Операционные системы

Огородников Юрий Юрьевич yogorodnikov@gmail.com

UNIX::конструкции bash

- Код последней ошибки: $?

UNIX::конструкции bash

- Код последней ошибки: $?
- Смотрим справку по if: help if

UNIX::конструкции bash

- Код последней ошибки: $?
- Смотрим справку по if: help if
- Набираем пример в nano:

```
if cd ..; then
   echo Success
else
   echo Fail
fi
```

UNIX::конструкции bash

- Код последней ошибки: $?
- Смотрим справку по if: help if
- Набираем пример в nano:

```
if cd ..; then
   echo Success
else
   echo Fail
fi
```

- Команда test: справка man test

UNIX::конструкции bash

- Код последней ошибки: $?
- Смотрим справку по if: help if
- Набираем пример в nano:

```
if cd ..; then
   echo Success
else
   echo Fail
fi
```

- Команда test: справка man test
- Набираем пример в nano:

```
if test "$1" = "-h"; then
   echo "$0 [-h]"
   echo Hmm...
   exit 0
fi
```

UNIX::конструкции bash

- Код последней ошибки: $?
- Смотрим справку по if: help if
- Набираем пример в nano:

```
if cd ..; then
   echo Success
else
   echo Fail
fi
```

- Команда test: справка man test
- Набираем пример в nano:

```
if test "$1" = "-h"; then
   echo "$0 [-h]"
   echo Hmm...
   exit 0
fi
```

- Аналог test: команда [

```
if [ "$1" = "-h" ]; then
```

UNIX::конструкции bash

- Код последней ошибки: $?
- Смотрим справку по if: help if
- Набираем пример в nano:

```
if cd ..; then
   echo Success
else
   echo Fail
fi
```

- Команда test: справка man test
- Набираем пример в nano:

```
if test "$1" = "-h"; then
   echo "$0 [-h]"
   echo Hmm...
   exit 0
fi
```

- Аналог test: команда [

```
if [ "$1" = "-h" ]; then
```

- Задача: найти способ, как проверить и с -h, и с --help

UNIX::конструкции bash

- Код последней ошибки: $?
- Смотрим справку по if: help if
- Набираем пример в nano:

```
if cd ..; then
   echo Success
else
   echo Fail
fi
```

- Команда test: справка man test
- Набираем пример в nano:

```
if test "$1" = "-h"; then
   echo "$0 [-h]"
   echo Hmm...
   exit 0
fi
```

- Аналог test: команда [

```
if [ "$1" = "-h" ]; then
```

- Задача: найти способ, как проверить и с -h, и с --help
- if ["$1" = "-h" -o "$1" = "--help"]; then

UNIX::PATH,which,циклы

- Задача: сделать аналог утилиты which
- Пишем справку:

```
if [ "$1" = "-h" -o "$1" = "--help" ]; then
   echo "$0 program"
   echo Which returns the pathname of the file which would be executed in
the current environment
   exit 0
fi
```

UNIX::PATH,which,циклы

- Задача: сделать аналог утилиты which
- Пишем справку:

```
if [ "$1" = "-h" -o "$1" = "--help" ]; then
   echo "$0 program"
   echo Which returns the pathname of the file which would be executed in
the current environment
   exit 0
fi
```

- Смотрим содержимое PATH: echo $PATH

UNIX::PATH,which,циклы

- Задача: сделать аналог утилиты which
- Пишем справку:

```
if [ "$1" = "-h" -o "$1" = "--help" ]; then
   echo "$0 program"
   echo Which returns the pathname of the file which would be executed in
the current environment
   exit 0
fi
```

- Смотрим содержимое PATH: echo $PATH
- Смотрим help for

UNIX::PATH,which,циклы

- Задача: сделать аналог утилиты which
- Пишем справку:

```
if [ "$1" = "-h" -o "$1" = "--help" ]; then
   echo "$0 program"
   echo Which returns the pathname of the file which would be executed in
the current environment
   exit 0
fi
```

- Смотрим содержимое PATH: echo $PATH
- Смотрим help for
- Вывести числа от 1 до 10 на экран:

```
for i in {1..10}; do
   echo $i
done
```

UNIX::PATH,which,циклы

- Задача: сделать аналог утилиты which
- Пишем справку:

```
if [ "$1" = "-h" -o "$1" = "--help" ]; then
   echo "$0 program"
   echo Which returns the pathname of the file which would be executed in
the current environment
   exit 0
fi
```

- Смотрим содержимое PATH: echo $PATH
- Смотрим help for
- Вывести числа от 1 до 10 на экран:

```
for i in {1..10}; do
   echo $i
done
```

- Поставить IFS=:

UNIX::PATH,which,циклы

- Задача: сделать аналог утилиты which
- Пишем справку:

```
if [ "$1" = "-h" -o "$1" = "--help" ]; then
   echo "$0 program"
   echo Which returns the pathname of the file which would be executed in
the current environment
   exit 0
fi
```

- Смотрим содержимое PATH: echo $PATH
- Смотрим help for
- Вывести числа от 1 до 10 на экран:

```
for i in {1..10}; do
   echo $i
done
```

- Поставить IFS=:
- Затем написать:

```
for path in $PATH; do
   echo $path
done
```

UNIX::PATH,which,циклы

- Смотрим содержимое PATH: echo $PATH
- Смотрим help for
- Вывести числа от 1 до 10 на экран:

```
for i in {1..10}; do
   echo $i
done
```

- Поставить IFS=:
- Затем написать:

```
for path in $PATH; do
   echo $path
done
```

- Проверить, есть ли нужный нам файл в $path (искать в справке по [или по test)

UNIX::PATH,which,циклы

- Проверить, есть ли нужный нам файл в $path (искать в справке по [или по test)
- Итоговое тело цикла:

```
filepath="$path/$1"
if [ -f "$filepath" -a -x "$filepath" ]; then
   echo "$filepath"
   exit 0
fi
```

UNIX::PATH,which,циклы

- Проверить, есть ли нужный нам файл в $path (искать в справке по [или по test)
- Итоговое тело цикла:

```
filepath="$path/$1"
if [ -f "$filepath" -a -x "$filepath" ]; then
   echo "$filepath"
   exit 0
fi
```

- Самостоятельно: поиск n-го члена последовательности Фибоначчи

UNIX::PATH,which,циклы

- Проверить, есть ли нужный нам файл в $path (искать в справке по [или по test)
- Итоговое тело цикла:

```
filepath="$path/$1"
if [ -f "$filepath" -a -x "$filepath" ]; then
   echo "$filepath"
   exit 0
fi
```

- Самостоятельно: поиск n-го члена последовательности Фибоначчи
- Цикл for

```
for (( i=1; i<10; i++ ))
do
   echo $i
done
```

UNIX::PATH,which,циклы

- Проверить, есть ли нужный нам файл в $path (искать в справке по [или по test)
- Итоговое тело цикла:

```
filepath="$path/$1"
if [ -f "$filepath" -a -x "$filepath" ]; then
   echo "$filepath"
   exit 0
fi
```

- Самостоятельно: поиск n-го члена последовательности Фибоначчи
- Цикл for

```
for (( i=1; i<10; i++ ))
do
   echo $i
done
```

- Цикл while:

```
i=0
while [ $i -lt 10 ]; do
   ... done
```

UNIX::PATH,which,циклы

- Проверить, есть ли нужный нам файл в $path (искать в справке по [или по test)
- Итоговое тело цикла:

```
filepath="$path/$1"
if [ -f "$filepath" -a -x "$filepath" ]; then
   echo "$filepath"
   exit 0
fi
```

- Самостоятельно: поиск n-го члена последовательности Фибоначчи
- Цикл for

```
for (( i=1; i<10; i++ ))
do
   echo $i
done
```

- Цикл while:

```
i=0
while [ $i -lt 10 ]; do
   ... done
```

- Считать арифметическое выражение:

```
let "n+=$n-1"
n=$(($n+1))
```

UNIX::If comparison

Сравнение чисел

- n1 -eq n2 # Возвращает истинное значение, если n1 равно n2;
- n1 -ge n2 # Возвращает истинное значение, если n1 больше или равно n2;
- n1 -gt n2 # Возвращает истинное значение, если n1 больше n2;
- n1 -le n2 # Возвращает истинное значение, если n1меньше или равно n2;
- n1 -lt n2 # Возвращает истинное значение, если n1 меньше n2;
- n1 -ne n2 # Возвращает истинное значение, если n1 не равно n2.

UNIX::If comparison

Сравнение строк

- str1 = str2 # Проверяет строки на равенство, возвращает истину, если строки идентичны;
- str1 != str2 # Возвращает истину, если строки не идентичны;
- str1 < str2 # Возвращает истину, если str1 меньше, чем str2;
- str1 > str2 # Возвращает истину, если str1 больше, чем str2;
- -n str1 # Возвращает истину, если длина str1 больше нуля;
- -z str1 # Возвращает истину, если длина str1 равна нулю.

UNIX::If comparison

Проверки файлов

- -d file # проверяет, существует ли файл, и является ли он директорией;
- -e file # проверяет, существует ли файл;
- -f file # проверяет, существует ли файл, и является ли он файлом;
- -r file # проверяет, существует ли файл, и доступен ли он для чтения;
- -s file # проверяет, существует ли файл, и не является ли он пустым;
- -w file # проверяет, существует ли файл, и доступен ли он для записи;
- -x file # проверяет, существует ли файл, и является ли он исполняемым;
- file1 -nt file2 # проверяет, новее ли file1, чем file2;
- file1 -ot file2 # проверяет, старше ли file1, чем file2;
- -O file # проверяет, существует ли файл, и является ли его владельцем текущий пользователь;
- -G file # проверяет, существует ли файл, и соответствует ли его идентификатор группы идентификатору группы текущего пользователя.

UNIX::массивы, функции и полезные фишки

Массивы

```
arr=( Hello World ) #инициализация массива
```

UNIX::массивы, функции и полезные фишки

Массивы

arr=(Hello World) #инициализация массива
arr[0]=Hello #альтернативная
arr[3]=World #инициализация массива

UNIX::массивы, функции и полезные фишки

Массивы

```
arr=( Hello World ) #инициализация массива
arr[0]=Hello #альтернативная
arr[3]=World #инициализация массива
echo ${arr[0]} ${arr[1]} #вывод элементов массива
```

UNIX::массивы, функции и полезные фишки

Массивы

arr=(Hello World) #инициализация массива
arr[0]=Hello #альтернативная
arr[3]=World #инициализация массива
echo ${arr[0]} ${arr[1]} #вывод элементов массива

- ${arr[*]}, ${arr[@]} #все записи в массиве;
- ${!arr[*]}, ${!arr[@]} #все индексы в массива;
- ${#arr[*]}, ${#arr[@]} #количество записей в массиве;
- ${#arr[0]} #длина первой записи в массиве.

UNIX::массивы, функции и полезные фишки

Функции

Определение функции:

```
function func1() {
echo "This is a function"
}
```

UNIX::массивы, функции и полезные фишки

Функции

Определение функции:

```
function func1() {
echo "This is a function"
}
```

Доступ к аргументам внутри функции: через $1, $2 и т.д.

UNIX::массивы, функции и полезные фишки

Функции

Определение функции:
function func1() {
echo "This is a function"
}
Доступ к аргументам внутри функции: через $1, $2 и т.д.
Возврат из функции: return exit_code

UNIX::массивы, функции и полезные фишки

Функции

Определение функции:
function func1() {
echo "This is a function"
}
Доступ к аргументам внутри функции: через $1, $2 и т.д.
Возврат из функции: return exit_code
Вызов функции:
func1 # без аргументов
func1 $USER $SOME_VAR # с аргументами

UNIX::массивы, функции и полезные фишки

Полезные фишки

- basename file #печатает последний компонент в пути к файлу;
- name=file_0456.mp4
  echo ${name##file_} #обрезать префикс file_
- name=file_0456.mp4
  echo ${name%%.mp4} #обрезать суффикс .mp4