Справочный материал для выполнения лабораторных работ

Основные команды UNIX-подобных операционных систем

1 Интерфейс командной строки

Будем для краткости называть *Unix-подобные* операционные системы *Unix-системами* или просто *Unix*. .

В *Unix* 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»). Устройства ввода и вывода текста вместе называют терминалом.

Регистрация

Во всех *Unix*-подобных ОС, установленных на 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;
- другую дополнительную информацию.

Регистрационное имя и пароль необходимы для процедуры регистрации пользователя. Идентификатор *UID* используется внутренними функциями системы самим пользователем он 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, но доступное, конечно, и администратору.

При работе в компьютерных классах мехмата для запуска терминала следует выбрать приложение LXTerminal, расположенное на Рабочем столе (после прохождения авторизации).

После авторизации пользователя с указанным именем, система сделает домашний каталог пользователя текущим. При запуске терминала также обычно выполняется некоторый начальный набор команд, который может вывести приветственное сообщение (все эти действия зависят от

особенностей настройки конкретной ОС). И, наконец, на экране появится приглашение к вводу команды:

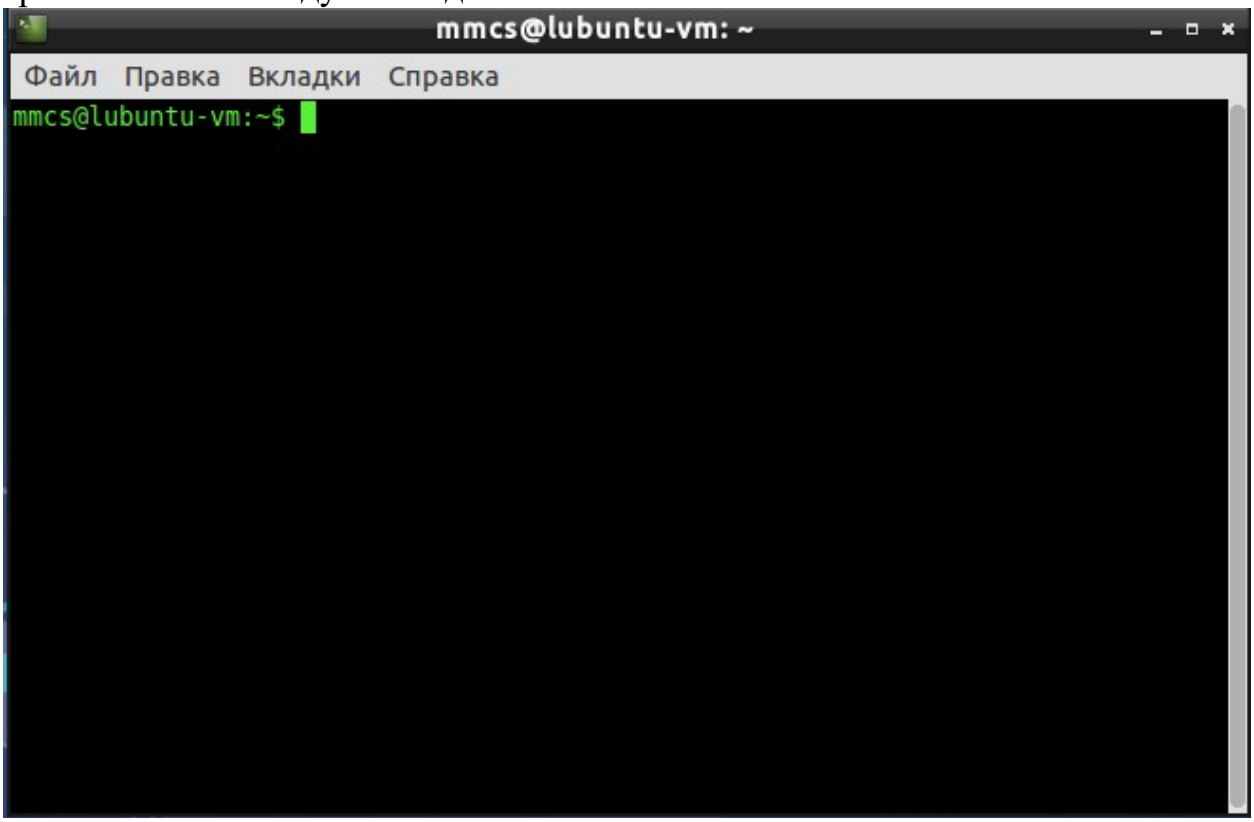

Рис. 1 Вид окна программы LXTerminal.

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

Командные оболочки Unix

В Unix используются различные командные оболочки (*command shells*), называемые также командными процессорами или интерпретаторами команд. По умолчанию в компьютерных классах мехмата установлен в качестве оболочки *bash*.

Основными функциями командных оболочек являются:

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

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;
- 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*.

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* (если вы уже написали какую-нибудь программу из серии “Hello, world!”) из текущего каталога может выглядеть так:

```
:~$ ./hello
```

В этом примере знаки :~$ в начале строки представляют 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*.

Получение справочной информации

Системы *Unix*, как правило, поставляются с огромным количеством справочной информации в электронном виде. 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
```

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 имя каталога 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. Стандартный поток вывода – это поток, куда программы записывают выходные данные. Большинство неинтерактивных команд (включая *echo*, *pwd, ls и другие*) работают со страндартными потоками ввода и вывода. Но эти потоки можно переопределить (перенаправить). 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.

Стандартный поток ввода перенаправляется при помощи операции «<».

Существует и другая возможность перенаправления вывода, когда новые выходные данные будут дописаны в конец существующего файла. Для этого используется операция (метасимвол) «>>». В следующем примере текущие дата и время будут дописаны в конец файла с именем «*dates.txt*»:

```
:~$ date >> dates.txt
```

Наряду с только что описанным способом дописывания информации в файл в UNIX есть отдельные команды, выполняющие подобные действия. В частности, команда cat (это сокращение от concatenate). На самом деле это некая утилита, которая позволяет вам сцеплять, связывать файлы и т.д. Она может выводить содержимое файла на стандартный вывод и многое другое в зависимости от командной строки, в которой используется.
Общий вид команды таков:

```
cat [filename1] [filename2... ]
```

Приведённая команда выводит содержимое перечисленных файлов (filename1, и т.д.) на стандартный поток вывода. Если ни одно имя не указано, то *cat* получает данные со стандартного потока ввода и выводит их в стандартный поток вывода. Благодаря использованию метасимволов перенаправления потоков можно объединить несколько файлов в один и дописать все это в конец 3-го файла. При этом команду *cat* следует записать в следующем виде:

```
:~$ cat filename1 filename2 >> filename3
```

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».

Простейшие команды для работы с содержимым файла

Команда grep сопоставляет строки исходных указанных файлов с шаблоном “строка”, заданным ограниченным регулярным выражением. Если файлы не указаны, используется стандартный поток ввода. Результат выполнения по умолчанию выдается в стандартный поток вывода. Общий вид команды.
grep [опции] строка [файл][файл] ...

Наиболее употребимые опции:

i	поиск без учета регистра
n	отображать номера строк, содержащих контекст “строка”
v	отображать строки, не содержащие контекст “строка”.

К полезным функциям работы с содержимым файла можно отнести
tail [опции] filename
просмотр конца файла. По умолчанию 10 последних строк. С помощью опций можно начать просмотр с любой позиции:

-n number	просмотр с указанной строки
-r number	отображение в обратном порядке
-f	непрерывная выдача файла по мере его заполнения

Команда *find* может искать файлы по имени, размеру, дате создания или модификации и некоторым другим критериям.
Общий синтаксис команды *find* имеет следующий вид:
find [список_каталогов] критерий_поиска
Параметр "список_каталогов" определяет, где искать нужный файл.
Пример.

```
find /usr/share /usr/src  -name readme.txt

/usr/share/cmake/Modules/readme.txt
/usr/share/kde4/apps/ksgmltools2/docbook/xml-dtd-4.1.2/readme.txt
/usr/share/doc/packages/wxGTK/gtk/readme.txt
/usr/share/doc/packages/wxGTK/readme.txt
/usr/share/doc/packages/wxGTK/base/readme.txt
/usr/share/doc/packages/unrar/readme.txt
/usr/share/xml/docbook/schema/dtd/4.1/readme.txt
/usr/share/sgml/docbook/dtd/4.1/readme.txt
/usr/share/sgml/docbook/dtd/3.1/readme.txt
/usr/src/UNIX-2.6.27.42-0.1/Documentation/dvb/readme.txt
```

Завершение работы с *UNIX*
Каждый сеанс работы с ОС *UNIX* должен заканчиваться вводом команды *logout* или *exit*. 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.

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.

Оболочка *bash* предоставляет пользователю развитые средства интерактивной работы. В частности, она поддерживает редактирование командной строки, повтор символов, макросы, буфер обмена, 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.

Удобство работы с командной строкой в оболочках UNIX обеспечивается специальными метасимволами. Приведём кратко самые употребимые из них:

> направляет стандартный вывод в файл

>> добавляет стандартный вывод в конец файла

< берет стандартный ввод из файла

| *p*1 | *p*2 направляет стандартный вывод процесса *p*1 на стандартный ввод *p*2

<<*str* *встроенный документ* (*here document*) – стандартный поток ввода начинается в строке непосредственно после символа, признак окончания – *str*

* соответствует любой строке из нуля и более символов, входящей в имя файла

? соответствует любому отдельному символу в имени файла

[*ccc*] соответствует любому отдельному символу из *ccc* в имени файла, разрешено использование диапазонов, например 0–9 или a–z

; указатель конца команды, *p*1 *; p*2 выполняет *p*1, затем *p*2

& аналогично “;”, но без ожидания завершения *p*1, т.е. команда p1 выполняется в фоновом режиме

$ пример - $var - выполняет подстановку значения переменной var.

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, 100), 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 ниже:

- *CTRL-p* - перейти к предыдущей команде
- Вниз, *CTRL-n* - перейти к следующей 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.

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:

- *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.

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 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 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/lib* – разделяемые библиотеки;
- */usr/local* – локальное программное обеспечение, файлы данных и библиотеки (этот каталог в некоторых системах может не использоваться);
- */usr/sbin* – утилиты администратора;

- */usr/share* – данные, совместно используемые различными прикладными программами;
- */usr/src* – исходные тексты различных компонент системы, включая ядро.

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 представить в 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.

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.

Литература

http://rus-UNIX.net/book1.php?name=book1/oglavl.html#toc5