



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Институт
Кафедра

Компьютерных наук
Прикладной математики

ЛАБОРАТОРНАЯ РАБОТА

По дисциплине: «Операционные системы Linux».

На тему: «Создание и использование сценариев(скриптов) в Linux».

Студент

ПМ-22

группа

подпись, дата

Щелконогов Е.А.

фамилия, инициалы

Руководитель

КТН

ученая степень, ученое звание

подпись, дата

Кургасов В.В.

фамилия, инициалы

Липецк 2024

Цель

Изучить основные возможности языка программирования высокого уровня Shell. Получить навыки написания и использования скриптов.

Ход работы

1. Используя команды `echo`, `printf`, вывести информационные сообщения на экран.

```
> echo Привет,мир
Привет,мир
> printf Привет,мир
Привет,мир%
```

2. Присвоить переменной `A` целочисленное значение. Просмотреть значение переменной `A`.

```
> a=10
> echo $a
10
```

3. Присвоить переменной `B` значение переменной `A`. Просмотреть значение переменной `B`.

```
> b=$a
> echo $b
10
```

4. Присвоить переменной C значение “путь до своего каталога”.
Перейти в этот каталог с использованием переменной.

```
> c="$HOME/.config/kitty"
> cd $c
```




5. Присвоить переменной D значение “имя команды”, а именно команды paste. Выполнить эту команду с использованием переменной.

```
> d="paste"
> $d test.txt test2.txt
halo      America ya!
halo      halo
halo      halo
```

6. Присвоить переменной E значение “имя команды”, а именно команды просмотра содержимого файла, просмотреть содержимое переменной. Выполнить эту команду с использованием переменной.

```
> e="cat"
> echo $e
cat
> $e test2.txt
America ya!
halo
halo
```



7. Присвоить переменной F значение “имя команды”, а именно сортировки содержимого текстового файла. Выполнить эту команду с использованием переменной.

```
> f="sort"
> echo $f
sort
> $f test.txt test2.txt

America ya!
halo
halo
halo
halo
halo
halo
> nvim test2.txt
> $f test.txt test2.txt

America ya!
Z
halo
halo
halo
halo
halo
halo
```

Написать скрипты, при запуске которых выполняются следующие действия:

- 1) Программа запрашивает значение переменной, а затем выводит значение этой переменной:

```
#!/bin/bash
echo "Введите значение для переменной"
read some_data
echo $some_data
```

```
> ./script1.sh
Введите значение для переменной
default
default
```

- 2) Программа запрашивает имя пользователя, затем здоровается с ним, используя значение переменной:

```
echo "Как тебя зовут?"
read name
echo "Привет, $name!"
```

```
> ./script1.sh
Как тебя зовут?
Егор
Привет, Егор!
```

- 3) Программа запрашивает значения двух переменных, вычисляет сумму (разность, произведение, деление) этих переменных. Результат выводится на экран (использовать команды а) EXPR б) BC)

```
#!/bin/bash
echo "Введи два числа"
read num1 num2
echo "Результат вычислений с помощью команды expr:\n"
expr $num1 + $num2
expr $num1 - $num2
expr $num1 \* $num2
expr $num1 / $num2
echo "Результат вычислений с помощью команды bc:\n"
echo "$num1+$num2" | bc
echo "$num1-$num2" | bc
echo "$num1*$num2" | bc
echo "$num1/$num2" | bc
```

```
> ./script1.sh
Введи два числа
2 3
Результат вычислений с помощью команды expr:\n
5
-1
6
0
Результат вычислений с помощью команды bc:\n
5
-1
6
0
```

- 4) Вычислить объем цилиндра. Исходные данные запрашиваются программой. Результат выводится на экран:

```

echo "Введите радиус цилиндра:"
read r
echo "Введите высоту цилиндра:"
read h

echo "3.14*$r^2*$h" | bc

```

```

> ./script1.sh
Введите радиус цилиндра:
5
Введите высоту цилиндра:
5
392.50

```

- 5) Используя позиционные параметры, отобразить имя программы, количество аргументов командной строки, значение каждого аргумента командной строки:

```

echo $0
args=("$@")
ELEMENTS=${#args[@]}
for (( i=0;i<$ELEMENTS;i++)); do
    echo $((i + 1)) : ${args[$i]}
done

```

```

> ./script1.sh hello hello hello
./script1.sh
1 : hello
2 : hello
3 : hello

```

- 6) Используя позиционный параметр, отобразить содержимое текстового файла, указанного в качестве аргумента командной строки. После паузы экран очищается:

```
cat "$1"  
sleep 10  
clear
```

```
> ./script1.sh test.txt  
america ya!
```

- 7) Используя оператор FOR, отобразить содержимое текстовых файлов текущего каталога поэкранно:

```
for file in *.txt; do  
    echo "≡ $file ≡"  
    more "$file"  
done
```

```
> ./script1.sh  
≡ test.txt ≡  
america ya!
```

- 8) Программой запрашивается ввод числа, значение которого затем сравнивается с допустимым значением. В результате этого сравнения на экран выдаются соответствующие сообщения:


```

min_value=10
max_value=100
read -p "Введите число: " user_input
if [[ $user_input =~ ^-?[0-9]+(\.[0-9]+)?$ ]]; then
    if (( $(echo "$user_input < $min_value" | bc -l) )); then
        echo "Введённое число меньше допустимого значения."
    elif (( $(echo "$user_input > $max_value" | bc -l) )); then
        echo "Введённое число больше допустимого значения."
    else
        echo "Введённое число находится в допустимом диапазоне."
    fi
else
    echo "Ошибка: Введите корректное число."
fi

```

```

> ./script1.sh
Введите число: 10
Введённое число находится в допустимом диапазоне.
> ./script1.sh
Введите число: b
Ошибка: Введите корректное число.

```

9) Программой запрашивается год, определяется, високосный ли он.

Результат выдается на экран:

```

#!/bin/bash
read -p "Введите год: " year
if [[ $year =~ ^[0-9]+$ ]]; then
    if (( (year % 4 == 0 && year % 100 != 0) || year % 400 == 0 )); then
        echo "Год $year является високосным."
    else
        echo "Год $year не является високосным."
    fi
else
    echo "Ошибка: Введите корректное целое число для года."
fi

```

```

> ./script1.sh
Введите год: 2016
Год 2016 является високосным.

```

10) Вводятся целочисленные значения двух переменных. Вводится диапазон данных. Пока значения переменных находятся в указанном диапазоне, их значения инкрементируются.

```
#!/bin/bash
read -p "Введите значение для переменной a: " a
read -p "Введите значение для переменной b: " b
read -p "Введите минимальное значение диапазона: " min_range
read -p "Введите максимальное значение диапазона: " max_range
while (( a ≥ min_range && a ≤ max_range && b ≥ min_range && b ≤ max_range )); do
    echo "Текущие значения: a = $a, b = $b"
    ((a++))
    ((b++))
done
echo "Значения переменных вышли за диапазон: a = $a, b = $b"
```

```
> ./script1.sh
Введите значение для переменной a: 1
Введите значение для переменной b: 2
Введите минимальное значение диапазона: 0
Введите максимальное значение диапазона: 10
Текущие значения: a = 1, b = 2
Текущие значения: a = 2, b = 3
Текущие значения: a = 3, b = 4
Текущие значения: a = 4, b = 5
Текущие значения: a = 5, b = 6
Текущие значения: a = 6, b = 7
Текущие значения: a = 7, b = 8
Текущие значения: a = 8, b = 9
Текущие значения: a = 9, b = 10
Значения переменных вышли за диапазон: a = 10, b = 11
```

- 11) В качестве аргумента командной строки указывается пароль. Если пароль введен верно, постранично отображается в длинном формате с указанием скрытых файлов содержимое каталога /etc.

```
#!/bin/bash
correct_password="GhostOfMyGenius"
if [ "$1" = "$correct_password" ]; then
    ls -la /etc | less
else
    echo "Ошибка: Неверный пароль."
    exit 1
fi
```

```
lrwxr-xr-x@ 1 root wheel 11 22 окт 10:49 /etc → private/etc
[END]
```

- 12) Проверить, существует ли файл. Если да, выводится на экран его содержимое, если нет - выдается соответствующее сообщение.

```
#!/bin/bash
read -p "Введите имя файла для проверки: " filename
if [[ -f "$filename" ]]; then
    echo "Содержимое файла $filename:"
    cat "$filename"
else
    echo "Файл $filename не существует."
fi
```

```
> ./script1.sh
Введите имя файла для проверки: test.txt
Содержимое файла test.txt:
america ya!
```

- 13) Если файл есть каталог и этот каталог можно читать, просматривается содержимое этого каталога. Если каталог отсутствует, он создается. Если файл не есть каталог, просматривается содержимое файла.

```
#!/bin/bash
read -p "Введите имя файла или каталога: " filename

if [[ -d "$filename" ]]; then
    if [[ -r "$filename" ]]; then
        echo "Содержимое каталога $filename:"
        ls -la "$filename"
    else
        echo "Каталог $filename существует, но недоступен для чтения."
    fi
elif [[ -e "$filename" ]]; then
    echo "Содержимое файла $filename:"
    cat "$filename"
else
    echo "Каталог $filename не существует. Создаем каталог."
    mkdir "$filename" && echo "Каталог $filename создан."
fi
```

```
> ./script1.sh
Введите имя файла или каталога: test.txt
Содержимое файла test.txt:
america ya!
> ./script1.sh
Введите имя файла или каталога: touchdir
Каталог touchdir не существует. Создаем каталог.
Каталог touchdir создан.
> cd touchdir
> touch file1 file2
> chmod +x file2
> cd ..
> ./script1.sh
Введите имя файла или каталога: touchdir
Содержимое каталога touchdir:
total 0
drwxr-xr-x  4 transhumanist  staff  128 11 ноя 18:29 .
drwxr-xr-x  5 transhumanist  staff  160 11 ноя 18:29 ..
-rw-r--r--  1 transhumanist  staff    0 11 ноя 18:29 file1
-rwxr-xr-x  1 transhumanist  staff    0 11 ноя 18:29 file2
```

- 14) Анализируются атрибуты файла. Если первый файл существует и используется для чтения, а второй файл существует и используется для записи, то содержимое первого файла перенаправляется во второй файл. В случае несовпадений указанных атрибутов или отсутствия файлов на экран выдаются соответствующие сообщения (использовать имена файлов и/или позиционные параметры).

```
#!/bin/bash
if [[ $# -ne 2 ]]; then
    echo "Ошибка: Укажите два файла в качестве аргументов."
    exit 1
fi
if [[ -e "$1" && -r "$1" ]]; then
    echo "Файл '$1' существует и доступен для чтения."
else
    echo "Ошибка: Файл '$1' либо не существует, либо не доступен для чтения."
    exit 1
fi
if [[ -e "$2" && -w "$2" ]]; then
    echo "Файл '$2' существует и доступен для записи."
else
    echo "Ошибка: Файл '$2' либо не существует, либо не доступен для записи."
    exit 1
fi
cat "$1" > "$2"
echo "Содержимое файла '$1' успешно перенаправлено в файл '$2'."
```

```
> ./script1.sh ./test.txt ./touchdir/file1
Файл './test.txt' существует и доступен для чтения.
Файл './touchdir/file1' существует и доступен для записи.
Содержимое файла './test.txt' успешно перенаправлено в файл './touchdir/file1'.
```

- 15) Если файл запуска программы найден, программа запускается (по выбору).

```
#!/bin/bash
read -p "Введите путь к файлу программы для проверки: " program_file
if [[ -e "$program_file" && -x "$program_file" ]]; then
    echo "Файл '$program_file' найден и является исполнимым."
    read -p "Хотите запустить программу? (y/n): " choice
    if [[ "$choice" = "y" || "$choice" = "Y" ]]; then
        "$program_file"
    else
        echo "Программа не была запущена."
    fi
else
    echo "Ошибка: Файл '$program_file' не найден или не является исполнимым."
fi
```

```
> ./script1.sh
Введите путь к файлу программы для проверки: ./script1.sh
Файл './script1.sh' найден и является исполнимым.
Хотите запустить программу? (y/n): y
Введите путь к файлу программы для проверки: ./touchdir/file1
Ошибка: Файл './touchdir/file1' не найден или не является исполнимым.
```

- 16) В качестве позиционного параметра задается файл, анализируется его размер. Если размер файла больше нуля, содержимое файла сортируется по первому столбцу по возрастанию, отсортированная информация помещается в другой файл, содержимое которого затем отображается на экране.

```
#!/bin/bash
if [[ $# -ne 1 ]]; then
    echo "Ошибка: Укажите файл в качестве аргумента."
    exit 1
fi
input_file="$1"
output_file="sorted_output.txt"
if [[ -e "$input_file" && -s "$input_file" ]]; then
    echo "Файл '$input_file' существует и имеет размер больше нуля."
    sort -k1,1 "$input_file" > "$output_file"
    echo "Отсортированное содержимое файла '$input_file':"
    cat "$output_file"
else
    echo "Ошибка: Файл '$input_file' не существует или имеет нулевой размер."
fi
```

```
> ./script1.sh ./test.txt
Файл './test.txt' существует и имеет размер больше нуля.
Отсортированное содержимое файла './test.txt':
90pp
america ya!
default
hallo
save me
z
zed
```

Вывод

В ходе данной лабораторной работы были изучены основные возможности языка программирования высокого уровня Shell, а также получены навыки написания и использования скриптов.

Контрольные вопросы

1. В чем отличие пользовательских переменных от переменных среды?

- Пользовательские переменные: создаются и используются пользователями в рамках текущей сессии. Они не доступны дочерним процессам, если не экспортированы.
- Переменные среды: глобальные переменные, доступные всем дочерним процессам, когда они экспортированы с помощью команды `export`.

2. Математические операции в SHELL?

- В Shell поддерживаются базовые арифметические операции: `+`, `-`, `*`, `/`, `%` для целочисленного деления. Для выполнения вычислений используется синтаксис `$((expression))`.
- Для работы с числами с плавающей точкой используется команда `bc`.

3. Условные операторы в shell?

- Основные условные операторы включают: `if`, `then`, `else`, `elif`, и `fi` для завершения блока.
- Проверки могут выполняться с использованием `test` или `[` (например, `[-f filename]` для проверки существования файла).
- `case` используется для сопоставления значений с шаблонами.

4. Принцип построения простых и составных условий?

- Простое условие проверяет одно выражение, например: `[-e filename]`.
- Составные условия используют операторы `&&` (логическое И) и `||` (логическое ИЛИ), например: `[-e filename] && [-r filename]`.

5. Циклы в SHELL?

- Циклы `for`, `while`, и `until` используются для итераций.

- `for` используется для перебора списка элементов, `while` выполняется, пока условие истинно, `until` — пока условие ложно.

6. Массивы и модули в SHELL?

- Массивы задаются как `array=(item1 item2 item3)`.
- Доступ к элементам осуществляется через `array[индекс]`.
- В Shell отсутствует встроенная поддержка модулей, но можно использовать внешние скрипты и функции для аналогичных целей.

7. Чтение параметров командной строки?

- Позиционные параметры `$1`, `$2`, ..., `$n` представляют переданные аргументы.
- `$@` и `$*` содержат все параметры, переданные скрипту.
- `shift` смещает параметры влево, удаляя первый.

8. Как различать ключи и параметры?

- Обычно ключи начинаются с `-` или `--` (например, `-f`, `--file`).
- Для обработки ключей используют команды `getopts` или `while`, где проверяются переданные аргументы.

9. Чтение данных из файлов?

- С помощью команды `cat`, `read` в цикле `while`, `awk`, или `sed`.
- Например: `while read line; do ... done < filename`.

10. Стандартные дескрипторы файлов?

- `0 (stdin)` — стандартный ввод.
- `1 (stdout)` — стандартный вывод.
- `2 (stderr)` — стандартный вывод ошибок.

11. Перенаправление вывода?

- `>` для перенаправления `stdout` в файл, `>>` для добавления в конец файла.
- `2>` для перенаправления `stderr`, `&>` для перенаправления `stdout` и `stderr` в один файл.

12. Подавление вывода?

- Подавить вывод можно с перенаправлением в /dev/null, например: `command > /dev/null 2>&1`.

13. Отправка сигналов скриптам?

- Сигналы, такие как SIGINT, SIGTERM, можно отправить с помощью команды `kill`.
- Обработка сигналов внутри скрипта осуществляется с помощью `trap`.

14. Использование функций?

- Синтаксис определения функции:

```
function_name() {  
    # Команды  
}
```
- Вызываются по имени: `function_name`.

15. Обработка текстов (чтение, выбор, вставка, замена данных)?

- Команды `awk`, `sed`, `grep` широко используются для обработки текста.
- `awk` для выбора и обработки данных, `sed` — для замены, вставки и удаления строк.

16. Отправка сообщений в терминал пользователя?

- Сообщения можно отправить с помощью команды `echo`, `wall` для массовой рассылки, или `write` для отправки конкретному пользователю.

17. BASH и SHELL - синонимы?

- BASH — это один из видов интерпретаторов командной строки Shell, имеющий больше возможностей по сравнению с классическим Bourne Shell (`sh`).
- Shell — общее понятие для всех интерпретаторов командной строки, таких как BASH, `zsh`, `fish`.

**18. PowerShell в операционных системах семейства Windows:
назначение и особенности?**

- PowerShell — это командная оболочка Windows и язык скриптов с поддержкой администрирования, основанный на .NET.
- PowerShell используется для управления операционной системой и приложениями через объекты, что обеспечивает мощные средства автоматизации.