оставлять нельзя, так как в этом случае система может посчитать, что данному пользователю пароль не требуется. Пример строки из файла */etc/passwd* (заметьте, что поле комментария в данном случае отсутствует)

```
john:*:1004:101::/home/john:/bin/sh
```

Дополнительную информацию о формате файла */etc/passwd* можно получить, набрав команду *man* по теме *passwd* в разделе 5 (форматы файлов). Эта команда будет выглядеть так

```
man 5 passwd
```

Файл регистрации групп пользователей

Информация о группах, известных системе, содержится в файле */etc/group*. Подобно файлу регистрации пользователей, информация в */etc/group* представляет собой набор строк, по одной для каждой зарегистрированной группы пользователей. Каждая строка содержит четыре поля, разделенных двоеточиями

- регистрационное имя группы;
- пароль группы (обычно это поле пустое, так как группам обычно не назначают пароли);
- значение *GID*, соответствующее данной группе;
- разделенный запятыми список пользователей, входящих в группу (может быть пустым).

Заметим, что пустой список пользователей в записи */etc/group* не означает, что в этой группе нет ни одного пользователя, так как *GID* основной группы пользователя определяется в файле */etc/passwd*.

Определение идентификаторов пользователей и групп

Чтобы определить *UID* пользователя, *GID* и имя его основной группы, а также список прочих групп, в который включен пользователь, можно использовать команду *id*. В случае ее использования без аргументов, команда выведет информацию о текущем пользователе. Если же указать в качестве аргумента имя зарегистрированного пользователя, вывод команды будет соответствовать указанному пользователю.

Частным случаем команды *id* является команда *groups*. Она выдает список имен всех групп, в которые включен текущий или указанный пользователь.

Ввод команды *who* без аргументов позволяет получить список пользователей, работающих в данный момент в системе. Если же набрать *whoami*, система выведет информацию о текущем пользователе.

Как обычно, дополнительную информацию о всех перечисленных командах можно получить с помощью команды *man*, например

```
$ man who
```

Изменение владельцев файлов

Владельцем файла становится пользователь, создавший этот файл. Группой владельца по умолчанию становится основная группа регистрации пользователя. Для изменения владельцев предназначена стандартная команда *chown* (*change owner*). Однако в современных системах владельца файлов может изменять только привилегированный пользователь (*root*). У обычного пользователя существует возможность изменения только группы владельцев, и то лишь в пределах тех групп, в которые входит сам пользователь. Для изменения группы владельцев удобно использовать команду *chgrp* (*change group*). Например, чтобы сделать группой владельцев каталога *newdir* группу *student*, можно ввести

```
chgrp student newdir
```

Существует возможность рекурсивного изменения владельцев для всех файлов и подкаталогов заданного каталога. Для этого следует использовать ключ *R*, например

```
chgrp -R student newdir
```

Заметим, что вместо одного имени файла или каталога в приведенных командах можно использовать множество имен, разделенных пробелами, или шаблоны имен файлов (это относится и к большей части других команд *Unix*, принимающих имена файлов в качестве последних аргументов).

Изменение прав доступа к файлам

Изменить права доступа к файлу может либо его владелец, либо привилегированный пользователь (*root*). Делается это командой *chmod* (*change mode*). Существует два формата использования этой команды: с использованием символьного и числового представления прав доступа. Использование числового представления позволяет одной командой изменить полный набор прав доступа, например

```
chmod 770 newdir
```

Данная команда установит права доступа в числовое значение 770, т. е. *rwxrwx*---, что даст полные права владельцу и группе владельца, и никаких прав всем остальным.

Использование символьного представления прав доступа в команде *chmod* может показаться несколько сложнее, но позволяет манипулировать отдельными битами прав доступа. Например, чтобы снять бит записи для группы владельца каталога *newdir*, достаточно ввести

```
chmod g-w newdir
```

Условный синтаксис этой команды таков

```
chmod {u,g,o,a}{+,-,=}{r,w,x} файлы ...
```

В качестве аргументов команда принимает указание классов пользователей

- «*u*» – владелец-пользователь (*user*),
- «*g*» – владелец-группа (*group*),
- «*o*» – остальные пользователи (*others*),
- «*a*» – все вышеперечисленные группы вместе (*all*).

Операцию, которую необходимо произвести с правами доступа:

- «+» – добавить,
- «-» – убрать,
- «=» – присвоить;

Права доступа («*r*», «*w*», «*x*») назначаемы каталогам и файлам.

Как и в команде *chgrp*, в *chmod* может использоваться ключ *R*, позволяющий рекурсивно обрабатывать содержимое подкаталогов.

Права доступа по умолчанию, команда umask

Очевидно, что при создании новых файлов и каталогов они уже будут обладать определенным набором прав доступа. Эти права доступа, устанавливаемые по умолчанию, определяются значением маски прав доступа, которая устанавливается командой *umask*. При вводе команды *umask* без аргументов она выведет текущее значение маски, при использовании восьмеричного числа в качестве аргумента будет установлено новое значение.

Маска прав доступа определяет, какие права должны быть удалены из полного набора прав, т. е. маска прав доступа является в некотором роде обратным значением прав доступа по умолчанию. Например, маска 022 приведет к сбросу битов записи для группы владельца и остальных пользователей. Заметим, что для обычных файлов (не каталогов) все биты выполнения (*x*) в правах по умолчанию будут сброшены независимо от текущей маски.

Пример, демонстрирующий эффект команды *umask*:

```
$ umask
002
$ mkdir dir1
$ ls -l
drwxrwxr-x   2 john     users        1024 Apr 21 07:29 dir1
$ umask 072
$ umask
072
$ mkdir dir2
$ ls -l
drwxrwxr-x   2 john     users        1024 Apr 21 07:29 dir1
drwx---r-x   2 john     users        1024 Apr 21 07:30 dir2
```

1.2. ПОСЛЕДОВАТЕЛЬНОСТЬ ВЫПОЛНЕНИЯ РАБОТЫ

1. Ознакомиться с теоретическим материалом.
2. Зарегистрироваться в системе под именем, выданным преподавателем.
3. Ознакомиться со следующими командами для пользовательской работы в ОС *Unix*: *man, apropos, ls, cd, pwd, mkdir, rmdir, cp, mv, rm, cat, echo, less, touch, grep, date, history*. Определить параметры, которые следует считать основными при использовании данных команд.

4. Определить абсолютный путь своего домашнего каталога.

5. Определить значения следующих переменных окружения: *PATH*, *MANPATH*, *PAGER*.

6. Определить границы файлового пространства, где система позволяет создавать собственные файлы и каталоги (возможно использование автоматического скрипта).

7. Проверить, возможно ли вмешательство в личное файловое пространство другого пользователя.

8. Ознакомиться с командами определения прав доступа к файлам и их изменения (команды *id, groups, ls -l, stat, chmod, chown, chgrp, umask*).

9. Найти запись в файле */etc/passwd*, соответствующую вашему регистрационному имени.

10. Определить свой *UID*, узнать, к каким группам относится ваше регистрационное имя, объяснить вывод команд *id, groups*.

11. Определить список групп, в которые входит пользователь *root*.

12. Узнать, какими правами доступа обладают вновь создаваемые файлы и каталоги (т. е. создать новый файл и новый каталог, и просмотреть для них права доступа).

13. Определить значение *umask*, при котором создаваемые файлы и каталоги будут недоступны для чтения, записи и исполнения никому, кроме владельца.

14. Сделать свой домашний каталог видимым для всех пользователей группы *users*.

15. Создать в домашнем каталоге подкаталог *tmp*, файлы в котором сможет создавать, удалять и переименовывать любой, входящий в группу *users*, при этом содержимое этого подкаталога не должно быть видимым всем прочим пользователям.

1.3. ТРЕБОВАНИЯ К ОТЧЕТУ

Отчет должен содержать следующие разделы:

1. Титульный лист, оформленный согласно утвержденному образцу.

2. Цели выполняемой лабораторной работы.

3. Задание на лабораторную работу.

4. Описание процесса выполнения работы: для каждого действия, производимого в командной строке, в отчет следует включить:

- краткое описание действия;
- вводимая команда или команды;

- реакция системы на ввод команд (если объем выводимых данных превышает несколько строк, всю информацию включать в отчет не следует).

5. Выводы.