Bash programming

part 1 - Introduction

Bash 1.1

Miro Slav - 2018

Оболочка операционной системы

Оболочка операционной системы

(от англ. shell «оболочка») интерпретатор команд операционной системы, обеспечивающий интерфейс для взаимодействия пользователя с функциями системы.

Что такое Unix shell?

- Обычная программа, запускающаяся после входа в систему
- Интерактивный командный интерпретатор
- Платформа интеграции (для утилит)
- Язык программирования

Основные типы shell в Unix

- Bourne shell совместимые (POSIX-совместимые)
  - **sh** исходная bourne shell (Steve Bourne, 1978)
  - **ksh** Korn shell (David Korn, 1983)
  - **ash** [BSD] Almquist shell (Kenneth Almquist,1989)
  - **bash** [GPL] Bourne-again shell (Brian Fox, 1989)
  - **zsh** [BSD] Z shell (Paul Falstad,1990)
  - **/bin/sh** Указывает на POSIX-совместимую shell
- C shell совместимые
  - **csh** Исходная C shell (Bill Joy, 1978)
  - **tcsh** [BSD] TENEX C shell (Ken Greer, 1981)

Какие могут быть различия между shell

- Интерактивные возможности
- Платформы
- Поиск соответствий строк и имён файлов
- Программные возможности
- Совместимость оболочек между собой

Как работать с командной строкой

- Находим приглашение командной строки $, #, user@host: $
- Вводим имя команды, опции, аргументы, запускаем на выполнение нажатием <Enter>

Что такое команды?

- исполняемая программа (бинарный файл, скрипт)
- встроенные в оболочку команды (shell built-ins)
- функция оболочки
- сокращение команды (an alias)

Маленькое упражнение

Определить список установленных оболочек в системе.

```
1 cat /etc/shells
2 ls -l <filename> # для каждого элемента /etc/shells
3 readlink -e <filename>
```

Командой type определить какой тип команды (встроенная в оболочку или внешняя программа)

```
1 type type
2 type dmesg
3 type ls
4 type -a test
```

Важные аббревиатуры внутри командной строки

- Для директорий
  - ~ Домашняя директория
  - ~<username> Домашняя директория пользователя
  - `..` Родительская директория
  - `.` Текущая директория
- Wildcards
  - `*` Любой набор символов `file*txt : file1.txt filefilefiletxt`
  - [<список>] символ из заданного набора
  - ? любой один символ

Переменные окружения

- `HOME`
- `PWD`
- `LANG`
- `LD_LIBRARY_PATH`
- `SHELL`
- `TERM`
- `DISPLAY`

Контроль

- `export VAR=value`
- declare -x
- echo

Переменные окружения наследуются при создании нового процесса

Горячие клавиши

- **Tab** – дополнение текущей команды
- История команд (Команда `history`)
  - Клавиши курсора – навигация по истории
  - Ctrl-R – поиск в истории по фрагменту
  - Ctrl-O (после выполнения вставить следующую команду из истории)
- Редактирование командной строки
  - Ctrl-W, Ctrl-U – удалить, начиная от курсора, слово или строку до начала строки
  - Alt-D, Ctrl-K – удалить, начиная от курсора, слово или строку до конца строки
- Управление терминалом
  - Shift-PgUp, Shift-PgDown – прокрутка терминала
  - Ctrl-S – заморозка терминала
  - Ctrl-Q – разморозка терминала
  - Ctrl-L – очистка терминала

Alias

Bash alias

Alias в Bash – это не более, чем клавиатурное сокращение или своего рода аббревиатура, позволяющая сократить количество нажимаемых клавиш для ввода длинных команд.

- `alias` – просмотр сокращений
- `alias <name>=''cmd1;cmd2''` – добавление/модификация сокращений
  `alias tls=''netstat -lpnt | grep 127.0.0.1''`
- `unalias <cmd>` – удаление сокращений

Практическое задание

Создать сокращение для команды `ls` так, чтобы она всегда вызывалась с параметрами `-l` и `-a`.

Bash против скриптовых языков

- В bash трудно делать сложные структуры данных
- В bash нет типизации переменных
- Но! В bash очень просто организовывать взаимодействие внешних программ
- Stream programming `(a | b | c )`

Когда использовать bash

Использовать

- Прототипирование
- Системные скрипты
- Автоматизация консоли

Не использовать

- Критичные по скорости
- GUI
- Сложные структуры данных

Раздвоения личности

- Bash vs POSIX shell
- Внешние и встроенные команды
  - `cd`,`help`,`type`,`alias`,`read` чисто встроенные команды
  - `rm`,`ls` внешние команды (не часть shell по сути)
  - `echo`,`pwd`,`test` встроены в shell, но есть как внешние

Спецсимволы

- # – Вся строка после # является комментарием
- ; – Разделение команд
- : – NOP оператор (похож на встроенный вызов true)
- `source` или `.` – скрипт выполняется в текущем экземпляре shell

Введение

Sha-Bang

```
1 #!/bin/bash
```

Режим совместимости с POSIX

```
1 #!/bin/sh
```

Экранирование

- Экранирование одного символа \
- Частичное экранирование "
- Полное экранирование '

Спецзначения для echo и sed

- \n – новая строка
- \r – возврат каретки
- \t – табуляция
- \v – вертикальная табуляция

```
1 echo -e "test \v test \v test"
```

- \b – перемещение на 1 символ назад
- \a – звуковой сигнал
- \0xxx – 8-миричное число
- \xXX – 16-ричное число

Подстановка команд

Синтаксис:

- `command`
- $(command)

Задание

Присвоить переменной LIST результат выполнения команды `ls -1`
Вывести на экран переменную LIST

Коды возврата

Согласно POSIX: 0 – успех

Код возврата доступен через переменную $?

Пример

```
1 /bin/true; echo $?
2 /bin/false; echo $?
```

Скрипт возвращает код последней команды, поэтому для корректного выхода необходимо использовать `exit`.

exec

Заменяет текущий shell переданной командой.
Часто используется для переназначения файловых дескрипторов.

Группировка

Пример

```
1 ( echo 1; echo 2) | tee file
```

(cmd1; cmd2)

Запускается новый shell

Пример

```
1 TEST=42; (echo $TEST; TEST=0; echo $TEST ); echo $TEST
```

Управление заданиями

- `bg` – поместить задачу в фоновое исполнение
- `fg` – "переключиться" на задачу
- `jobs` – список активных задач
- `wait [pid|id]` – ждать завершения всех, либо конкретных задач

Задание

- Запустить задачу "`sleep 100 &`" в фоновом режиме
- Запустить задачу "`sleep 100`" на текущей косоли и остановить ее (Ctrl-Z)
- С помощью команды "`jobs`" посмотреть статус активных задач
- С помощью команды "`bg`" перевести остановленную задачу в фоновый режим
- С помощью команды "`fg`" переключиться на одну из фоновых задач

Перенаправление ввода/вывода

- “>” – Перенаправление в файл

Пример

```
1 echo stdout > test.txt
```

- “>&” – Перенаправление в другой дескриптор

Пример

```
1 (echo stdout; echo stderr >&2) > test.txt
```

Перенаправление ввода/вывода

- “&>”

Пример

```
1 (echo stdout; echo stderr >&2) &> test.txt
```

- “>>” – Добавление в файл

Пример

```
1 echo stdout >> test.txt
```

- “<” – Чтение из файла

Пример

```
1 cat < test.txt
```

Перенаправление ввода/вывода

- “«” – Here-документ
- “<>” – Открывает файловый дескриптор из файла/другого дескритора

Пример

```
exec 3<>test.txt; echo test >&3;  cat <test.txt
```

- “n<&-” – Закрывает файловый дескриптор

Пример

```
exec 3<&-; echo test >&3
```

- “|” – pipe

Мультистрочный ввод (Here-документ)

```
program <<LABEL
Тут
    много
     строк
LABEL
```

Пример

Передадим несколько строк в COM-порт 1

```
1 cat >/dev/ttyS0 <<E_O_F
2 ATZ
3 ATDT 8w0170123456
4 E_O_F
```