Цель работы

- Изучение основных конструкций командной оболочки Bash.Знакомство со Bash-скриптами.

Теоретический материал

Что такое Bash

Bash (Bourne Again Shell) — это мощная командная оболочка Unix, которая используется для выполнения различных задач в терминале. Bash предоставляет интерактивный интерфейс, в котором пользователи могут вводить команды, а затем получать результаты. Она также поддерживает скрипты оболочки, которые представляют собой текстовые файлы, содержащие последовательность команд Bash для автоматизации задач.

Bash является оболочкой по умолчанию в популярных дистрибутивах Linux, таких как Ubuntu, Red Hat Enterprise Linux, CentOS. Это делает ее неотъемлемой частью экосистемы Linux.

Переменные

Переменные используются для хранения данных и значений, которые можно использовать в скриптах и командах. Чтобы создать переменную, присвойте ей значение без объявления типа данных (без пробелов):

```
name="Алексей"
```

Для доступа к значению переменной используйте префикс $:

```
echo $name
```

Типы переменных:

- локальные — видимы только в текущем скрипте или функции;
- переменные окружения — доступны во всех процессах и скриптах.

Особые переменные:

- $0 - имя текущего скрипта;
- $1, $2, ... - аргументы, переданные скрипту;
- $? - код возврата последней выполненной команды;
- $@ - все аргументы, переданные скрипту;
- $# - количество переданных аргументов.

Переменные являются инструментом, позволяющим хранить и манипулировать данными для автоматизации задач. С их помощью можно хранить имена файлов, пути к каталогам или результаты команд, а затем использовать эту информацию повторно в скрипте. Это устраняет необходимость дублирования значений, а заодно минимизирует вероятность ошибок. Кроме того, переменные позволяют легко передавать данные между различными частями скрипта, способствуя написанию модульного и организованного кода.

Типы данных

Bash не имеет строгой системы типов данны. Наиболее часто используемые типы:

- целые числа — целые числа без десятичной точки, например 10 или -5;
- вещественные числа — числа с десятичной точкой, например 3.14 или -0.5;
- строки — последовательности символов, заключенные в кавычки, например "Hello, world" или 'Bash scripting';
- массивы — коллекции элементов, доступ к которым осуществляется по индексу;
- переменные окружения — переменные, которые содержат информацию о текущем состоянии оболочки, например $HOME (домашний каталог пользователя) или $PATH (путь к исполняемым файлам).

Bash также поддерживает преобразование типов с помощью команд, таких как echo и expr. Например, можно преобразовать целое число в строку с помощью echo $((10)), а строку в целое число — с помощью expr 10 + 5.

Ветвления

Ветвление в Bash позволяет управлять потоком выполнения скрипта в зависимости от выполнения определенных условий. В Bash для ветвления используются операторы `if`, `elif` и `else`.

Оператор if:

```
if [ condition ]; then # пробелы обязательны
    # Выполнить, если условие истинно
fi
```

Оператор elif:

```
if [ condition1 ]; then
  # Выполнить, если condition1 истинно
elif [ condition2 ]; then
    # Выполнить, если condition2 истинно
fi
```

Оператор else:

```
if [ condition1 ]; then
  # Выполнить, если condition1 истинно
elif [ condition2 ]; then
    # Выполнить, если condition2 истинно
else
    # Выполнить, если все предыдущие условия ложны
fi
```

В условиях Bash можно использовать операторы сравнения

```
-lt  <  less than
-gt  >  greater than
-le  <=  less or equal
-ge  >=  greater or equal
-eq  =  equal
-nq  !=  not equal
```

Например:

```
if [ $age -ge 18 ]; then
    echo "Вы можете голосовать"
else
    echo "Вы не можете голосовать"
fi
```

Оператор case:

```
case $variable in
    pattern1)
        # Выполнить, если переменная соответствует шаблону 1
        ;;
    pattern2)
        # Выполнить, если переменная соответствует шаблону 2
        ;;
    *)
        # Выполнить, если переменная не соответствует ни одному шаблону
        ;;
esac
```

Циклы

Циклы позволяют многократно выполнять блок кода до тех пор, пока не будет выполнено определенное условие. В Bash есть три типа циклов:

Цикл for:

```
for variable in list; do
    # Выполнить для каждого элемента в списке
done
```

```
for (( i=0;i<10;i++ )); do
    # Выполнить 10 раз, изменяя переменную i от 0 до 9
done
```

Цикл while:

```
while condition; do
    # Выполнить, пока условие истинно
done
```

Цикл until:

```
until condition; do
    # Выполнять, пока условие ложно
done
```

Функции

Функции позволяют группировать команды в именованные блоки, которые можно вызывать по мере необходимости. Они используются для:

- организации кода,
- повышения читаемости кода,
- повторного использования.

Для создания функции используется следующий синтаксис:

```
function function_name() {
    # Код функции
}
```

Функции могут принимать аргументы, которые передаются в нее при вызове. Аргументы можно получить с помощью специальной переменной $1, $2 и т.д.

Например:

```
function greet1() {
    echo "Привет, $1!" # $1 локальная переменная функции
}

greet2() {
    echo "Привет, $1"
}

function greet3 {
    echo "Привет, $1"
}

greet1 John
greet2 John
greet3 John
```

Вывод у всех функций будет одинаковый:

```
Привет, John
```

Функции могут возвращать значения с помощью оператора return. Возвращаемое значение можно получить с помощью команды echo $?.

Например:

```
function sum() {
    return $(($1 + $2))
}
result=$(sum 10 15)
echo $result
```

Практическое задание

Написание скриптов на Bash — отличный способ повысить эффективность, упростив повторяющиеся процессы.

1. Для начала необходимо убедится, что вы работаете в оболочке Bash (вы можете посмотреть список всех переменных окружения при помощи команды *env*)

```
echo $SHELL

# в выводе вы должны увидеть используемую оболочку
# /bin/bash
```

2. Создайте новый файл с помощью команды *touch*
3. Откройте файл в любом текстовом редакторе, например Nano или Vim
4. В первой строке файла необходимо указать *shebang*. Эта строка показывает, что данный файл является Bash-скриптом:

```
#!/bin/bash
```

5. Напишите скрипт, который принимает в качестве первого аргумента название каталога, а в качестве второго название файла. При выполнении печатает на экране текст "Создаю каталог <название каталога>", создаёт каталог с таким именем и создаёт в нем файл с указанным именем файла.

```
#!/bin/bash

dir_name=$1
file_name=$2

echo Создаю каталог <$dir_name>
mkdir dir_name

echo Создаю в каталоге <$dir_name> файл <$file_name>Oki
touch $dir_name/$file_name

exit 0
```

6. Для выполнения скрипта можно воспользоваться командой *bash*, или сделать скрипт исполняемым с помощью *chmod +x*

```
# 1 вариант
$ bash script.sh

# 2 вариант
$ chmod +x script.sh
$ ./script.sh
```

7. Выполните скрипт 2 раза, указывая одинаковые имена каталогов и разные имена файлов. Проанализируйте вывод.
8. Выполним проверку на наличие папки с заданным именем, ввставьте код в имеющиеся скрипт

```
...

if [ -d $dir ]
then
  echo "Каталог с таким именем существует"
else
  echo "Создаю каталог <$dir>"
  mkdir $dir
fi

...
```

9. Повторите пункт 7
10. По аналогии добавьте проверку на наличие файла. Замените опцию *-d* в условии на опцию *-f*.
11. Напишем скрипт который предложит пользователю выбрать ,что создать: каталог или файл

```
 echo "Выберите что будем создавать (1, 2)"
echo "1 : создать каталог"
echo "2 : создать файл"
echo -n "Ваш выбор: "
read choice

case $choice in
  1)
  echo -n "Введите название каталога: "
  read dir
  mkdir $dir && echo "Каталог создан" || echo "Не удалось создать каталог"
  ;;
  2)
  echo -n "Введите название файла: "
  read file
  touch $file && echo "Файл создан" || echo "Не удалось создать файл"
  ;;
  *)
  echo "Введено значение отличное от 1 или 2"
esac

exit 0
```

Символы && и || используются для объединения команд. При использовании && следующая команда выполняется, только если выполнилась первая команда. При использовании || следующая команда выполнится только в случае, когда первая команда **не** выполнилась. 12. Попробуйте с помощью данного скрипта создать папку с номером группы в названии и файл с вашей фамилией в названии. Попробуйте создать каталог с тем же названием ещё раз 13. При помощи *man* изучите команду для работы с архивами *tar*. Напишите скрипт, который выполняет резервное копирование заданного каталога. Скрипт должен уточнять у пользователя каталог для резервирования, путь по которому необходимо сохранить резервную копию, так же необходимо ли архивировать резервируемый каталог. В зависимости от выбора пользователя каталог должен быть скопирован или заархивирован. Имя архива должно содержать текущую дату. 14. Проверьте работоспособность написанного скрипта. 15. (*) Создать скрипт, осуществляющий вывод меню, с последующим выводом выбранного действия, состоящего из следующих пунктов: - текущий пользователь, - объем используемой памяти, - системная дата и время, - время запуска системы, - выход.

Контрольные вопросы

1. Что такое локальные переменные Bash.
2. Что такое переменные окружения. Перечислите основные переменные и их назначение.
3. Какие условные операторы можно использовать для сравнения числовых переменных?
4. Что делают символьные операции && и ||?
5. Что делает команда test? Какие опции можно ей передать?

Содержание отчёта

1. Тема и цель работы
2. Ход работы с скриншотами показывающими выполненные действия
3. Ответы на контрольные вопросы
4. Вывод