Основы

Логические операторы И, ИЛИ и узнать код завершения программы

```
# Логическая И
ls -al && echo "Success!"
# Логическая ИЛИ
ls -al || echo "Fail!"
# Узнать код завершения.
# Если результат 0, значит программа завершилась успешно
echo $?
```

Потоки ввода-вывода

- STDIN(0) - стандартный поток ввода, номер потока 0
- STDOUT(1) - стандартный поток вывода, номер потока 1
- STDERR(2) - стандартный поток ошибок, номер потока 2

```
# Перенаправление ввода из файла file
program < file
# Перенаправление вывода (STDOUT) в файл file (запись с начала файла)
program > file
# Перенаправление вывода (STDOUT) в файл file в режиме дополнения файла
```

```
program >> file
# Перенаправление ошибок (STDERR) в файл file (запись с начала файла)
program 2> file
# Перенаправление ошибок (STDERR) в файл file в режиме дополнения файла
program 2>> file
# Перенаправление вывода (STDOUT) и ошибок (STDERR) в файл file (запись с начала файла)
program > file 2>&1
```

Конвейер - перенаправление ввода-вывода между процессами

```
ls -al | grep file
ls -al | grep -P '\.[cs]+'
cat /var/log/syslog | grep 'mysql' | grep -v 'file' | wc -1

# Вывод процессов
ps afx | grep ssh | grep -v pts
ps afx | grep [s]sh

cd /var/log
cat syslog | grep error | grep -v kernel | grep -i 'sqlite' | wc -l

ls -a | sort | wc -l
df -h | grep '/dev/sda'
```

Bash-скрипты

Переменные

- Переменные окружения

```
$PATH
$UID
$PWD
```

- Пользовательские переменные

```
var1=test
echo $var1
```

- Специальные переменные

```
$1...$9
$?
```

Примеры:

```
# Просмотр переменных окружения
printenv
# Просмотр значения переменной окружения
echo $PWD
echo $OLDPWD
# Задать переменную
var1=test
echo $var1
```

```
# '' одинарные кавычки означают текст. В одинарных нельзя указывать переменные
var2='adada ababa bububu'
echo $var2
var1='text in var1'
var2='text in var2'
# "" двойные ковычки означают текст, но в них можно указывать переменные
var3="$var2 $var1 bla"
echo $var3

# Помещение результата работы команды в переменную
var_ls=$(ls -al)
# или
var_ls=`ls -al`
echo $var_ls
```

Первый скрипт на Bash

```
cat > testscript
#!/bin/bash

directory=$1
hidden_count=$(ls -A $directory | grep '^\.' | wc -l)

echo "Hidden files in $directory found: $hidden_count"

# Способы запуска без прав на запуск
bash testscript /etc

# Способы запуска с правами на запуск
```

```
chmod +x testscript
./testscript /etc
/home/username/testscript /etc
```

Методы запуска скрипта

- Относительный путь: `./testscript`
- Абсолютный путь: `/home/username/testscript`
- Команда (Должен быть в `$PATH`): `testscript`
- Через команду bash: `bash testscript`
- Первые три варианта требуют шебанг(`#!/bin/bash` в начале файла скрипта) и права на исполнение (`chmod +x testscript`)

Однострочные скрипты

- Разделитель команд: `;`
- `apt update; apt upgrade; echo "Upgrade complete!"`

Циклы и ветвления

Условия if и ветвления

Синтаксис:

```
if [ выражение ]
        then
                Действия, если выражение истинно
        else
```

```
                Действия в противоположном случае
fi
```

Пример:

```
if [ -e file_name ]
        then
                echo "true"
        else
                echo "false"
fi
```

Варианты условий

Операции сравнения строк

- = или == возвращает true (истина), если строки равны
- != возвращает true (истина), если строки не равны
- -z возвращает true (истина), если строка пуста
- -n возвращает true (истина), если строка не пуста

Операции проверки файлов

- -e возвращает true (истина), если файл существует (exists)
- -d возвращает true (истина), если каталог существует (directory)

Операции сравнения целых чисел (наиболее используемые)

- -eq возвращает true (истина), если числа равны (equals)
- -ne возвращает true (истина), если числа не равны (not equal)

`man test` - посмотреть доступные операции в документации

Редактируем скрипт

```
# Открываем скрип в редакторе nano
nano testscript

# Содержимое редактора
#!/bin/bash

# Передаём в переменную путь к директории
directory=$1

if [ -n $directory ]
        then
                echo "Dir OK"
        else
                echo "No dir"
                # Завершаем скрипт, если не указана
директория, присваиваем код ошибки
                exit 1
fi

# Считаем количество скрытых файлов
hidden_count=$(ls -A $directory | grep '^\.' | wc -l)
```

```
# Выводим результат
echo "Hidden files in $directory found: $hidden_count"
```

Цикл for

Синтаксис:

```
for имя_переменной in значения
do
        тело_цикла
done
```

Примеры:

```
for h in {01..24}
do
        echo $h
done

for (( c=1; c<=5; c++ ))
do
        echo "Попытка номер $c"
done
```

Цикл While

Синтаксис:

```
while [ условие ]
do
```

```
        тело_цикла
done
```

Пример:

```
c=10

while [ $c -ge 0]
do
        echo "Test $c"
        let "c = c - 1"
done
```

Дополнительно

Определение типа команды в Bash

`type ls` - показать тип команды и стандартные параметры

`type -a ls` - к выводу добавляется информация о местоположении команды

`which ls` - показать путь к команде