

Bash-скрипты

Кузнецов Антон Михайлович

Санкт-Петербургский Академический Университет

13 сентября 2013 г.

- Сделать исполняемым:
`chmod 555 scriptname`
а затем просто запустить:
`./scriptname`
- **Лучше запускать так, а не через `bash scriptname`**

Пример с переменными

```
a=1
b=$a
echo $b # 1
hello="A B C"
echo hello # hello
echo $hello # A B C
echo "$hello" # A B C
echo '$hello' # $hello
a='cat foo.txt'
b=" ' "
c=' " '
```

- `$@` – параметры скрипта (все!)
- `$0` – имя скрипта
- `$1`, `$2`, `$3`, ... – параметры скрипта по одному
- `$#` – количество переданных аргументов
- `$*`, `$@` – специальная переменная, содержащая все аргументы
- `$?` – код возврата

Команда `test` (или "`[]`") проверяет выполнение некоторого условия. С ее помощью формируются операторы выбора и цикла.

Минус команды `test`:

```
[ privet ]  
echo $? # 0  
[ ]  
echo $? # 1
```

`Test` возвращает 0 (истина), если в скобках стоит непустое слово. **Причина: 0 – код возврата, если все хорошо, т.е. истина**

- if condition
 then
 command1
 else
 command2
 fi
- if test condition точно тоже, что и if [condition]
- if [[condition]] - возможны &&, ||
- if ((арифметическое выражение))

`[[$foo > 7]]` – сравнивает строки

`[$foo > 7]` – перенаправление вывода! :)

Правильно

`(($foo > 7))` - сравнивает числа

`[$foo -gt 7]`

`[[$foo -gt 7]]`

[bar == "\$foo"] - неверно

Правильно

[bar = "\$foo"]

[[bar == "\$foo"]]

- `if EXPR; then команды; fi`
- `if EXPR; then команды; else другое; fi`
- `[! EXPR]` – отрицание
- `[(EXPR)]` – скобки
- `[EXPR1 -a EXPR2]` – И
- `[[EXPR1 && EXPR2]]` – И
- `[EXPR1 -o EXPR2]` – ИЛИ
- `[[EXPR1 || EXPR2]]` – ИЛИ
- `((арифим. выражение))`
- **man test**

- [*-e FILE*] - файл существует
- [*-d FILE*] - это директория
- [*-f FILE*] - это обычный файл
- [*-s FILE*] - размер ненулевой
- [*-r FILE*] - доступен для чтения
- [*-w FILE*] - доступен для записи
- [*-x FILE*] - исполняемый

```
if ! grep -q $USER /etc/passwd; then  
echo "not a local account"  
fi
```

```
grep -q $USER /etc/passwd if [ $? -ne 0 ]; then  
echo "not a local account"  
fi
```

```
if [ "$(whoami)" != root ]; then echo "Oh"; fi
```

```
echo "1 Запуск программы nano"  
echo "2 Запуск программы vi"  
echo "3 Выход"  
read doing  
case $doing in  
1) /usr/bin/nano ;;  
2) /usr/bin/vi ;;  
3) exit 0;;  
*) echo "Введено неправильное действие"  
esac
```

```
function function_name { command... }
```

```
function_name () { command... }
```

НЕ СТОИТ СМЕШИВАТЬ ОПРЕДЕЛЕНИЯ! Функции могут принимать входные аргументы и возвращать код завершения.

return – завершает исполнение функции.

Может возвращать "код завершения"(int), который записывается в переменную \$?.

Если "код завершения" не указан – возвращается код последней команды в функции

Нет опережающего объявления функции, но...

```
foo1 (){  
echo Вызов функции "foo2" из "foo1"  
foo2  
}  
foo2 (){  
echo "Функция foo2"  
}  
foo1
```

```
if [ $USER = student ]
then
student_greet () {
echo "Привет, student!"
}
fi
student_greet
# Работает только у пользователя student, другие получат
сообщение об ошибке.
```

- bc – утилита, выполняющая вычисления с произвольной точностью.
- echo "scale=4;(321-123)/123 bc -l
scale=4 – количество знаков после запятой
- echo "obase=16;ibase=10;123 bc
преобразование из десятичного в шестнадцатеричный вид
- var3=\$(bc -l << EOF
scale = 9; s (1.7)
EOF)

- Длина строки `${#string}`
- Длина подстроки в строке
`expr "$string": '$substring'`
`stringZ=abcABC123ABCabc`
`echo `expr "$stringZ": 'abc[A-Z]*.2'` # 8`
- `expr index $string $substring` - номер позиции совпадения в `$string` с символом в `$substring`.
- Извлечение подстроки `${string:position}` либо `${string:position:length}`

```
expr "$string": '\($substring\).'
```

Находит и извлекает первое совпадение `$substring` в `$string`, где `$substring` – это регулярное выражение.

Удаление части строки

`${string#substring}` - удаление самой короткой

`stringZ=abcABC123ABCabc`

```
echo ${stringZ#a*C} # 123ABCabc
```

Замена подстроки -

`${string/substring/replacement}` – первое

`${string//substring/replacement}` – все `substring`

```
OPERATION=docToPdf
SUFFIX=pdf
directory=$PWD
for file in $directory/*
do
filename=${file%.doc}
$OPERATION $file > "$filename.$SUFFIX"
rm -f $file
done
```

```
#!/bin/bash
if [ "$UID" -eq 0 ]; then echo "Run this as root!"; exit 1; fi
killall -q dhclient # На всякий случай:
ip link set eth0 down # The wired interface
ip link set eth1 up # The wireless interface
iwconfig eth1 essid "WirelessForAll"
dhclient -v eth1
```

Задание

- Посчитать количество учетных записей на компьютере состоящих в группе, название которой передано в качестве параметра скрипта.
- Написать рекурсивную функцию для вычисления n -го числа Фибоначчи. Вычислить с ее помощью число, номер которого передан параметром.
- Написать Bash-скрипт, который переворачивает расширение файла. Например, `a.pdf` превращается в `a.fdp`
- Измените нужные файлы таким образом, чтобы при входе в систему каждый пользователь получал в терминале приветствие вроде "Hi, USER!" (с конкретным именем пользователя). Если это `root`, напишите ему что-то особенное. Если пользователь в качестве оболочки использует не `bash`, то напишите ему что-нибудь страшное.

- Напишите Bash-скрипт, который выдает n штук случайных паролей, каждый длиной m . Числа m и n должны задаваться в параметрах. Если параметры не заданы, то должно выводиться сообщение об ошибке. Пароли могут содержать буквы латиницы в верхнем и нижнем регистре, а также цифры.
Указание. Случайную информацию можно брать из `/dev/random` и `/dev/urandom` (чем они отличаются?). Могут как-то помочь утилиты `tr` и `fold`.
- Напишите скрипт, позволяющий посмотреть содержимое архивов, переданных в качестве параметра скрипту. Должны обрабатываться следующие типы архивов: `tar`, `gz`, `bz2`, `zip`