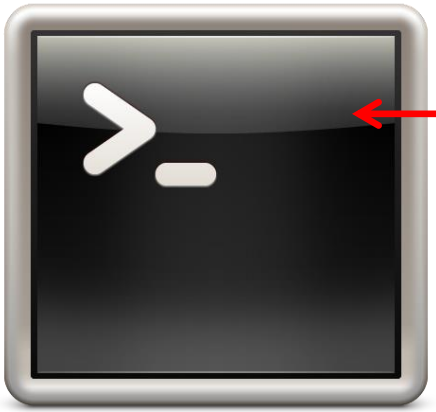
Программирование на bash

Программирование на bash



Оболочка (shell) – интерпретатор команд

sh (Bourne shell, 1977) – стандарт и доступна почти в любом дистрибутиве

bash (Bourne again shell, 1989) – усовершенствованная и модернизированная вариациая sh

Также существуют: csh, tcsh, zsh, …

Первая программа

Задание:

Создайте файл с названием “*my_first_script.sh*”.

Внесите в него следующий текст:

#! /bin/bash
echo “Hello, world!”

Первая программа

Для запуска программы выполните следующие команды:

chmod +x *my_first_script.sh*
./my_first_script.sh

Переменные

Имя:

- буквы, цифры, “_”;
- имя не может начинаться с цифры.

Значения:

- числа;
- строки (если есть пробелы, то в кавычках);
- отдельные символы.

Переменные

Присваивание значения переменной:

<имя>=<значение>

Считывание значения переменной:

$<имя> *или* ***${***<имя>***}***

Пример:

```
path=~/Docs
path2=$path/file1.txt          # path2=~/Docs/file1.txt
echo “Path is $path2”          # Path is ~/Docs/file1.txt
echo “Path is ${path}2”        # Path is ~/Docs2
```

Переменные

Присваивание значения переменной:

<имя>=<значение>

Считывание значения переменной:

$<имя> *или* ***${***<имя>***}***

Пример:

```
path=~/Docs
path2=$path/file1.txt          # path2=~/Docs/file1.txt
echo “Path is $path2”          # Path is ~/Docs/file1.txt
echo “Path is ${path}2”        # Path is ~/Docs2
```

Переменные

Использование знака «$» в текстовых строках:

var1=‘Price: $78’

или

var1=“Price: \$78”

Переменные

Считать значение переменной с клавиатуры:

read переменная

Пример:

read var1

Аргументы

Передача аргументов скрипту:
./script.sh arg1 arg2 arg3 …

Обработка внутри скрипта:
$1 первый аргумент
$2 второй аргумент
…

$0 имя скрипта
$# количество аргументов

Пример:

var=“Первый аргумент равен $1”
echo “Всего было передано $# аргументов

Аргументы

Задание:

Напишите скрипт на bash, который принимает на вход два аргумента и выводит на экран строку следующего вида:

Arguments are: $1=первый_аргумент $2=второй_аргумент

Например, если ваш скрипт называется ./script.sh, то при запуске его ./script.sh one two на экране должно появиться:

Arguments are: $1=one $2=two

а при запуске ./script.sh three four будет:

Arguments are: $1=three $2=four

Арифметические операции

Синтаксис:

let “переменная = выражение”

Пример:

Let “c = 1 + 1”
Let “c = a + b”

Операции:

+, -, /, *	Стандартные
%	Остаток от деления
**	Возведение в степень

Сокращенная форма:
let “a=a+b” эквивалентно let “a+=b”

Ветвление

Синтаксис:

```
If   [[ условие1 ]]
then
        # действия, если условие1 истинно

elseif   [[ условие2 ]]
then
        # действия, если условие2 истинно

else
        # действия, если все условия ложны
fi
```

Ветвление

Условия (строки):

-z <строка>	Строка пуста
-n <строка>	Строка не пуста
<стр1> **==** <стр2>	Строки равны
<стр1> **!=** <стр2>	Строки не равны
<стр1> **<** <стр2>	Меньше
<стр1> **>** <стр2>	Больше

Ветвление

Условия (числа):

-eq	Равно
-ne	Не равно
-lt	Меньше
-le	Меньше или равно
-gt	Больше
-ge	Больше или равно

Ветвление

Условия (файлы):

-e <путь>	Путь существует
-f <путь>	Это файл
-d <путь>	Это директория
-s <путь>	Размер файла больше 0
-x <путь>	Файл исполняемый

Логические операторы:

!	Отрицание
&&	Логическое «И»
||	Логическое «Или»

Ветвление

Синтаксис:

```
case переменная in
значение1)
        # действия, если переменная==значение1
        ;;
значение2)
        # действия, если переменная==значение2
        ;;
*)
        # действия в остальных случаях
esac
```

Ветвление

Задание:

Напишите скрипт на bash, который ожидает ввода с клавиатуры один аргумент (целое число от 0 до бесконечности), который будет обозначать число студентов в аудитории. В зависимости от значения числа нужно вывести разные сообщения.

Соответствие входа и выхода должно быть таким:

Вход	**Выход**
0	No students
1	1 student
2	2 students
3	3 students
4	4 students
5 и больше	A lot of students

Циклы

Цикл с параметром

Синтаксис:

for переменная **in** список_значений
do
	# действия, каждый раз переменная принимает
	# следующее значение из списка
done

Циклы

Цикл с предусловием

Синтаксис:

while [[условие]]
do
        # действия, пока условие истинно
done

break	Досрочный выход из цикла
continue	Переход к следующему шагу

Циклы

Задание:

Напишите скрипт на bash, который будет определять в какую возрастную группу попадают пользователи. При запуске скрипт должен вывести сообщение "enter your name:" и ждать от пользователя ввода имени (используйте read, чтобы прочитать его). Когда имя введено, то скрипт должен написать "enter your age:" и ждать ввода возраста (опять нужен read). Когда возраст введен, скрипт пишет на экран "<Имя>, your group is <группа>", где <группа> определяется на основе возраста по следующим правилам:

- младше либо равно 16: "child",
- от 17 до 25 (включительно): "youth",
- старше 25: "adult".

После этого скрипт опять выводит сообщение "enter your name:" и всё начинается по новой. Если в какой-то момент работы скрипта будет введено пустое имя или возраст 0, то скрипт должен написать на экран "bye" и закончить свою работу.

Внешние программы

Синтаксис:
переменная=`программа`

Пример:
a=`echo “test”`
files=`ls ~`

Внешние программы

Код возврата:
0 корректное завершение
не 0 в процессе работы были ошибки

Узнать код:
$?
Выйти с кодом:
exit *код*

Пример:
touch file.txt
echo $?

Внешние программы

Проверка кода возврата:

```
if `программа`
then
# действия, если код 0
else
# действия, если код не 0
fi
```

Функции

Задание функции:

имя_функции ()
{
действия
}

Использование функции:
…
имя_функции
…

Функции

Функции с параметрами:

имя_функции ()
{
# действия с **$1, $2, … , $#**
}

Использование функции:

…
имя_функции **аргумент1 аргумент2 ...**
…

Функции

В функции можно определить **глобальные** и **локальные** переменные:

```
имя_функции ()
{
var_global=1
local var_local=1
}
```

При использовании:

```
имя_функции
echo $var_global # выведет 1
echo $var_local # ничего не выведет
```

Функции

Компактная запись:
имя_функции () { действ1**;** действ2**;** }

Актуально и в других конструкциях:

if [[$var==”test ”]]; then
…
for i in 1 2 3 4 5; do
…

Функции

Задание:

Напишите скрипт на bash, который будет искать наибольший общий делитель (НОД, greatest common divisor, GCD) двух чисел.
После ввода чисел скрипт считает их НОД и выводит на экран сообщение **"GCD is <посчитанное значение>"**, например, для чисел 15 и 25 это будет "GCD is 5". После этого скрипт опять входит в режим ожидания двух натуральных чисел. Если в какой-то момент работы пользователь ввел вместо этого пустую строку, то нужно написать на экран "**bye**" и закончить свою работу.
Вычисление НОД несложно реализовать с помощью алгоритма Евклида. Вам нужно написать функцию gcd, которая принимает на вход два аргумента (назовем их **M** и **N**). **Если аргументы равны**, то мы нашли НОД -- он равен **M** (или **N**), нужно выводить соответствующее сообщение на экран (см. выше). Иначе нужно сравнить аргументы между собой. Если **M больше N**, то запускаем ту же функцию gcd, но в качестве первого аргумента передаем (**M**-**N**), а в качестве второго **N**. Если же наоборот, **M** меньше **N**, то запускаем функцию gcd с первым аргументом **M**, а вторым (**N**-**M**).