# Linux and more.
# Bash scripting

Artem Trunov for Ozon Masters

# Что уже знаете про оболочку и CLI

- Запуск команд
- Как проверить код выхода программы
- Немного о переменных

# Что узнаем

- Больше о переменных
- Циклы
- Функции

# Мотивация

- Она слабая
- bash - это не тот инструмент, с которым работают дата-сайентисты, аналитики и проч.
- Но при работе на сервере постоянно сталкиваетесь с этим
  - Просто зафиксировать последовательность команд - пригодится в докер-файле позже на нашем курсе
- Не будем углубляться в тонкие детали, пробежимся по основным свойствам
  - Переменные
  - Циклы
  - Логические условия

# Hello, World!

```
#!/bin/bash
echo “Hello World!”
echo Hello $USER
```

1. `bash hello.sh`
2. `chmod +x hello.sh; ./hello.sh`
3. `/home/$USER/hello.sh`

./ (или полный путь /home/...) необходимо добавлять, когда программа не в папке из списка папок в $PATH (**echo $PATH**)

# Переменные

VAR1=4 #без  пробелов!
export var2=”I am var2” #export делает переменную доступной внутри другой прог.

echo $VAR1 “$VAR2” #переменные с пробелами квотируются обязательно!
echo $VAR1$VAR2 # так можно
echo ${VAR1}${var2} # но лучше так.
echo ${VAR1}или${var2}
echo “${VAR1} или ${var2}”

day=$(date +%A) #захват вывода в переменную day
day=`date +%A`

# source

- source - встроенная команда оболочки - выполняет скрипт без образования нового процесса
- Используется, например, для установки значений переменных.

Создайте файл **source-test.sh** со следующей строчкой:
**`export qq="ку-ку"`**

Запустите этот файл двумя способами и убедитесь, что после первого запуска переменная $qq в вашей оболочке не установлена, а после второго - установлена.

1. bash source-test.sh; echo $qq
2. bource source-test.sh; echo $qq

# Циклы

```
for x in $var2
do
echo $x
done
```

```
for x in $var2; do echo $x; echo ${x}_; done
```

```
for x in 1 2 3 4;
for x in {1..4};
for x in `seq 4`;
END=4; for ((i=1;i<=END;i++));
```

# Проверка кода выхода программы

```
$ ls ~/.ssh >/dev/null
$ echo $?
0

$ ls -al ку-ку
ls: cannot access 'ку-ку': No such file or directory
$ echo $?
2
$ echo $?
0

$ ls -al ку-ку 2>/dev/null
$ code=$?
```

# Проверка кода выхода программы

**if** ls -al ку-ку 2>/dev/null
**then** echo No error
**else** echo Error
**fi**

**If command** выполняется в соответствии с кодом выхода программы

**if** ls -al ку-ку 2>/dev/null; **then** echo No Error; **else** echo Erorr; **fi**

# Проверка кода выхода программы

ls -al ку-ку 2>/dev/null **&&** echo No error || echo Error

- Command1 && command2
  - Если команда 1 **успешна**, выполнить команду2
- Command1 || command2
  - Если команда 1 **неуспешна**, выполнить команду2
- Command1 && command2 || command3
  - Если команда 1 успешна, выполнить команду2, иначе выполнить команду3

ls -al ку-ку 2>/dev/null && mv ку-ку q-q || touch q-q
ls -al ку-ку 2>/dev/null && mv ку-ку q-q || touch q-q || echo Can not create

ls -al ку-ку 2>/dev/null && mv ку-ку q-q || touch q-q || echo Can not create && echo Can

# Логические условия

$ if **test** $code -eq 0
then echo No Error
else echo Error
fi
Error

```
artem@artem-ubuntu2:~$ if test $code -eq 0
> then echo No Error
> else echo Error
> fi
Error
```

# Логические условия: test vs. [ vs. [[

**if [** $code -eq 0 **]; then** echo success; **else** echo failure; fi

**if [[** $code -eq 0 **]]; then** echo success; **else** echo failure; fi

**test** *эквивалентно* **[** *и работает в новых и старых скриптах, POSIX*
**[[** *- форма с новыми свойствами, используемая в bash, zsh*

**if [** $code -eq 0 **] &&** [ $USER=artem ]; **then** echo success; **else** echo failure; fi

**if [[** $code -eq 0 **&&** $USER=artem ]]; **then** echo success; **else** echo failure; fi

https://unix.stackexchange.com/questions/306111/what-is-the-difference-between-the-bash-operators-vs-vs-vs

# Comparison operators

- Numeric
  - -eq, -ne, -lt, -gt, -le, -ge
- String
  - string1 == string2, string1 = string2, !=, >, <
  - -z string #True если длина нулевая
  - -n string #True если длина ненулевая

# Bash file operators

- -e #True если файл или директория существует
- -f #существует файл
- -d #существует папка
- -s # существует и размером 0
- file1 -nt file2 # file1 более новый чем file2 (по дате модификации)

...и еще много других. *man bash*

[ -f ку-ку ] && mv ку-ку q-q || touch q-q || echo Cant create && echo Can

# Функции

```
function f1 {
  [ -f ку-ку ] && mv ку-ку q-q || touch q-q || echo Cant create && echo Can
}
f1

function f2 {
  [ -f  $1 ] && mv $1 ${1}.back || touch ${1}.back || echo Cant create && echo Can
}
f2 ку-ку
```

$1, $2 - аргументы
$0, $*, $# - путь к программе, все агрументы, число аргументов

# Пример скрипта на bash

- Checker для домашки №2
  - См. Папку checker в datamove/practice-repo