

Лабораторная работа №13

Работа с shell скриптами

Mohamed Musa

Содержание

1	Цель работы	5
2	Задание	6
2.1	Условные операторы	9
2.2	Циклы	14
2.3	Функции	17
2.4	Работа с аргументами командной строки	20
2.5	Отладка скриптов	22
2.6	Создание и запуск скриптов	23
3	Выполнение лабораторной работы	25
3.1	Задание 1: Скрипт с getopt и grep	25
3.2	Задание 2: Программа на языке Си	26
3.3	Задание 3: Управление файлами	27
3.4	Задание 4: Архивация с tar	28
4	Выводы	30
4.1	Освоенные технологии	30
4.2	Практическое применение	31
	Список литературы	32

Список иллюстраций

3.1	Скрипт <code>getopts_script.sh</code>	26
3.2	Программа <code>number_check.c</code>	27
3.3	Выполнение скриптов	28
3.4	Результаты архивации	29

Список таблиц

1 Цель работы

Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

2 Задание

1. Написать командный файл с использованием команд `getopts` и `grep` для анализа командной строки с ключами
2. Написать программу на языке Си для определения знака числа с передачей кода завершения в оболочку
3. Написать командный файл для создания и удаления пронумерованных файлов
4. Написать командный файл для архивации файлов с помощью команды `tar`

Правила именования:

- Начинается с буквы или подчеркивания
- Может содержать буквы, цифры, подчеркивания
- Регистрозависимые (`VAR` и `var` — разные переменные)
- Не использовать зарезервированные слова

Примеры:

```
name="John"  
age=25  
_temp="temporary"  
USER_HOME="/home/user"
```

2.0.1 Использование переменных

Обращение к переменной:

```

echo $variable          # 1234567890 1234567890
echo ${variable}        # 1 1234567890 1234567890 (12345678901234567890)
echo "${variable}"      # 1 1234567890 (1234567890 1234567890)

```

Примеры:

```

name="Alice"
echo "Hello, $name"      # Hello, Alice
echo "Hello, ${name}!"   # Hello, Alice!
echo "Path: ${HOME}/docs" # Path: /home/user/docs

```

2.0.2 Специальные переменные

Параметры скрипта:

- \$0 — имя скрипта
- \$1, \$2, ..., \$9 — позиционные параметры (аргументы)
- \${10}, \${11}, ... — параметры с номером > 9
- \$# — количество аргументов
- \$@ — все аргументы как отдельные слова
- \$* — все аргументы как одна строка
- \$? — код возврата последней команды
- \$\$ — PID текущего процесса
- \$! — PID последнего фонового процесса

Переменные окружения:

- \$HOME — домашняя директория пользователя
- \$USER — имя пользователя
- \$PWD — текущая директория
- \$OLDPWD — предыдущая директория

- \$PATH — пути поиска команд
- \$SHELL — путь к текущей оболочке
- \$HOSTNAME — имя хоста

Примеры использования:

```
#!/bin/bash

echo "Имя пользователя: $0"
echo "Путь к скрипту: $1"
echo "Путь к папке: $2"
echo "Путь к файлу: $#"
echo "Путь к текущей оболочке: $@"
echo "Имя пользователя: $USER"
echo "Путь к домашней папке: $HOME"
```

2.0.3 Операции с переменными

Длина строки:

```
string="Hello"
echo ${#string}          # 5
```

Подстроки:

```
string="Hello World"
echo ${string:0:5}        # Hello (начало 0, длина 5)
echo ${string:6}          # World (начало 6, длина 5)
```

Замена подстроки:


```
string="Hello World"
echo ${string/World/Bash}      # Hello Bash (XXXXXX XXXXXXXXXX)
echo ${string//o/O}            # Hello WOrld (XXX XXXXXXXXXX)
```

Значение по умолчанию:

```
echo ${variable:-default}      # default XXXX variable XXXXXXX
echo ${variable:=default}      # XXXXXXXXXX default XXXX XXXXXXX
```

2.1 Условные операторы

2.1.1 Оператор if

Синтаксис:

```
if [ condition ]; then
    # XXXXXXXX XXXX XXXXXXX
elif [ condition2 ]; then
    # XXXXXXXX XXXX condition2 XXXXXXX
else
    # XXXXXXXX XXXX XXX XXXX
fi
```

Примеры:

```
if [ $age -gt 18 ]; then
    echo "XXXXXXXXXXXXXXXXXXXX"
else
    echo "XXXXXXXXXXXXXXXXXXXX"
fi
```

2.1.2 Операторы сравнения

Числовые сравнения:

- `-eq` — равно (equal)
- `-ne` — не равно (not equal)
- `-gt` — больше (greater than)
- `-ge` — больше или равно (greater or equal)
- `-lt` — меньше (less than)
- `-le` — меньше или равно (less or equal)

Примеры:

```
if [ $num -eq 10 ]; then
    echo "Число равно 10"
fi

if [ $age -ge 18 ]; then
    echo "Взрослый"
fi
```

Строковые сравнения:

- `=` или `==` — строки равны
- `!=` — строки не равны
- `<` — меньше (лексикографически)
- `>` — больше (лексикографически)
- `-z` — строка пустая (zero length)
- `-n` — строка не пустая (non-zero length)

Примеры:

```

if [ "$name" = "Alice" ]; then
    echo "Привет, Alice!"
fi

if [ -z "$variable" ]; then
    echo "Переменная пуста"
fi

if [ -n "$variable" ]; then
    echo "Переменная не пуста"
fi

```

Проверка файлов:

- -e — файл существует (exists)
- -f — обычный файл (file)
- -d — директория (directory)
- -r — файл доступен для чтения (readable)
- -w — файл доступен для записи (writable)
- -x — файл исполняемый (executable)
- -s — файл не пустой (size > 0)
- -L — символическая ссылка (link)

Примеры:

```

if [ -f "/etc/passwd" ]; then
    echo "Файл существует"
fi

if [ -d "/home/user" ]; then

```

```

        echo "XXXXXXXXXX XXXXXXXXXX"
    fi

    if [ -x "./script.sh" ]; then
        echo "XXXXXX XXXXXXXXXXXX"
    fi

```

2.1.3 Логические операторы

Операторы:

- && — логическое И (AND)
- || — логическое ИЛИ (OR)
- ! — логическое НЕ (NOT)

Примеры:

```

# & (AND)

if [ $age -ge 18 ] && [ $age -le 65 ]; then
    echo "XXXXXXXXXXXX XXXXXXX"
fi

# || (OR)

if [ "$name" = "Alice" ] || [ "$name" = "Bob" ]; then
    echo "XXXXXXXXXX ||"
fi

# ! (NOT)

if [ ! -f "/tmp/file.txt" ]; then
    echo "XXXXX || XXXXXXXXXXXX"
fi

```

2.1.4 Оператор case

Синтаксис:

```
case $variable in
    pattern1)
        # .....
        ;;
    pattern2)
        # .....
        ;;
    * )
        # .....
        ;;
esac
```

Примеры:


```
case $choice in
    1)
        echo "..... 1"
        ;;
    2)
        echo "..... 2"
        ;;
    [3-5])
        echo "..... 3, 4 ..... 5"
        ;;
    * )
        echo "....."
```

```
;;  
esac
```







2.2 Циклы

2.2.1 Цикл for

Синтаксис 1 (диапазон):

```
for variable in list; do  
    #   
done
```

Примеры:

```
#    
for i in 1 2 3 4 5; do  
    echo "done  
  
#   
for i in {1..10}; do  
    echo $i  
done  
  
#    
for i in {0..20..2}; do  
    echo $i      # 0, 2, 4, ..., 20  
done
```

```
# Пример 1: Цикл for по файлам
for file in *.txt; do
    echo "Файл: $file"
done

# Пример 2: Цикл for по аргументам
for arg in "$@"; do
    echo "Аргумент: $arg"
done
```

Синтаксис 2 (C-style):

```
for ((i=0; i<10; i++)); do
    echo $i
done
```

2.2.2 Цикл while

Синтаксис:

```
while [ condition ]; do
    # Команды
done
```

Примеры:

```
# Пример: Цикл while с счетчиком
counter=1
while [ $counter -le 5 ]; do
    echo "Счетчик: $counter"
    ((counter++))
done
```

```
done

# 1. Задаем файл, который будем читать
while IFS= read -r line; do
    echo "Читаем строку: $line"
done < file.txt

# 2. Задаем условие, которое будет выполняться до тех пор, пока не будет нажата клавиша Ctrl+C
while true; do
    echo "Нажмите Ctrl+C для выхода"
    sleep 1
done
```

2.2.3 Цикл until

Синтаксис:

```
until [ condition ]; do
    # ...
done
```

Примеры:

```
counter=1
until [ $counter -gt 5 ]; do
    echo "Счетчик: $counter"
    ((counter++))
done
```


2.2.4 Управление циклами

Команды:

- `break` — выход из цикла
- `continue` — переход к следующей итерации

Примеры:

```
# break
for i in {1..10}; do
    if [ $i -eq 5 ]; then
        break      # XXXXX XXX i=5
    fi
    echo $i
done

# continue
for i in {1..10}; do
    if [ $((i % 2)) -eq 0 ]; then
        continue   # XXXXXXXXXXXX XXXXXX
    fi
    echo $i
done
```

2.3 Функции

2.3.1 Объявление функций

Синтаксис 1:

```
function_name() {
    # 123456789
}
```

Синтаксис 2:

```
function function_name {
    # 123456789
}
```

Примеры:

```
# 123456789 123456789
greet() {
    echo "123456789!"
}

# 123456789 123456789
greet

# 123456789 1 123456789123456789
greet_user() {
    echo "123456789, $1!"
}

greet_user "Alice"    # 123456789, Alice!
```

2.3.2 Параметры функций

Доступ к параметрам:

- \$1, \$2, ... — позиционные параметры
- \$# — количество параметров
- \$@ — все параметры
- \$* — все параметры как одна строка

Примеры:

```
sum() {
    local result=$(( $1 + $2 ))
    echo $result
}

result=$(sum 5 3)
echo "Сумма: $result"      # Сумма: 8
```

2.3.3 Локальные переменные

Использование local:

```
my_function() {
    local local_var="локальная переменная"
    global_var="глобальная переменная"
    echo $local_var
}

my_function
echo $global_var      # глобальная переменная
echo $local_var       #  локальная переменная (локальная)
```

2.3.4 Возврат значений

Команда `return`:

```
is_even() {  
    if [ $((($1 % 2)) -eq 0 ]; then  
        return 0      # 0 (четное)  
    else  
        return 1      # 1 (нечетное)  
    fi  
}  
  
if is_even 4; then  
    echo "четное"  
fi
```

Вывод через `echo`:

```
multiply() {  
    echo $((($1 * $2))  
}  
  
result=$(multiply 6 7)  
echo "результат: $result"      # результат: 42
```

2.4 Работа с аргументами командной строки

2.4.1 Обработка аргументов

Примеры:

```
#!/bin/bash

# Программа принимает три аргумента

if [ $# -eq 0 ]; then
    echo "Программа требует три аргумента"
    exit 1
fi

# Программа принимает три аргумента

for arg in "$@"; do
    echo "Аргумент: $arg"
done

# Программа принимает три аргумента
echo "Аргумент 1: $1"
echo "Аргумент 2: $2"
```

2.4.2 Команда getopt

Обработка опций:

```
#!/bin/bash

while getopt "a:b:c" opt; do
    case $opt in
        a)
            echo "Опция -a с аргументом: $OPTARG"
            ;;
        b)
            ;;
    esac
done
```

```

        echo "XXXXX -b X XXXXXXXXXXXX: $OPTARG"
        ;;
c)
    echo "XXXXX -c"
    ;;
\?)
    echo "XXXXXXXXX XXXX: -$OPTARG"
    exit 1
    ;;
esac
done

```

2.5 Отладка скриптов

2.5.1 Опции отладки

Команды:

```

bash -x script.sh      # XXXXXXXXXXXX XXXXXXXXXXXX
bash -v script.sh      # XXXXX XXXXX XXXXXXXXX
bash -n script.sh      # XXXXXXXXX XXXXXXXXXXXX

```

В скрипте:

```

#!/bin/bash

set -x      # XXXXXXXXX XXXXXXXXXXXX
# XXXX
set +x      # XXXXXXXXXXXX XXXXXXXXXXXX

```

```
set -e      # Остановка скрипта при ошибке
set -u      # Остановка скрипта при использовании несуществующей переменной
set -o pipefail  # Остановка скрипта при ошибке в команде, связанной с конвейером
```

2.5.2 Вывод отладочной информации

Примеры:

```
# Вывод значения переменной
echo "DEBUG: variable=$variable"

# Вывод информации о выполнении условия
if [ condition ]; then
    echo "DEBUG: условие выполнено"
fi

# Вывод информации о выполнении функции
function_name() {
    echo "DEBUG: вход в функцию: $@"
    # ... тело функции ...
    echo "DEBUG: выход из функции"
}
```

2.6 Создание и запуск скриптов

2.6.1 Создание скрипта

Шаги:

1. Создать файл: `touch script.sh`

2. Добавить shebang: `#!/bin/bash`
3. Написать код
4. Сделать исполняемым: `chmod +x script.sh`
5. Запустить: `./script.sh`

Пример:

```
# Создаем файл
cat > hello.sh << 'EOF'
#!/bin/bash
echo "Hello, World!"
EOF

# Делаем файл исполняемым
chmod +x hello.sh

# Запускаем
./hello.sh
```

2.6.2 Способы запуска

Различные способы:

<code>./script.sh</code>	# <code>./script.sh</code> (нужно <code>chmod +x</code>)
<code>bash script.sh</code>	# <code>bash script.sh</code>
<code>sh script.sh</code>	# <code>sh script.sh</code>
<code>source script.sh</code>	# <code>source script.sh</code> (нужно <code>chmod +x</code>)
<code>. script.sh</code>	# <code>. script.sh</code>

3 Выполнение лабораторной работы

3.1 Задание 1: Скрипт с getopt и grep

Создан скрипт `getopts_script.sh`, который анализирует командную строку с ключами и выполняет поиск в файле.

Поддерживаемые ключи: - `-i<inputfile>` — прочитать данные из указанного файла - `-o<outputfile>` — вывести данные в указанный файл - `-p<XXXXXX>` — указать шаблон для поиска - `-C` — различать большие и малые буквы - `-n` — выдавать номера строк

Содержимое скрипта показано на рисунке Рисунок 3.1.

```
1
foot
#!/bin/bash

if [ $# -lt 2 ]; then
    echo "used0 <the resulted directory> <ka: lab-name>lab> <directory name>"
    exit 1
fi

SOURCE_DIR=$1
DEST_DIR=$2
BACKUP_DATA=$(date +%Y%m%d_%H%M%S)
ARCHIVE_NAME="backup_${BACKUP_DATA}.tar.gz"

if [ ! -d "$SOURCE_DIR" ]; then
    echo "error : the directory '$SOURCE_DIR' doesnt exist"
    exit 1
fi

if [ ! -d "$DEST_DIR" ]; then
    mkdir -p "$DEST_DIR"
    if [ $? -ne 0 ]; then
        echo "error making the directory '$DEST_DIR' was unsuccessful"
        exit 1
    fi
    echo "direcozy '$DEST_DIR' was created"
fi

echo "creating the archive for '$SOURCE_DIR' into '$DEST_DIR'/.f....."
tar -czf "$DEST_DIR/$ARCHIVE_NAME" -C "$SOURCE_DIR" .

if [ $? -eq 0 ]; then
    echo "the archive was successfully created"
    echo "the archive was saved like: '$DEST_DIR/$ARCHIVE_NAME'"
    echo "the size of the archive is : $(du -h "$DEST_DIR/$ARCHIVE_NAME" | cut -f1)"
else
    echo "error: failed to create the archive"
    exit 1
fi

exit 0
```

Рисунок 3.1: Скрипт getopts_script.sh

3.2 Задание 2: Программа на языке Си

Создана программа `number_check.c`, которая определяет знак введенного числа и передает код завершения в оболочку.

Коды завершения: - 0 — число равно нулю - 1 — число больше нуля - 2 — число меньше нуля

Содержимое программы показано на рисунке Рисунок 3.2.


```

[backup.sh строка 18: [: отсуствует символ «]»
creating the archive for 'test' into 'archive' /.....
tar: Рубный отказ от создания пустого архива
Попытка: «tar --help» или «tar --usage» для
получения более подробного описания.
[backup.sh строка 32: [: отсуствует символ «]»
error: failed to create the archive
[ceazen@MuhammadMusa lab1115] /backup.sh australia austral_backup
[backup.sh строка 18: data: команда не найдена
[backup.sh строка 11: BACKUP_DATA: команда не найдена
[backup.sh строка 13: [: отсуствует символ «]»
[backup.sh строка 19: [: отсуствует символ «]»
creating the archive for 'australia' into 'australi_backup' /.....
tar: Рубный отказ от создания пустого архива
Попытка: «tar --help» или «tar --usage» для
получения более подробного описания.
[backup.sh строка 32: [: отсуствует символ «]»
error: failed to create the archive
[ceazen@MuhammadMusa lab1115] vi backup.sh
[ceazen@MuhammadMusa lab1115] /backup.sh australia austral_backup
[backup.sh строка 11: BACKUP_DATA: команда не найдена
[backup.sh строка 13: [: отсуствует символ «]»
[backup.sh строка 19: [: отсуствует символ «]»
creating the archive for 'australia' into 'australi_backup' /.....
tar: SOURCE_DIR: Функция open завершилась с ошибкой (child) : Нет такого файла или каталога
australi_backup/backup_tar.gz: Функция open завершилась с ошибкой: Error is not recoverable: exiting now
Нет такого файла или каталога
tar (child): Error is not recoverable: exiting now
[backup.sh строка 32: [: отсуствует символ «]»
error: failed to create the archive
[ceazen@MuhammadMusa lab1115] vi backup.sh
[backup.sh строка 11: BACKUP_DATA: команда не найдена
[backup.sh строка 13: [: отсуствует символ «]»
[backup.sh строка 19: [: отсуствует символ «]»
creating the archive for 'australia' into 'australi_backup' /.....
tar: SOURCE_DIR: Функция open завершилась с ошибкой (child): australi_backup/backup_tar.gz: Функция open завершилась с ошибкой: Нет такого файла или каталога
tar: Error is not recoverable: exiting now
Нет такого файла или каталога
tar (child): Error is not recoverable: exiting now
[backup.sh строка 32: [: отсуствует символ «]»
error: failed to create the archive
[ceazen@MuhammadMusa lab1115] vi backup.sh
[ceazen@MuhammadMusa lab1115] vi backup.sh
[ceazen@MuhammadMusa lab1115] ls
archive backup.sh test
[ceazen@MuhammadMusa lab1115] vi backup.sh
[ceazen@MuhammadMusa lab1115] /backup.sh test archive
[backup.sh строка 13: [: отсуствует символ «]»
[backup.sh строка 19: [: отсуствует символ «]»
creating the archive for 'test' into 'archive' /.....
tar: SOURCE_DIR: Функция open завершилась с ошибкой: Нет такого файла или каталога
tar: Error is not recoverable: exiting now
[backup.sh строка 32: [: отсуствует символ «]»
error: failed to create the archive
[ceazen@MuhammadMusa lab1115] ls
archive backup.sh test
[ceazen@MuhammadMusa lab1115] ls archive
ls: невозможно получить доступ к 'archive': Нет такого файла или каталога
[ceazen@MuhammadMusa lab1115] ls archive
ls: невозможно получить доступ к 'archive': Нет такого файла или каталога
[ceazen@MuhammadMusa lab1115] touch args.sh
[ceazen@MuhammadMusa lab1115]

```

Рисунок 3.3: Выполнение скриптов

3.4 Задание 4: Архивация с tar

Создан скрипт `tar_script.sh` для архивации файлов с помощью команды `tar`.

Возможности: - Архивация всех файлов в указанной директории - Архивация только файлов, измененных менее недели назад (с использованием `find`) - Создание архивов с временными метками

Результаты выполнения показаны на рисунке Рисунок 3.4.

```
1 foot
foot
./backup.sh: строка 32: [: отсутствует символ «]»
error: failed to create the archive
[ceazer@MohammedUsa lab11]$ ls
archive backup.sh test
[ceazer@MohammedUsa lab11]$ ls archive
ls: невозможно получить доступ к 'archive': Нет такого файла или каталога
[ceazer@MohammedUsa lab11]$ ls archive
backup_20250921_220017.tar.gz
[ceazer@MohammedUsa lab11]$ touch args.sh
[ceazer@MohammedUsa lab11]$ vi args.sh
[ceazer@MohammedUsa lab11]$ chmod +x args.sh
[ceazer@MohammedUsa lab11]$ ./args.sh hello
the total of the arguments: 1
    echo the name of the script: ./args.sh
    echo all arguments : hello

    echo the arguments :
./args.sh: строка 11:      echo "arguments $count: $arg"
[ceazer@MohammedUsa lab11]$ vi args.sh
[ceazer@MohammedUsa lab11]$ ./args.sh hello
./args.sh: строка 16: неожиданный конец файла во время поиска «"»
[ceazer@MohammedUsa lab11]$ vi args.sh
[ceazer@MohammedUsa lab11]$ ./args.sh hello
./args.sh: строка 16: неожиданный конец файла во время поиска «"»
[ceazer@MohammedUsa lab11]$ vi args.sh
[ceazer@MohammedUsa lab11]$ ./args.sh hello
./args.sh: строка 16: неожиданный конец файла во время поиска «"»
[ceazer@MohammedUsa lab11]$ vi args.sh
[ceazer@MohammedUsa lab11]$ touch countfiles
[ceazer@MohammedUsa lab11]$ touch countfiles.sh
[ceazer@MohammedUsa lab11]$ vi countfiles.sh
[ceazer@MohammedUsa lab11]$ chmod +x countfiles.sh
[ceazer@MohammedUsa lab11]$ ./countfiles.sh archive
./countfiles.sh: строка 3: 1-: команда не найдена
error : Directory '' does not exists
[ceazer@MohammedUsa lab11]$ ./countfiles.sh archive
./countfiles.sh: строка 3: 1-: команда не найдена
error : Directory '' does not exists
[ceazer@MohammedUsa lab11]$ vi countfiles.sh
[ceazer@MohammedUsa lab11]$ ./countfiles.sh archive
./countfiles.sh: строка 3: 1-: команда не найдена
error : Directory '' does not exists
[ceazer@MohammedUsa lab11]$ vi countfiles.sh
[ceazer@MohammedUsa lab11]$ ./countfiles.sh archive
content fo directory archive
==
./countfiles.sh: строка 14: синтаксическая ошибка рядом с неожиданным маркером «do»
./countfiles.sh: строка 14:      if [ -e "$file" ]; do
[ceazer@MohammedUsa lab11]$ ./countfiles.sh archive
content fo directory archive
==
./countfiles.sh: строка 14: синтаксическая ошибка рядом с неожиданным маркером «do»
./countfiles.sh: строка 14:      if [ -e "$file" ]; do
[ceazer@MohammedUsa lab11]$ vi countfiles.sh
[ceazer@MohammedUsa lab11]$ ./countfiles.sh archive
content fo directory archive
==
total files : 1
total directoried: 1
[ceazer@MohammedUsa lab11]$
```

Рисунок 3.4: Результаты архивации

4 Выводы

В ходе выполнения лабораторной работы были получены практические навыки программирования в командной оболочке `bash`:

4.1 Освоенные технологии

1. ☒ Команда `getopts`

- Обработка опций командной строки
- Работа с аргументами опций
- Валидация входных параметров

2. ☒ Интеграция C и `shell`

- Компиляция программ на C
- Передача кодов завершения
- Анализ результатов выполнения

3. ☒ Управление файлами

- Создание пронумерованных файлов
- Удаление файлов по шаблону
- Работа с циклами и условиями

4. ☒ Архивация данных

- Использование команды `tar`

- Поиск файлов с помощью `find`
- Фильтрация по времени модификации

4.2 Практическое применение

Полученные навыки могут быть применены для:

- **Автоматизации задач** — создание скриптов для рутинных операций
- **Системного администрирования** — управление файлами и процессами
- **Обработки данных** — поиск и фильтрация информации
- **Интеграции программ** — связывание различных утилит

Все четыре скрипта успешно выполняют поставленные задачи и демонстрируют различные аспекты программирования в `bash`.

Список литературы

- Advanced Bash-Scripting Guide: <https://tldp.org/LDP/abs/html/>
- Bash Reference Manual: <https://www.gnu.org/software/bash/manual/>
- Linux Command Line and Shell Scripting Bible