Операционные системы

Огородников Юрий Юрьевич
yogorodnikov@gmail.com

UNIX

- Создаем helloArgs.sh: #/bin/bash
  echo $1 $2 $3

UNIX

- Создаем helloArgs.sh: #/bin/bash
  echo $1 $2 $3
- Выставляем право на исполнение: chmod +x helloArgs.sh

UNIX

- Создаем helloArgs.sh: #/bin/bash
  echo $1 $2 $3
- Выставляем право на исполнение: chmod +x helloArgs.sh
- Запускаем ./helloArgs.sh hi 1 2 3

UNIX

- Создаем helloArgs.sh: #/bin/bash
  echo $1 $2 $3
- Выставляем право на исполнение: chmod +x helloArgs.sh
- Запускаем ./helloArgs.sh hi 1 2 3
- Попробуем передать 10-й аргумент и вывести его через $10

UNIX

- Создаем helloArgs.sh: #/bin/bash
  echo $1 $2 $3
- Выставляем право на исполнение: chmod +x helloArgs.sh
- Запускаем ./helloArgs.sh hi 1 2 3
- Попробуем передать 10-й аргумент и вывести его через $10
- Доступ к 10-му аргументу: ${10}

UNIX

- Создаем новую директорию: mkdir test, и переходим в нее

UNIX

- Создаем новую директорию: mkdir test, и переходим в нее В данном каталоге создаем следующую структуру файлов/директорий:

  1.txt

  2.txt

  1900

  1901

  ...

  1999

  2000

UNIX

- Создаем новую директорию: mkdir test, и переходим в нее В данном каталоге создаем следующую структуру файлов/директорий:

  1.txt

  2.txt

  1900

  1901

  ...

  1999

  2000

- Создать файл: либо echo qwe > 1.txt, либо touch 2.txt
- Создать директорию 1, в которой будет директория 2: mkdir -p 1/2

UNIX

- Создаем новую директорию: mkdir test, и переходим в нее В данном каталоге создаем следующую структуру файлов/директорий:

  1.txt

  2.txt

  1900

  1901

  ...

  1999

  2000
- Создать файл: либо echo qwe $>$ 1.txt, либо touch 2.txt
- Создать директорию 1, в которой будет директория 2: mkdir -p 1/2
- Создать директории с именами от 1900 до 2000: mkdir {1900..2000}

UNIX

- Создаем новую директорию: mkdir test, и переходим в нее В данном каталоге создаем следующую структуру файлов/директорий:

  1.txt

  2.txt

  1900

  1901

  ...

  1999

  2000
- Создать файл: либо echo qwe > 1.txt, либо touch 2.txt
- Создать директорию 1, в которой будет директория 2: mkdir -p 1/2
- Создать директории с именами от 1900 до 2000: mkdir {1900..2000}
- Удалить все созданные файлы и папки: попробовать rm *

UNIX

- Создаем новую директорию: mkdir test, и переходим в нее В данном каталоге создаем следующую структуру файлов/директорий:

  1.txt

  2.txt

  1900

  1901

  ...

  1999

  2000
- Создать файл: либо echo qwe > 1.txt, либо touch 2.txt
- Создать директорию 1, в которой будет директория 2: mkdir -p 1/2
- Создать директории с именами от 1900 до 2000: mkdir {1900..2000}
- Удалить все созданные файлы и папки: попробовать rm *
- Создадим файлы 1.txt, 2.txt, и создадим файл с именем -rf (echo>-rf), и запускаем заново rm *

UNIX

- Создаем новую директорию: mkdir test, и переходим в нее В данном каталоге создаем следующую структуру файлов/директорий:

  1.txt

  2.txt

  1900

  1901

  ...

  1999

  2000

- Создать файл: либо echo qwe > 1.txt, либо touch 2.txt
- Создать директорию 1, в которой будет директория 2: mkdir -p 1/2
- Создать директории с именами от 1900 до 2000: mkdir {1900..2000}
- Удалить все созданные файлы и папки: попробовать rm *
- Создадим файлы 1.txt, 2.txt, и создадим файл с именем -rf (echo>-rf), и запускаем заново rm *
- Как удалить все правильно: в UNIX все ключи нужно передавать в программу **до ключа --**. Пишем в консоли: rm -- *

UNIX

- Создаем новую директорию: mkdir test, и переходим в нее В данном каталоге создаем следующую структуру файлов/директорий:

  1.txt

  2.txt

  1900

  1901

  ...

  1999

  2000

- Создать файл: либо echo qwe > 1.txt, либо touch 2.txt
- Создать директорию 1, в которой будет директория 2: mkdir -p 1/2
- Создать директории с именами от 1900 до 2000: mkdir {1900..2000}
- Удалить все созданные файлы и папки: попробовать rm *
- Создадим файлы 1.txt, 2.txt, и создадим файл с именем -rf (echo>-rf), и запускаем заново rm *
- Как удалить все правильно: в UNIX все ключи нужно передавать в программу **до ключа --**. Пишем в консоли: rm -- *
- Скрытые файлы: .dat, например

UNIX::переменные

- Команда set — переменные запущенной командной оболочки.

UNIX::переменные

- Команда set — переменные запущенной командной оболочки.
- Более привычно: env

UNIX::переменные

- Команда set — переменные запущенной командной оболочки.
- Более привычно: env
- Вывод значений переменных: echo $USER

UNIX::переменные

- Команда set — переменные запущенной командной оболочки.
- Более привычно: env
- Вывод значений переменных: echo $USER
- Узнать текущую рабочую директорию: echo $PWD

UNIX::переменные

- Команда set — переменные запущенной командной оболочки.
- Более привычно: env
- Вывод значений переменных: echo $USER
- Узнать текущую рабочую директорию: echo $PWD
- Вывести произвольное число: echo $RANDOM

UNIX::переменные

- Команда set — переменные запущенной командной оболочки.
- Более привычно: env
- Вывод значений переменных: echo $USER
- Узнать текущую рабочую директорию: echo $PWD
- Вывести произвольное число: echo $RANDOM
- Двойные кавычки: вывод текста с подстановкой значений переменных: echo "Hello, $USER!"

UNIX::переменные

- Команда set — переменные запущенной командной оболочки.
- Более привычно: env
- Вывод значений переменных: echo $USER
- Узнать текущую рабочую директорию: echo $PWD
- Вывести произвольное число: echo $RANDOM
- Двойные кавычки: вывод текста с подстановкой значений переменных: echo "Hello, $USER!"
- Использование одинарных кавычек: ничего не будет заменено: 'Hello, $USER!'

UNIX::переменные

- Команда set — переменные запущенной командной оболочки.
- Более привычно: env
- Вывод значений переменных: echo $USER
- Узнать текущую рабочую директорию: echo $PWD
- Вывести произвольное число: echo $RANDOM
- Двойные кавычки: вывод текста с подстановкой значений переменных: echo "Hello, $USER!"
- Использование одинарных кавычек: ничего не будет заменено: 'Hello, $USER!'
- Вывести USER во множественном числе: echo ${USER}s

UNIX::переменные

- Команда set — переменные запущенной командной оболочки.
- Более привычно: env
- Вывод значений переменных: echo $USER
- Узнать текущую рабочую директорию: echo $PWD
- Вывести произвольное число: echo $RANDOM
- Двойные кавычки: вывод текста с подстановкой значений переменных: echo "Hello, $USER!"
- Использование одинарных кавычек: ничего не будет заменено: 'Hello, $USER!'
- Вывести USER во множественном числе: echo ${USER}s
- Создать свою переменную: set Admin=hog, или Admin=hog

UNIX::переменные

- Команда set — переменные запущенной командной оболочки.
- Более привычно: env
- Вывод значений переменных: echo $USER
- Узнать текущую рабочую директорию: echo $PWD
- Вывести произвольное число: echo $RANDOM
- Двойные кавычки: вывод текста с подстановкой значений переменных: echo "Hello, $USER!"
- Использование одинарных кавычек: ничего не будет заменено: 'Hello, $USER!'
- Вывести USER во множественном числе: echo ${USER}s
- Создать свою переменную: set Admin=hog, или Admin=hog
- В терминале запускаем bash, смотрим env

UNIX::переменные

- Команда set — переменные запущенной командной оболочки.
- Более привычно: env
- Вывод значений переменных: echo $USER
- Узнать текущую рабочую директорию: echo $PWD
- Вывести произвольное число: echo $RANDOM
- Двойные кавычки: вывод текста с подстановкой значений переменных: echo "Hello, $USER!"
- Использование одинарных кавычек: ничего не будет заменено: 'Hello, $USER!'
- Вывести USER во множественном числе: echo ${USER}s
- Создать свою переменную: set Admin=hog, или Admin=hog
- В терминале запускаем bash, смотрим env
- Вводим команду export Admin=Putin, смотрим env

UNIX::переменные

- Команда set — переменные запущенной командной оболочки.
- Более привычно: env
- Вывод значений переменных: echo $USER
- Узнать текущую рабочую директорию: echo $PWD
- Вывести произвольное число: echo $RANDOM
- Двойные кавычки: вывод текста с подстановкой значений переменных: echo "Hello, $USER!"
- Использование одинарных кавычек: ничего не будет заменено: 'Hello, $USER!'
- Вывести USER во множественном числе: echo ${USER}s
- Создать свою переменную: set Admin=hog, или Admin=hog
- В терминале запускаем bash, смотрим env
- Вводим команду export Admin=Putin, смотрим env
- Возвращаемся в первый bash (ctrl+D), проверяем echo $Admin

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))
```