Командный интерпретатор bash

16 февраля 2017 г.

Что это?

- Командная оболочка

Что это?

- Командная оболочка
- Может использоваться интерактивно:

```
user@host:dir$ echo test
test
user@host:dir$
```

Что это?

- Командная оболочка
- Может использоваться интерактивно:

```
user@host:dir$ echo test
test
user@host:dir$
```

- Можно писать сценарии:

test.sh

```
#!/bin/sh
echo test
```

Оно вам не надо

- Неприятно
- Мерзко
- Использовать для чего-то с логикой сильно сложнее, чем «запустить программу для всех файлов в данной директории», сомнительно
  - Напишите лучше на питоне

Вам не убежать

- Установлено на всех *nix-системах

Вам не убежать

- Установлено на всех *nix-системах
- Альтернатив нет

Вам не убежать

- Установлено на всех *nix-системах
- Альтернатив нет
- Часто другого способа общаться с компьютером нет:
  - Не установлена графическая оболочка
  - Удалённый доступ

```
ssh you@rain.ifmo.ru
```

Оно вам надо

- Одноразовые задачи
- Короткие и простые скрипты
  - Полэкрана вынести ещё можно

Оно вам надо

- Одноразовые задачи
- Короткие и простые скрипты
  - Полэкрана вынести ещё можно
- *Однострочники*

```
cat a.log | sed -re "s/([^ ]+).*/\1/" | \
  sort | uniq -c | sort -rn | head -n 10
```

План

- Запускать одну команду интерактивно
- Запускать много команд интерактивно
- Скрипты
- Полезные команды

man

- `man` *тип название*
  - 1 — системные утилиты
  - 2 — системные вызовы
  - 3 — библиотечные функции
  - 6 — игры
  - ...
- Читайте `man man`, чтобы узнать больше

man

- Когда в указании видите “слово(цифра)”, читайте “man цифра слово”
  - read(2) → `man 2 read`
  - signal(7) → `man 7 signal`
- Не всегда цифра уникальна
  - `man 2 pipe`
  - `man 7 pipe`

Опции запуска

- `rm --recursive --force dir`
- `rm -r -f dir`
- `rm -rf dir`
- `rm --help`

Опции запуска

- `rm --recursive --force dir`
- `rm -r -f dir`
- `rm -rf dir`
- `rm --help`
- Как отличить аргумент от ключа?
  - Заархивировать файл “`-cf`”

Опции запуска

- `rm --recursive --force dir`
- `rm -r -f dir`
- `rm -rf dir`
- `rm --help`
- Как отличить аргумент от ключа?
  - Заархивировать файл “`-cf`”
  - `tar -cf x.tar -- -cf`

Перенаправление ввода-вывода

- `grep lol < input.txt > output.txt`
- `grep lol < input.txt >> output.txt`

Перенаправление ввода-вывода

- `grep lol < input.txt > output.txt`
- `grep lol < input.txt >> output.txt`
- `g++ -Wall bad.cpp 2> /dev/null`

Перенаправление ввода-вывода

- `grep lol < input.txt > output.txt`
- `grep lol < input.txt >> output.txt`
- `g++ -Wall bad.cpp 2> /dev/null`
  - 0 = stdin
  - 1 = stdout
  - 2 = stderr

Перенаправление ввода-вывода

- `grep lol < input.txt > output.txt`
- `grep lol < input.txt >> output.txt`
- `g++ -Wall bad.cpp 2> /dev/null`
  - 0 = stdin
  - 1 = stdout
  - 2 = stderr
- `g++ -Wall bad.cpp >/dev/null 2>&1`

Перенаправление ввода-вывода

- `grep lol < input.txt > output.txt`
- `grep lol < input.txt >> output.txt`
- `g++ -Wall bad.cpp 2> /dev/null`
  - 0 = stdin
  - 1 = stdout
  - 2 = stderr
- `g++ -Wall bad.cpp >/dev/null 2>&1`
  - Не то же самое, что
    `g++ -Wall bad.cpp 2>&1 >/dev/null`

globbing

- Сопоставление с образцом
  - `ls *.cpp`
  - `ls 2017-02-??.log`
  - `ls [a-e]*`

globbing

- Сопоставление с образцом
  - `ls *.cpp`
  - `ls 2017-02-??.log`
  - `ls [a-e]*`
- Скрытые файлы
  - `.bashrc`

globbing

- Сопоставление с образцом
  - `ls *.cpp`
  - `ls 2017-02-??.log`
  - `ls [a-e]*`
- Скрытые файлы
  - `.bashrc`

Опасность!

```
$ ls
ImportantDir file1 file2 -rf
$ rm *    # rm ImportantDir file1 file2 -rf
$ ls
-rf       # oops
```

Переменные

- У программы есть *переменные окружения*
  - environ(7), getenv(3)

env.c

```c
#include <stdio.h>
int main(int argc, char **argv, char **envp) {
  for (; *envp; envp++)
    printf("%s\n", *envp);
  return 0;
}
```

```
$ gcc env.c -o env; ./env
PATH=...
USER=...
HOME=...
...
```

Переменные

- Объявление
  - `export VAR=val; prog`
  - `VAR=val prog`

Переменные

- Объявление
  - `export VAR=val; prog`
  - `VAR=val prog`
- Использование
  - `echo $HOME`
  - `PATH=$PATH:$HOME/bin:$HOME/.bin`

Кавычки

- `prog "$HOME"` напечатает путь к домашней директории
  - Подставлять значения переменных
  - `prog "${HOME}"` — то же самое
- `prog '$HOME'` напечатает `$HOME`
  - Не менять ничего

Кавычки

- Какие две команды одинаковые?
  - `prog $HOME`
  - `prog "$HOME"`
  - `prog '$HOME'`

Кавычки

- Какие две команды одинаковые?
  - `prog $HOME`
  - `prog "$HOME"`
  - `prog '$HOME'`

Никакие!

```
$ X="1 2 3"
$ prog $HOME # три аргумента ["1","2","3"]
$ prog "$HOME" # один аргумент ["1 2 3"]

$ prog '$HOME' # один аргумент ["$HOME"]
```

Арифметика

```
$ A=$((2+2))
$ echo $A
4
$ A=$((A+$A))
$ echo $A
8
```

Специальные переменные

- $1, $2, ... — аргументы
- $* — все аргументы
- $# — число аргументов
- $? — код возврата последнего процесса
- $$ — номер процесса
- ... ещё много

Специальные переменные

- $1, $2, ... — аргументы
- $* — все аргументы
- $# — число аргументов
- $? — код возврата последнего процесса
- $$ — номер процесса
- ...ещё много
- Как получить `i`-й аргумент?

Специальные переменные

- $1, $2, ... — аргументы
- $* — все аргументы
- $# — число аргументов
- $? — код возврата последнего процесса
- $$ — номер процесса
- ...ещё много
- Как получить i-й аргумент?
  - `val=${!i}`

Запуск нескольких команд

- Подряд: `cmd1; cmd2; cmd3;`
- Ленивое И: `cmd1 && cmd2 && cmd3`
- Ленивое Или: `cmd1 || cmd2 || cmd3`

Запуск нескольких команд

- Подряд: `cmd1; cmd2; cmd3;`
- Ленивое И: `cmd1 && cmd2 && cmd3`
- Ленивое Или: `cmd1 || cmd2 || cmd3`
- Код возврата
  - `return 0;` — хорошо, программа завершилась без ошибок

Запуск нескольких команд

- Подряд: `cmd1; cmd2; cmd3;`
- Ленивое И: `cmd1 && cmd2 && cmd3`
- Ленивое Или: `cmd1 || cmd2 || cmd3`
- Код возврата
  - `return 0;` — хорошо, программа завершилась без ошибок
  - `if (0) { ... }` — плохо, не заходим в `if`
  - Путаница

Запуск нескольких команд

- Пайпы: `cmd1 | cmd2 | cmd3`

Запуск нескольких команд

- Пайпы: `cmd1 | cmd2 | cmd3`
- Именованные пайпы: `mkfifo`

Запуск нескольких команд

- Пайпы: `cmd1 | cmd2 | cmd3`
- Именованные пайпы: `mkfifo`
- Подкоманды
  - `echo ‘pwd‘`
  - `echo $(pwd)`

Многозадачность

- Запуск в *фоне*

```
$ long_computation_1 &
$ long_computation_2 &
$ wait
$ wait
```

Написание скриптов

- shebang

script.sh

```
#!/bin/sh
echo hello
```

script.py

```
#!/usr/bin/env python3
print("Hello")
```