Лекция 3. Работа в bash

Александр Смаль

CS центр

4 марта 2013

Санкт-Петербург

Манипулирование файлами и каталогами

- Манипулирование файлами: `ls`, `touch`, `cp`, `mv`, `rm`.
- Манипулирование каталогами: `mkdir`, `rmdir`, `ls`, `cd`.
- Управление владельцами и правами: `chown`, `chgrp`, `chmod`.
- Создание ссылок: `ln`.
- Поиск файлов: `find`, `locate`.
- Узнать тип файла: `file`.
- Содержимое файлов: `cat`, `more`, `less`, `head`, `tail`.

Манипулирование файлами и каталогами (примеры)

Манипулирование файлами:

```
$ touch file.txt
$ chmod 700 file.txt
$ cp file.txt newfile.txt
$ rm file.txt
$ mv newfile.txt file.txt
```

Манипулирование каталогами:

```
$ pwd
$ mkdir downloads
$ cd downloads
$ ls -la ../test
$ cd -
$ rmdir downloads
```

Файловые маски

Bash поддерживает следующие файловые маски:

- `*.jpg` — `*` заменяет любую последовательность символов.
- `lecture?.txt` — `?` заменяет один символ.
- `lecture{1,2,3}.txt` — `{}` подставляют значения, заданные через запятую.
- `lecture[1235].txt` — `[]` совпадают с любым из перечисленных символов. Поддерживаются промежутки: `[a-c]`, `[1-5]`.

Процессы

- Каждый процесс запускается от имени какого-то пользователя и группы (т.е. имеет `uid` и `gid`).
- При запуске процессу задаётся идентификатор `pid`.
- Список процессов:
  ```
  $ ps au
  ```
- Средства управления процессами:
  ```
  $ kill 5904
  $ killall firefox
  $ nice -10 firefox
  ```
- Для каждого процесса создаётся папка `/proc/<pid>`.

Запуск программ

- Запускать можно только те программы, на запуск которых у пользователя есть права.
- Файлы программ ищутся в специальных местах (`/bin`, `/usr/bin`, ...).

  ```
  $ firefox
  ```

  Можно узнать, где находится программа:

  ```
  $ which firefox
  ```

  Программы по умолчанию **не ищутся** в текущем каталоге, поэтому нужно указывать путь.

  ```
  $ myprog
  myprog: command not found
  $ ./myprog
  ```

Запуск программ (продолжение)

- Расширение файла не влияет на то, как он запускается.
- Если в первой строке текстового файла указать имя интерпретатора, то он будет использоваться для запуска этого файла. Пример:
  `#! /usr/bin/python`
- Запуск можно сделать в фоновом режиме
  `$ ./myprog &`
- Список запущенных задач и управление ими: `jobs`, `fg`, `bg`.

```
$ find /
  <CTRL-Z>
$ jobs
$ fg
```

Потоки ввода/вывода

- У каждого процесса есть три стандартных потока ввода/вывода: `stdin`, `stdout` и `stderr`.
- Дескрипторы файлов `stdin`, `stdout` и `stderr` — 0, 1 и 2.
- Потоки можно перенаправлять в файл и из файла:

```
$ ls -lR > dir-tree.list
$ grep test < dir-tree.list
```

- '>' — перезаписывает файл, '>>' — дописывает в конец.

Перенаправление потоков. Конвейеры

- По умолчанию '>' перенаправляет stdout.
  `$ ls -y 2>error.txt`
  Перенаправление stderr в файл "error.txt".
- '&>' перенаправляет stdout и stderr.
  `$ grep test -r /etc &>results.txt`
- Потоки можно перенаправлять друг в друга:
  `$ ls -y >/dev/null 2>&1`
- Последовательность команд можно связывать в конвейер при помощи символа '|':
  `$ cat *.txt | sort | uniq > result-file`
- Команда xargs переводит stdin в аргументы:
  `$ find . -name '*.txt' | xargs vi`

Bash скрипты

- Скрипт должен начинаться с
  `#!/bin/bash`
- В скрипте можно определять строковые переменные:
  `a='qwerty'`
  `echo $a`
- В скрипте определены специальные переменные:
  - `$0` — имя скрипта.
  - `$1`, `$2`, ... — параметры скрипта.
  - `$#` — общее количество параметров переданных скрипту.
  - `$*` — все аргументы (в строку), `$@` — в столбик.
  - `$!` — PID последнего запущенного в фоне процесса.
  - `$$` — PID самого скрипта.
  - `$?` — код возврата последней запущенной программы.

Bash скрипты

- Задать код возврата: `exit <кода возврата>`.
- Условное выражение:
  `if <команда>; then <команды>; [else <команды>]; fi`
  `if <команда>; then <команды>; [elif <команда>; then <команды>]; fi`
- Циклы:
  `for <переменная> in <список>; do <команды>; done`
  `while <команда>; do <команды>; done`
  `until <команда>; do <команды>; done`
- `True` соответствует коду возврата 0.

Bash скрипты (продолжение)

- Разделитель между командами: ';' или перевод строки.
- Текст в одинарных кавычках не изменяется, в двойных — происходит интерполяция переменных:
  `$ S='test'; echo '$S': "$S"`
- Команда в '` `' заменяется на вывод этой команды:
  `$ for i in 'find -name '*.txt''; do rm $i; done`
- В качестве условия можно обычно используется команда test (а точнее её синоним [ ]):
  `$ if [ $a -eq $b ]; then echo $b; done`
- С переменными можно производить арифметические операции:
  ```
  $ K=0; while [ $K -le 100 ];
  do echo $K; K=$(($K+1)); done
  ```

Как выйти из vi?

Самый важный вопрос сегодняшней лекции!

Как выйти из vi?

Самый важный вопрос сегодняшней лекции!
Wikipedia: *В отличие от многих привычных редакторов, vi имеет модальный интерфейс. Это означает, что одни и те же клавиши в разных режимах работы выполняют разные действия.*

Как выйти из vi?

Самый важный вопрос сегодняшней лекции!
Wikipedia: *В отличие от многих привычных редакторов, vi имеет модальный интерфейс. Это означает, что одни и те же клавиши в разных режимах работы выполняют разные действия.*
Есть много способов выйти из vi. Вот некоторые из них:

Как выйти из vi?

Самый важный вопрос сегодняшней лекции!
Wikipedia: *В отличие от многих привычных редакторов, vi имеет модальный интерфейс. Это означает, что одни и те же клавиши в разных режимах работы выполняют разные действия.*
Есть много способов выйти из vi. Вот некоторые из них:

1. `:q<Enter>` или `:q!<Enter>`

Как выйти из vi?

Самый важный вопрос сегодняшней лекции!
Wikipedia: *В отличие от многих привычных редакторов, vi имеет модальный интерфейс. Это означает, что одни и те же клавиши в разных режимах работы выполняют разные действия.*
Есть много способов выйти из vi. Вот некоторые из них:

1. `:q<Enter>` или `:q!<Enter>`
2. `ZQ`

Как выйти из vi?

Самый важный вопрос сегодняшней лекции!
Wikipedia: *В отличие от многих привычных редакторов, vi имеет модальный интерфейс. Это означает, что одни и те же клавиши в разных режимах работы выполняют разные действия.*
Есть много способов выйти из vi. Вот некоторые из них:

1. `:q<Enter>` или `:q!<Enter>`
2. `ZQ`
3. `<CTRL-Z>`
   ```
   $ kill %1
   $ fg
   ```

Команда man

- Большинство программ поставляется с инструкциями (man pages, manual pages).
- Эти инструкции можно прочитать при помощи команды `man`.
- Начните пользоваться `man` с команды:

Команда man

- Большинство программ поставляется с инструкциями (man pages, manual pages).
- Эти инструкции можно прочитать при помощи команды `man`.
- Начните пользоваться `man` с команды:
  `$ man man`

Спасибо за внимание!