Содержание

Глава 2: Основные команды и иерархия каталогов

2.1 Оболочка Bourne Shell (bash): /bin/sh

Оболочка (**shell**) - одна из самых важных частей системы Unix. Это программа, выполняющая команды, которые пользователи передает в терминал. Эти команды могут быть другими программами или встроенными функциями оболочки. Оболочка - это еще и среда программирования. Программисты Unix часто разбивают типичные задачи на более мелкие компоненты и применяют оболочку для управления ими.

Существует множество различных оболочек Unix (bash, zsh, fish…), но все они - производные от оболочки Bourne shell (/bin/sh), стандартной оболочки, разработанной в компании Bell Labs для ранних версий Unix. Любая система Unix требует ту или иную версию Bourne shell для корректной работы.

Система Linux использует расширенную версию оболочки Bourne под названием bash, или Bourne-again. Оболочка bash - это оболочка по умолчанию в большинстве дистрибутивов Linux, и каталог /bin/sh обычно указывает (через линк) на bash в системе Linux.

2.3 Основные команды

ls

Перечисляет содержимое каталога.

```
ls -l   # подробный список
```

cp

Копирует файлы.

```
cp file1 file2
```

mv

Переименовывает или перемещает файлы.

```
mv file1 file2
```

touch

Создает файл или обновляет его временную метку.

```
touch file
```

rm

Удаляет файл (без возможности восстановления).

```
rm file
```

echo

Выводит аргументы в стандартный вывод.

```
echo Hello again.
```

2.4 Перемещение по каталогам

Пути в linux начинаются с / - корневого каталога (**root directory**).

. - означает текущий каталог.

.. означает родительский для текущего каталога каталог.

Пусть начинающийся с / называют **абсолютным путем**. Пусть, идущий не от / (а например от . или ..) называют **относительным путем**.

cd

Изменяет текущий рабочий каталог. Без аргумента возвращает в домашний каталог. cd dir # перейти в каталог cd # в домашний каталог

mkdir

Создает новый каталог.

```
mkdir dir
```

rmdir

Удаляет пустой каталог.

```
rmdir dir
rm -r dir   # удалить каталог со содержимым (осторожно!)
```

Wildcards (шаблоны поиска)

Символы расширения имен файлов:

* - любое количество символов

? - ровно один символ

Заключить в одинарные кавычки для буквального вывода

```
echo *        # все файлы
echo at*      # начинаются на 'at'
echo *at      # заканчиваются на 'at'
echo b?at     # boat, brat и т.д.
echo '*'      # выведет звездочку
```

2.5 Команды среднего уровня

grep

Выводит строки, соответствующие выражению. Поддерживает регулярные выражения.

```
grep root /etc/passwd         # поиск 'root'
grep -i pattern file             # без учета регистра
grep -v pattern file             # инвертированный поиск
grep root /etc/*                 # поиск в нескольких файлах
```

less

Просмотр больших файлов постранично. Пробел - вперед, b - назад, q - выход. Поиск: /word (вперед), ?word (назад), n - следующее совпадение.

```
less /usr/share/dict/words
grep ie /usr/share/dict/words | less
```

pwd

Выводит текущий рабочий каталог.

```
pwd
```

```
pwd -P  # истинный путь (без symlinks)
```

diff

Показывает различия между двумя файлами.

```
diff file1 file2
diff -u file1 file2  # унифицированный формат
```

file

Определяет тип файла.

```
file file
```

find и locate

find - поиск в реальном времени. locate - поиск по индексу (быстрее).

```
find dir -name file -print  # с кавычками для шаблонов: find dir -name '*pattern*'
locate file
```

head и tail

Просмотр начала/конца файла (по умолчанию 10 строк).

```
head /etc/passwd        # первые 10 строк
tail /etc/passwd        # последние 10 строк
head -5 /etc/passwd     # первые 5 строк
tail +5 /etc/passwd     # со строки 5
```

sort

Сортировка строк в алфавитно-цифровом порядке. -n для числовой сортировки.

```
sort file
sort -n file  # числовой порядок*
```

passwd

Изменяет пароль пользователя. Запросит старый пароль и дважды новый.

```
passwd
```

chsh

Изменяет оболочку пользователя (zsh, ksh, tcsh и т.д.).

```
chsh
```

2.7 Файлы с точками (дот-файлы)

Скрытые файлы, начинающиеся с точки (.). Не отображаются без параметра -a. Примеры: .bashrc, .login, .ssh.

```
ls -a           # показать дот-файлы
ls              # скрыты дот-файлы
.??*            # шаблон для дот-файлов (без . и ..)
```

2.8 Переменные окружения и оболочки

Переменные оболочки

Временные переменные для хранения текстовых значений.

```
STUFF=blah      # присвоение (без пробелов!)
echo $STUFF     # обращение к переменной
```

Переменные окружения

Передаются всем дочерним процессам. Создаются командой export.

```
STUFF=blah
export STUFF    # переменная окружения
```

2.9 PATH

Специальная переменная окружения - список каталогов для поиска команд (разделены двоеточием).

```
echo $PATH                          # показать  PATH
PATH=dir:$PATH                      # добавить в начало
PATH=$PATH:dir                      # добавить в конец
```

2.10 Специальные символы

Общепринятые названия специальных символов, используемых в Linux и Unix.

Символ	Название	Значение
*	Звездочка (star, asterisk)	Регулярное выражение, символ шаблона
.	Точка (dot)	Текущий каталог, разделитель имени файла/хоста
!	Восклицательный знак (bang)	Отрицание, история команд
|	Вертикальная черта (pipe)	Конвейер (канал) для команд
/	Косая черта (slash)	Разделитель каталогов, команда поиска
\	Обратная косая черта (backslash)	Литералы, макросы (но НЕ каталоги)
$	Доллар (dollar)	Переменные, конец строки
'	Одинарные кавычки (tick, quote)	Не интерпретируемые как шаблоны литеральные строки
`	Обратный апостроф (backtick, backquote)	Подстановка команд
«	Двойные кавычки (double quote)	Частично интерпретируемые строки (полулитеральные)
^	Карет (caret)	Отрицание, начало строки
~	Тильда (tilda, squiggle)	Отрицание, ярлык каталога
#	Октоторп (hash, sharp, pound)	Комментарии, препроцессор, подстановка
[ ]	Квадратные скобки (square brackets)	Диапазоны
{ }	Фигурные скобки (curly brackets, braces)	Блоки выражений, диапазоны
_	Нижнее подчеркивание (underscore, under)	Заменитель пробела в именах

2.11 Редактирование в командной строке

Клавиши со стрелками работают для редактирования, но лучше использовать сочетания управляющих клавиш (Ctrl).

Сочетание клавиш	Действие
Ctrl+B	Переместить курсор влево
Ctrl+F	Переместить курсор вправо
Ctrl+P	Просмотреть предыдущую команду
Ctrl+N	Просмотреть следующую команду
Ctrl+A	Переместить курсор в начало строки
Ctrl+E	Переместить курсор в конец строки
Ctrl+W	Стереть предыдущее слово
Ctrl+U	Стереть от курсора до начала строки
Ctrl+K	Стереть от курсора до конца строки
Ctrl+Y	Вставить стертый текст

2.13 Онлайн-поддержка

Страницы руководства (**man pages**) - основной источник документации.

Просмотр справки для команды:

```
man ls
```

Поиск по ключевому слову:

```
man -k keyword
# Пример: man -k sort
```

Разделы руководства:

Раздел	Описание
1	Пользовательские команды
2	Системные вызовы
3	Библиотеки программирования Unix
4	Интерфейсы устройств и драйверы
5	Файлы конфигурации системы
6	Игры
7	Форматы файлов, соглашения, кодировки
8	Системные команды и серверы

Просмотр определённого раздела:

```
man 5 passwd
```

Дополнительная справка:

- **–help** или **-h** - краткая справка для команды
- **info command** - документация формата GNU info
- **/usr/share/doc** - документация пакетов

2.14 Ввод и вывод командной оболочки

Перенаправление стандартного вывода

Отправить вывод в файл (перезаписать):

```
command > file
```

Добавить вывод в конец файла:

```
command >> file
```

Конвейер (pipe) - отправить вывод одной команды в другую:

```
head /proc/cpuinfo | tr a-z A-Z
```

Стандартная ошибка (stderr)

Перенаправить ошибку в файл:

```
ls /fffffffff > f 2> e
```

Перенаправить ошибку в того же места, что и стандартный вывод:

```
ls /fffffffff > f 2>&1
```

Идентификаторы потоков: **1** = stdout (по умолчанию), **2** = stderr.

Перенаправление ввода

Направить файл на стандартный ввод:

```
head < /proc/cpuinfo
```

Примечание: большинство команд Unix принимают имена файлов как аргументы, поэтому такое перенаправление редко используется.

2.16 Перечисление процессов и управление ими

Процесс - запущенная программа с уникальным числовым идентификатором (**PID**).

Просмотр процессов:

```
ps
```

Основные поля вывода:

Поле	Описание
PID	Идентификатор процесса
TTY	Терминальное устройство
STAT	Состояние (S=sleeping, R=running)
TIME	Процессорное время (мин:сек)
COMMAND	Команда запуска

Параметры команды ps

- **ps x** - ваши процессы

- **ps ax** - все процессы в системе
- **ps u** - подробная информация
- **ps w** - полные имена команд
- **ps aux** - комбинация параметров

Проверка конкретного процесса:

```
ps $$ # Текущая оболочка
```

Завершение процесса

Базовое завершение:

```
kill pid
```

Сигнал	Описание
TERM (15)	Завершение (по умолчанию)
STOP	Приостановка процесса
CONT	Продолжение процесса
INT (2)	Прерывание (аналог Ctrl+C)
KILL (9)	Принудительное завершение (не игнорируется)

```
kill -STOP pid        # Приостановить
kill -CONT pid        # Продолжить
kill -9 pid           # Принудительное завершение
kill -l               # Список сигналов
```

Используйте KILL только в крайнем случае!

Управление заданиями

- **Ctrl+Z** - отправить сигнал TSTP (приостановка)
- **fg** - вывести на передний план
- **bg** - переместить на задний план
- **jobs** - список приостановленных процессов

Фоновые процессы

Запуск процесса в фоне:

```
gunzip file.gz &
```

Оболочка вернёт приглашение и выведет PID процесса. Процесс продолжит работу после выхода из системы.

Для удалённого сеанса используйте **nohup** для сохранения процесса после отключения.

Проблемы фоновых процессов:

- Процесс может ожидать ввода → использовать перенаправление (раздел 2.14)
- Вывод может помешать текущей работе → перенаправить в файл

Обновление экрана терминала:

- **Ctrl+L** - очистка и обновление (bash)
- **Ctrl+R** - обновление строки (или поиск истории)

2.17 Режимы файлов и права доступа

Каждый файл Unix имеет права доступа, определяемые командой `ls -l`. Режим файла состоит из четырех частей:

- **Тип файла** - первый символ: `-` (обычный файл) или d (каталог)
- **Права пользователя** - владельца файла
- **Права группы** - для пользователей в группе файла
- **Права остальных** - для всех прочих пользователей

Каждый набор прав содержит до 3 символов:

Символ	Значение
r	Чтение
w	Запись
x	Выполнение
-	Нет прав

При наличии s вместо x в правах пользователя - это **setuid**, программа выполняется от имени владельца файла (часто суперпользователя).

Изменение прав доступа

Используйте команду chmod:

```
chmod g+r file       # добавить чтение группе
chmod o+r file       # добавить чтение остальным
chmod go+r file      # одной командой
chmod go-r file      # удалить права
chmod 644 file       # абсолютный режим (восьмеричная система)
```

Частые абсолютные режимы:

Режим	Назначение	Применение
644	u: rw, g: r, o: r	Файлы
600	u: rw, g: -, o: -	Файлы (приватные)
755	u: rwx, g: rx, o: rx	Каталоги, программы
700	u: rwx, g: -, o: -	Каталоги, программы (приватные)

Режим	Назначение	Применение
711	u: rwx, g: x, o: x	Каталоги (доступны только владельцу)

Права каталогов: требуются оба права - `r` (просмотр содержимого) и `x` (доступ к файлам).

Маска по умолчанию (umask):

```
umask 022    # все видят созданные файлы
umask 077    # только владелец видит созданные файлы
```

Поместите в файл запуска для постоянного применения (см. глава 13).

Использование символических ссылок

Символическая ссылка (symlink) - это файл-псевдоним, указывающий на другой файл или каталог, скрывая пути каталогов.

В `ls -l` отображается с типом файла `l`:

```
lrwxrwxrwx 1 ruser users 11 Feb 27 13:52 somedir -> /home/origdir
```

При доступе к `somedir` система перенаправляет на `/home/origdir`. Целевой объект может даже не существовать, но программы вернут ошибку при попытке доступа.

Создание:

```
ln -s target linkname
```

- **target** - путь на файл/каталог
- **linkname** - имя создаваемой ссылки
- **флаг -s** - обязателен для символической ссылки

Символические vs жесткие ссылки: Без флага **-s** команда `ln` создает **жесткую ссылку** - второе реальное имя того же файла. Жесткие ссылки еще более запутанные; избегайте их до раздела 4.6.

2.18 Архивирование и сжатие файлов

Две основные утилиты для работы с архивами и сжатием: **gzip** и **tar**.

Утилита gzip

gzip (GNU Zip) - стандартная программа сжатия файлов. Файлы заканчиваются на `.gz`.

```
gunzip file.gz    # распаковать и удалить .gz
gzip file         # сжать файл
```

gzip не создает архивы (не упаковывает несколько файлов в один)!

Утилита tar

tar - создает архивы из нескольких файлов и каталогов.

```
tar cvf archive.tar file1 file2 ...   # создание архива
tar xvf archive.tar                   # распаковка
tar tvf archive.tar                   # просмотр содержимого
```

Флаг	Значение
c	Режим создания архива (create)
x	Режим извлечения (extract)
t	Режим содержания (table of contents)
v	Подробный вывод (verbose); vv - с размерами и правами
f	Файл-параметр; следующий аргумент - имя архива
p	Сохранить права доступа при распаковке

Перед распаковкой всегда проверьте архив флагом t - убедитесь в рациональной структуре каталогов. Создайте временный каталог, если сомневаетесь.

Примечание: tar не удаляет исходный архив после распаковки.

Сжатые архивы (.tar.gz)

Для работы со сжатым архивом - сначала `gunzip`, потом `tar`:

```
gunzip file.tar.gz
tar xvf file.tar
```

Или в один конвейер:

```
zcat file.tar.gz | tar xvf -
```

Быстрое сокращение - используйте флаг `z` в tar:

```
tar ztvf file.tar.gz   # просмотр
tar zxvf file.tar.gz   # распаковка
tar zcvf file.tar.gz file1 file2   # создание
```

Примечание: `.tgz` = `.tar.gz` (сокращение для FAT-файловых систем).

Утилита zcat

zcat действует как `gunzip -dc` - распаковывает и отправляет результат в стандартный вывод. Используется в конвейерах с tar.

Другие утилиты сжатия

Утилита	Суффикс	Распаковка	Свойства
bzip2	.bz2	bunzip2	Сильнее сжимает текст, медленнее gzip
xz	.xz	unxz	Сильнее сжимает текст, медленнее gzip
zip/unzip	.zip	unzip	Совместимы с Windows
compress	.z	gunzip	Древний стандарт Unix (только распаковка)

2.19 Основная иерархия каталогов Linux

Структура каталогов определена Стандартом иерархии файловой системы (FHS). Вот основные подкаталоги корневого каталога `/` :

Каталог	Описание
/bin	Исполняемые файлы основных Unix-команд (`ls`, `cp` и т.д.). Большинство на C, некоторые — скрипты оболочки.
/dev	Файлы устройств (см. глава 3).
/etc	Центральный каталог конфигурации системы: пароли пользователей, загрузочные файлы, сетевые настройки.
/home	Домашние (личные) каталоги обычных пользователей.
/lib	Библиотеки (library). Содержит разделяемые библиотеки, используемые исполняемыми файлами. /usr/lib включает статические и прочие вспомогательные файлы.
/proc	Системная статистика через интерфейс каталогов/файлов. Информация о процессах и параметры ядра.
/run	Данные времени выполнения системы: PID-файлы, сокеты, состояние, системный журнал. (В старых системах — /var/run)
/sys	Интерфейс к устройствам и системе (см. глава 3).
/sbin	Системные исполняемые файлы для администраторов. Обычные пользователи не имеют доступа.
/tmp	Временные файлы. Любой пользователь может читать/писать. ⚠ **Не сохраняйте важные файлы** — очищается при загрузке.
/usr	Основная иерархия системы Linux (см. раздел ниже). Содержит программы и данные пользователей.
/var	Переменные данные: системные журналы, отслеживание активности, кэши, управляемые системными программами.
/boot	Файлы загрузчика и ядра. Первый этап запуска Linux (см. глава 5).
/media	Базовый каталог для съемных носителей (флеш-накопители).
/opt	Дополнительное ПО третьих производителей. Используется не всеми системами.

Каталог /usr

Содержит большую часть пользовательских программ и данных системы. Структура повторяет корневой каталог (например, `/usr/bin` и `/usr/lib`):

Каталог	Описание
/usr/bin	Пользовательские исполняемые файлы.
/usr/sbin	Системные исполняемые файлы.

Каталог	Описание
/usr/lib	Библиотеки (статические, разделяемые и вспомогательные файлы).
/usr/include	Файлы заголовков для компилятора C.
/usr/local	Место для установки собственного ПО администраторами. Структура как в / и /usr.
/usr/man	Страницы руководства.
/usr/share	Файлы, совместимые со всеми Unix-системами. Вспомогательные данные, читаемые программами. (Исторически — для сетевого общего доступа)

Историческая причина: Разделение системы между `/` и `/usr` было сделано для снижения требований к пространству корневого каталога.

Местонахождение ядра

Ядро Linux обычно расположено как двоичный файл:

```
/vmlinuz    или    /boot/vmlinuz
```

Загрузчик (boot loader) загружает этот файл в память при запуске системы (см. глава 5). После запуска основной файл ядра не используется.

Загружаемые модули ядра расположены в папке:

```
/lib/modules
```

Ядро загружает и выгружает эти модули по требованию во время работы системы.