Лабораторная работа №13

Работа с shell скриптами

Mohamed Musa

Содержание

# 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 # простое обращение  
echo ${variable} # с фигурными скобками (рекомендуется)  
echo "${variable}" # в кавычках (сохраняет пробелы)

**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 скрипта: $0"  
echo "Первый аргумент: $1"  
echo "Второй аргумент: $2"  
echo "Количество аргументов: $#"  
echo "Все аргументы: $@"  
echo "Пользователь: $USER"  
echo "Домашняя директория: $HOME"

### 2.0.3 Операции с переменными

**Длина строки:**

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

**Подстроки:**

string="Hello World"  
echo ${string:0:5} # Hello (с позиции 0, длина 5)  
echo ${string:6} # World (с позиции 6 до конца)

**Замена подстроки:**

string="Hello World"  
echo ${string/World/Bash} # Hello Bash (первое вхождение)  
echo ${string//o/0} # Hell0 W0rld (все вхождения)

**Значение по умолчанию:**

echo ${variable:-default} # default если variable пустая  
echo ${variable:=default} # присвоить default если пустая

## 2.1 Условные операторы

### 2.1.1 Оператор if

**Синтаксис:**

if [ condition ]; then  
 # команды если истина  
elif [ condition2 ]; then  
 # команды если condition2 истина  
else  
 # команды если все ложь  
fi

**Примеры:**

if [ $age -gt 18 ]; then  
 echo "Совершеннолетний"  
else  
 echo "Несовершеннолетний"  
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, Алиса!"  
fi  
  
if [ -z "$variable" ]; then  
 echo "Переменная пустая"  
fi  
  
if [ -n "$variable" ]; then  
 echo "Переменная не пустая"  
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 "Директория существует"  
fi  
  
if [ -x "./script.sh" ]; then  
 echo "Скрипт исполняемый"  
fi

### 2.1.3 Логические операторы

**Операторы:**

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

**Примеры:**

# И (AND)  
if [ $age -ge 18 ] && [ $age -le 65 ]; then  
 echo "Трудоспособный возраст"  
fi  
  
# ИЛИ (OR)  
if [ "$name" = "Alice" ] || [ "$name" = "Bob" ]; then  
 echo "Известное имя"  
fi  
  
# НЕ (NOT)  
if [ ! -f "/tmp/file.txt" ]; then  
 echo "Файл не существует"  
fi

### 2.1.4 Оператор case

**Синтаксис:**

case $variable in  
 pattern1)  
 # команды  
 ;;  
 pattern2)  
 # команды  
 ;;  
 \*)  
 # команды по умолчанию  
 ;;  
esac

**Примеры:**

case $choice in  
 1)  
 echo "Выбрана опция 1"  
 ;;  
 2)  
 echo "Выбрана опция 2"  
 ;;  
 [3-5])  
 echo "Выбрана опция 3, 4 или 5"  
 ;;  
 \*)  
 echo "Неизвестная опция"  
 ;;  
esac

## 2.2 Циклы

### 2.2.1 Цикл for

**Синтаксис 1 (диапазон):**

for variable in list; do  
 # команды  
done

**Примеры:**

# Перебор чисел  
for i in 1 2 3 4 5; do  
 echo "Число: $i"  
done  
  
# Диапазон  
for i in {1..10}; do  
 echo $i  
done  
  
# С шагом  
for i in {0..20..2}; do  
 echo $i # 0, 2, 4, ..., 20  
done  
  
# Перебор файлов  
for file in \*.txt; do  
 echo "Обработка: $file"  
done  
  
# Перебор аргументов  
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:**

# Счетчик  
counter=1  
while [ $counter -le 5 ]; do  
 echo "Итерация: $counter"  
 ((counter++))  
done  
  
# Чтение файла построчно  
while IFS= read -r line; do  
 echo "Строка: $line"  
done < file.txt  
  
# Бесконечный цикл  
while true; do  
 echo "Нажмите Ctrl+C для выхода"  
 sleep 1  
done

### 2.2.3 Цикл until

**Синтаксис:**

until [ condition ]; do  
 # команды  
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 # выход при i=5  
 fi  
 echo $i  
done  
  
# continue  
for i in {1..10}; do  
 if [ $((i % 2)) -eq 0 ]; then  
 continue # пропустить четные  
 fi  
 echo $i  
done

## 2.3 Функции

### 2.3.1 Объявление функций

**Синтаксис 1:**

function\_name() {  
 # команды  
}

**Синтаксис 2:**

function function\_name {  
 # команды  
}

**Примеры:**

# Простая функция  
greet() {  
 echo "Привет!"  
}  
  
# Вызов функции  
greet  
  
# Функция с параметрами  
greet\_user() {  
 echo "Привет, $1!"  
}  
  
greet\_user "Alice" # Привет, 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"  
 global\_var="глобальная"  
 echo $local\_var  
}  
  
my\_function  
echo $global\_var # доступна  
echo $local\_var # не доступна (пустая)

### 2.3.4 Возврат значений

**Команда return:**

is\_even() {  
 if [ $(($1 % 2)) -eq 0 ]; then  
 return 0 # истина (четное)  
 else  
 return 1 # ложь (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  
  
# Проверка количества аргументов  
if [ $# -eq 0 ]; then  
 echo "Использование: $0 <аргументы>"  
 exit 1  
fi  
  
# Обработка всех аргументов  
for arg in "$@"; do  
 echo "Аргумент: $arg"  
done  
  
# Доступ к конкретным аргументам  
echo "Первый аргумент: $1"  
echo "Второй аргумент: $2"

### 2.4.2 Команда getopts

**Обработка опций:**

#!/bin/bash  
  
while getopts "a:b:c" opt; do  
 case $opt in  
 a)  
 echo "Опция -a с значением: $OPTARG"  
 ;;  
 b)  
 echo "Опция -b с значением: $OPTARG"  
 ;;  
 c)  
 echo "Опция -c"  
 ;;  
 \?)  
 echo "Неверная опция: -$OPTARG"  
 exit 1  
 ;;  
 esac  
done

## 2.5 Отладка скриптов

### 2.5.1 Опции отладки

**Команды:**

bash -x script.sh # трассировка выполнения  
bash -v script.sh # вывод строк скрипта  
bash -n script.sh # проверка синтаксиса

**В скрипте:**

#!/bin/bash  
  
set -x # включить трассировку  
# код  
set +x # выключить трассировку  
  
set -e # прервать при ошибке  
set -u # ошибка при использовании неопределенных переменных  
set -o pipefail # ошибка в конвейере

### 2.5.2 Вывод отладочной информации

**Примеры:**

# Вывод значений переменных  
echo "DEBUG: variable=$variable"  
  
# Проверка условий  
if [ condition ]; then  
 echo "DEBUG: Условие истинно"  
fi  
  
# Трассировка функций  
function\_name() {  
 echo "DEBUG: Вход в функцию с параметрами: $@"  
 # код 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:**

# Создание  
cat > hello.sh << 'EOF'  
#!/bin/bash  
echo "Hello, World!"  
EOF  
  
# Права на выполнение  
chmod +x hello.sh  
  
# Запуск  
./hello.sh

### 2.6.2 Способы запуска

**Различные способы:**

./script.sh # прямой запуск (требует chmod +x)  
bash script.sh # запуск через bash  
sh script.sh # запуск через sh  
source script.sh # выполнение в текущей оболочке  
. script.sh # то же что source

# 3. Выполнение лабораторной работы

## 3.1 Задание 1: Скрипт с getopts 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<шаблон> — указать шаблон для поиска - -C — различать большие и малые буквы - -n — выдавать номера строк

Содержимое скрипта показано на рисунке [Рисунок 1](#fig-content1).

|  |
| --- |
| Рисунок 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 2](#fig-content2).

|  |
| --- |
| Рисунок 2: Программа number\_check.c |

## 3.3 Задание 3: Управление файлами

Создан скрипт file\_manager.sh для создания и удаления пронумерованных файлов.

**Функциональность:** - Создание указанного количества файлов (1.tmp, 2.tmp, …, N.tmp) - Удаление всех созданных файлов - Использование опций командной строки -n и -d

Запуск скриптов показан на рисунке [Рисунок 3](#fig-run1).

|  |
| --- |
| Рисунок 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 4](#fig-run2).

|  |
| --- |
| Рисунок 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