

# Скрипты Bash

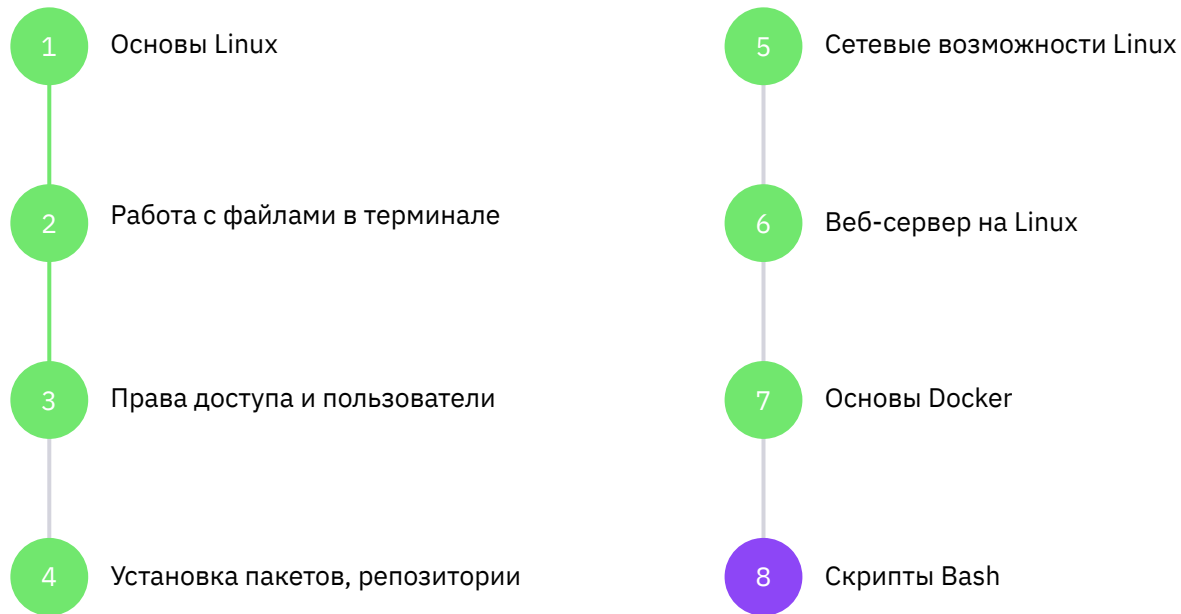




# Содержание урока










# План курса





## Что будет на уроке сегодня

-  Узнаем особенности оболочки bash
-  Изучим типы команд
-  Разберём потоки ввода-вывода
-  Научимся использовать конвейеры
-  Создадим свой bash-скрипт
-  Изучим условия в bash
-  Разберём циклы в bash



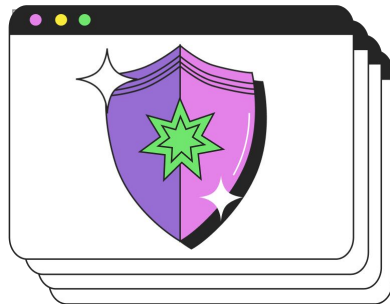
# Bash – оболочка и язык программирования



## Что такое оболочка



Пользователь



Оболочка (Bash)



Операционная  
система



# Bash — язык программирования

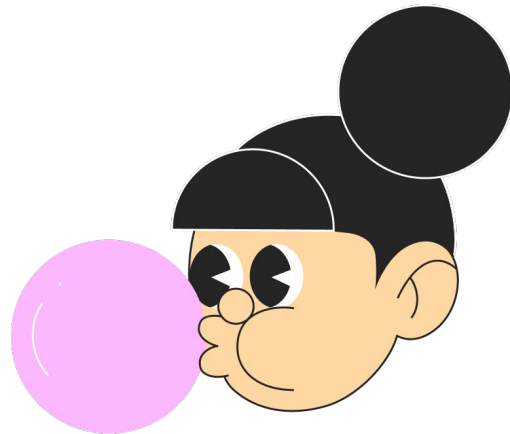
- Интерпретатор — `/bin/bash`
- Переменные
- Условия
- Циклы
- Функции
- Однострочные скрипты





## Код возврата (завершения) — exit code

- Код ошибки последней команды
- Проверить: `echo $?`
- Условная связка (И): `ls -al && echo "Success!"`
- Условная связка (ИЛИ): `ls -al || echo "Fail!"`



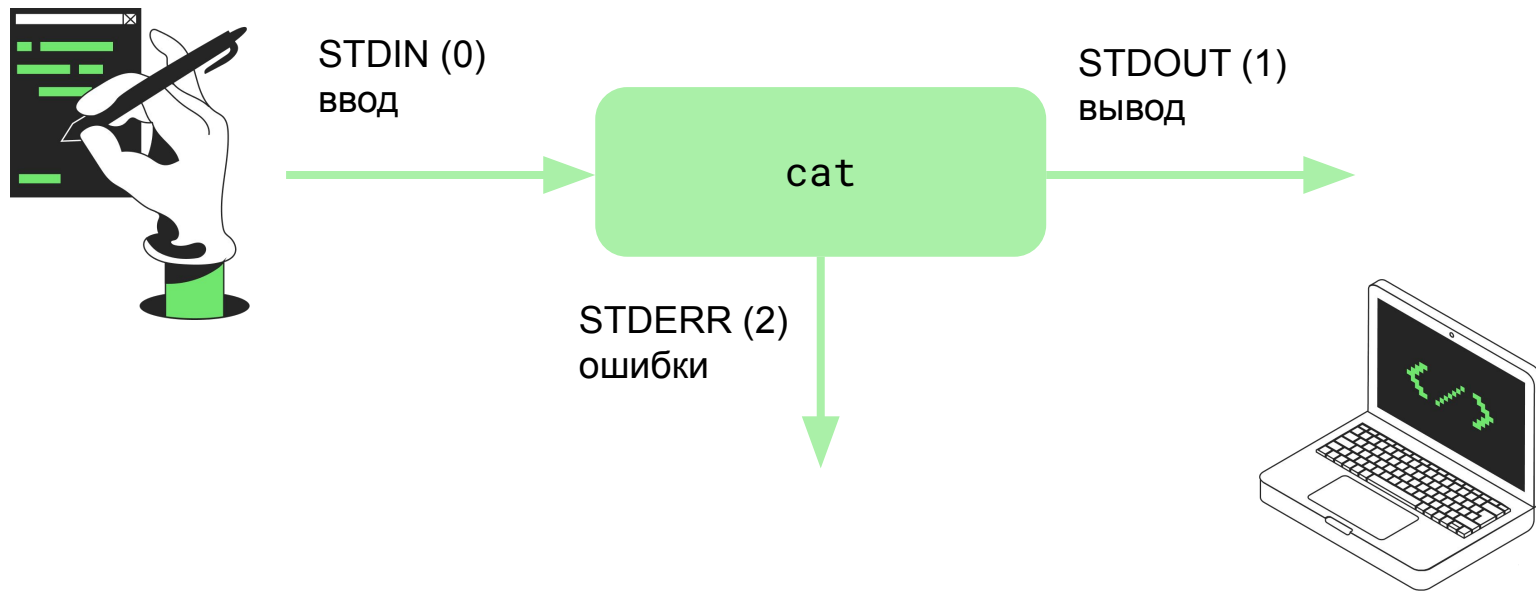




# Потоки ввода-вывода



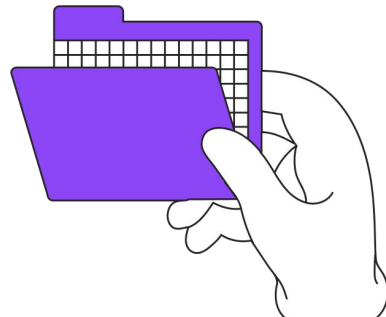
## Потоки ввода-вывода





## Перенаправление потоков ввода-вывода

- `program < file` – 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 2> 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 > file 2>&1` – перенаправление вывода (STDOUT) и ошибок (STDERR) в файл `file` (запись с начала файла)





## Конвейер (pipeline, pipe)

- Перенаправление ввода-вывода между процессами
- `ls -al | grep file`
- `ls -al | grep -P '\.[cs]+'`
- `cat /var/log/syslog | grep 'mysql' | grep -v 'file' | wc -l`



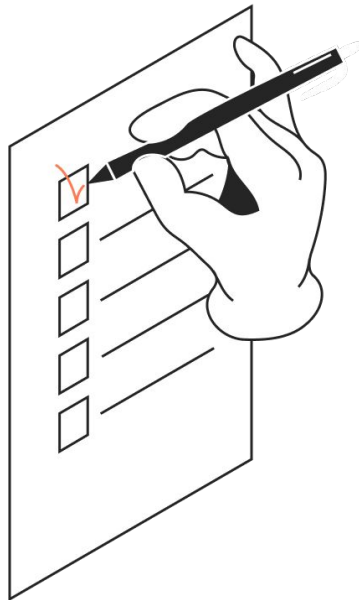


# Bash-скрипты



## Переменные и их классификация

- Переменные окружения
  - `$PATH`
  - `$UID`
  - `$PWD`
- Пользовательские переменные
  - `var1=test`
  - `echo $var1`
- Специальные переменные
  - `$1...$9`
  - `$?`

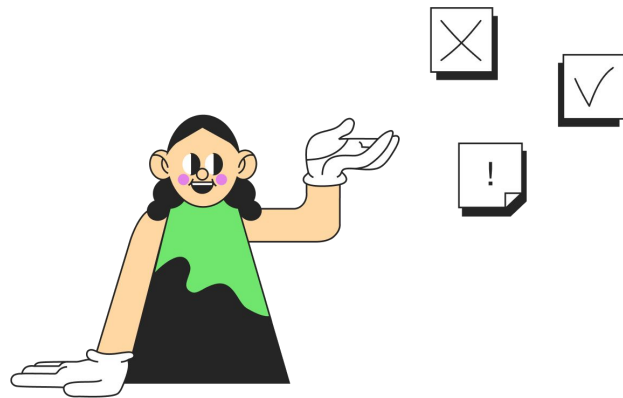




## Создаём первый скрипт на bash

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

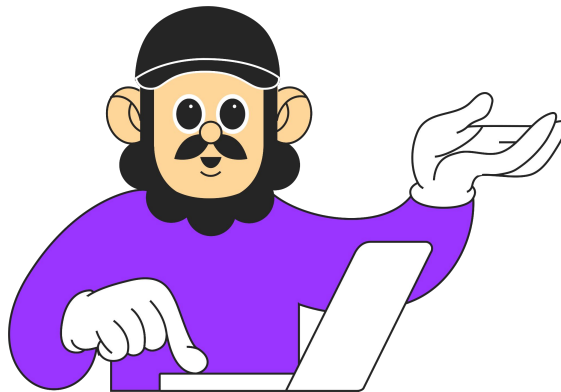
```
directory=$1  
hidden_count=$(ls -A $directory | grep '^\. ' | wc -l)  
  
echo "Hidden files in $directory found: $hidden_count"
```





## Методы запуска скрипта

- Относительный путь: `./testscript`
- Абсолютный путь: `/home/db/test/testscript`
- Команда (должен быть в `$PATH`): `testscript`
- Через команду `bash`: `bash testscript`
- Первые три варианта требуют шебанг и права на исполнение







## Однострочные скрипты

- Разделитель команд — ";"
- Удобны для выполнения в терминале
- Применимы все основные конструкции из bash
- `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)





# Цикл for

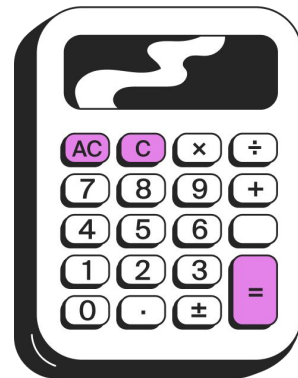
## Синтаксис:

```
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
```





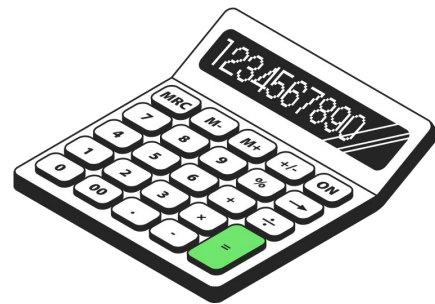
# Цикл while

## Синтаксис:

```
while [ условие ]  
do  
    Тело_цикла  
done
```

## Пример:

```
c=10  
  
while [ $c -ge 0 ]  
do  
    echo "Test"  
    let "c = c - 1"  
done
```



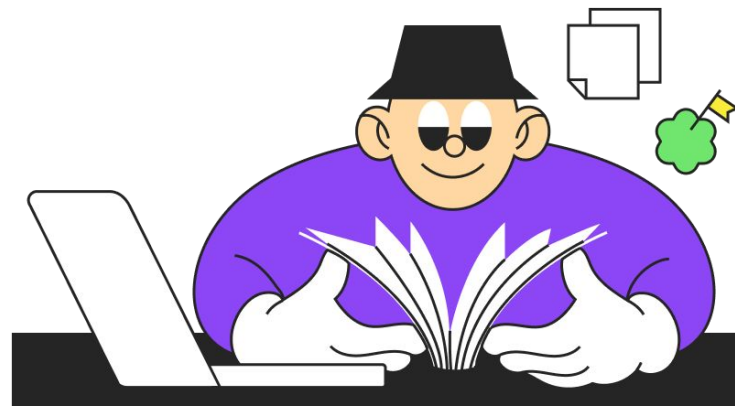


# Итоги занятия



## На этом уроке мы

- 📌 Познакомились с оболочкой bash
- 📌 Узнали, какие бывают команды
- 📌 Научились работать с потоками ввода-вывода
- 📌 Узнали, как использовать конвейер
- 📌 Познакомились с переменными в bash
- 📌 Создали простой скрипт
- 📌 Научились работать с условиями и циклами







Спасибо за внимание!