

Отчет по лабораторной работе №9

Операционные системы

Фадин В.В.

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Написать скрипт, который при запуске будет делать резервную копию самого себя.	6
2.2	Написать пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Например, скрипт может последовательно распечатывать значения всех переданных аргументов.	8
2.3	Написать командный файл — аналог команды ls (без использования самой этой команды и команды dir). Требуется, чтобы он выдавал информацию о нужном каталоге и выводил информацию о возможностях доступа к файлам этого каталога.	10
2.4	Написать командный файл, который получает в качестве аргумента командной строки формат файла (.txt, .doc, .jpg, .pdf и т.д.) и вычисляет количество таких файлов в указанной директории. Путь к директории также передаётся в виде аргумента командной строки	12
3	Выводы	14
4	Ответы на контрольные вопросы	15

Список иллюстраций

2.1	backup_self.sh	6
2.2	backup_self.sh	7
2.3	process_args.sh	8
2.4	process_args.sh	9
2.5	mysl.sh	10
2.6	mysl.sh	11
2.7	mysl.sh	12
2.8	mysl.sh	13


Список таблиц

1 Цель работы

Изучить основы программирования в оболочке ОС UNIX/Linux. Научиться писать небольшие командные файлы.

2 Выполнение лабораторной работы

2.1 Написать скрипт, который при запуске будет делать резервную копию самого себя.



```
GNU nano 7.2 backup_self.sh
#!/bin/bash

echo "Starting backup script..."

SCRIPT_FILE=$(basename "$0")
echo "Script file name: $SCRIPT_FILE"

mkdir -p ~/backup
echo "Backup directory created: ~/backup"

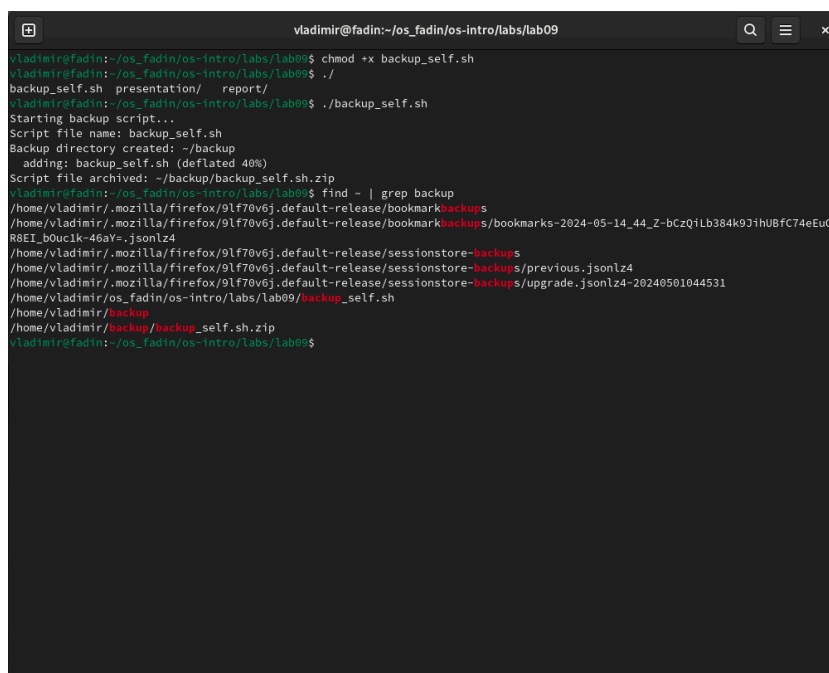
zip ~/backup/"$SCRIPT_FILE".zip "$0"
echo "Script file archived: ~/backup/$SCRIPT_FILE.zip"
```

Рис. 2.1: backup_self.sh

Тут происходит следующее: 1. `SCRIPT_FILE=$(basename "$0")`: эта строка получает текущее имя файла сценария с помощью команды `basename`, которая удаляет путь к каталогу из имени файла. 2. `mkdir -p ~/backup`: эта строка создает каталог `backup` в вашем домашнем каталоге, если он еще не существует. Опция `-p` указывает `mkdir` создать каталог с родителями,

то есть каталог будет создан, даже если родительский каталог не существует. 3. `zip ~/backup/"$SCRIPT_FILE".zip "$0"`: эта строка архивирует файл сценария с помощью `zip`. `$0` относится к текущему файлу сценария, а `~/backup/"$SCRIPT_FILE".zip` — это имя выходного файла. Команда `zip` создаст `zip`-архив файла сценария и сохранит его в каталоге `backup`.

Чтобы использовать этот скрипт, сохраним его в файл (например, `backup_self.sh`), сделаем файл исполняемым с помощью `chmod +x backup_self.sh`, а затем запустим его с `./backup_self.sh`.



```
vladimir@fadin:~/os_fadin/os-intro/labs/lab09
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ chmod +x backup_self.sh
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ ./
backup_self.sh presentation/ report/
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ ./backup_self.sh
Starting backup script...
Script file name: backup_self.sh
Backup directory created: ~/backup
  adding: backup_self.sh (deflated 40%)
Script file archived: ~/backup/backup_self.sh.zip
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ find - | grep backup
/home/vladimir/.mozilla/firefox/9lf70v6j.default-release/bookmarksbackups
/home/vladimir/.mozilla/firefox/9lf70v6j.default-release/bookmarks-2024-05-14_44_Z-bCzQiLb384k9JihUBfC74eEu0
R8EI_b0ucIk-46av-.jsonlz4
/home/vladimir/.mozilla/firefox/9lf70v6j.default-release/sessionstore-backups
/home/vladimir/.mozilla/firefox/9lf70v6j.default-release/sessionstore-backups/previous.jsonlz4
/home/vladimir/.mozilla/firefox/9lf70v6j.default-release/sessionstore-backups/upgrade.jsonlz4-20240501044531
/home/vladimir/os_fadin/os-intro/labs/lab09/backup_self.sh
/home/vladimir/backup
/home/vladimir/backup/backup_self.sh.zip
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$
```

Рис. 2.2: `backup_self.sh`

2.2 Написать пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Например, скрипт может последовательно распечатывать значения всех переданных аргументов.



```
GNU nano 7.2 process_args.sh
#!/bin/bash

echo "Процесс скрипта с аргументами..."

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

Рис. 2.3: process_args.sh

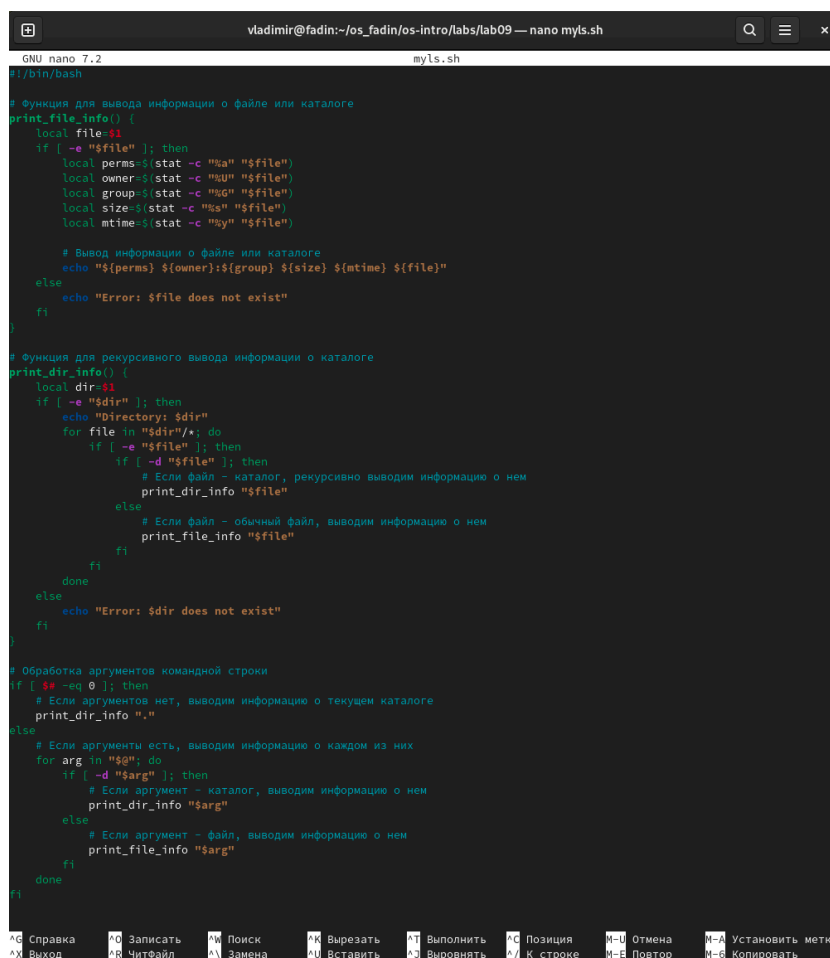
В этом примере `$@` — это специальная переменная `bash`, содержащая все аргументы командной строки. Цикл `for` проходит по каждому аргументу и выводит его значение с помощью `echo`.


```
vladimir@fadin:~/os_fadin/os-intro/labs/lab09
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ nano process_args.sh
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ ls -l
итого 8
-rwxr-xr-x. 1 vladimir vladimir 268 мая 16 19:16 backup_self.sh
drwxr-xr-x. 1 vladimir vladimir 56 мая 14 21:04 presentation
-rwxr-xr-x. 1 vladimir vladimir 140 мая 16 19:28 process_args.sh
drwxr-xr-x. 1 vladimir vladimir 62 мая 14 21:04 report
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ ./pr
presentation/ process_args.sh
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ ./process_args.sh
Процесс скрипта с аргументами...
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ ./process_args.sh arg1 arg2 arg3 arg4 arg5 arg6 arg7 arg8 arg9 arg10 arg11 ar
g12
Процесс скрипта с аргументами...
Аргумент: arg1
Аргумент: arg2
Аргумент: arg3
Аргумент: arg4
Аргумент: arg5
Аргумент: arg6
Аргумент: arg7
Аргумент: arg8
Аргумент: arg9
Аргумент: arg10
Аргумент: arg11
Аргумент: arg12
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$
```

Рис. 2.4: process_args.sh

2.3 Написать командный файл — аналог команды ls (без использования самой этой команды и команды dir).

Требуется, чтобы он выдавал информацию о нужном каталоге и выводил информацию о возможностях доступа к файлам этого каталога.



```
GNU nano 7.2 myls.sh
#!/bin/bash

# Функция для вывода информации о файле или каталоге
print_file_info() {
    local file=$1
    if [ -e "$file" ]; then
        local perms=$(stat -c "%a" "$file")
        local owner=$(stat -c "%u" "$file")
        local group=$(stat -c "%g" "$file")
        local size=$(stat -c "%s" "$file")
        local mtime=$(stat -c "%y" "$file")

        # Вывод информации о файле или каталоге
        echo "${perms} ${owner}:${group} ${size} ${mtime} ${file}"
    else
        echo "Error: $file does not exist"
    fi
}

# Функция для рекурсивного вывода информации о каталоге
print_dir_info() {
    local dir=$1
    if [ -e "$dir" ]; then
        echo "Directory: $dir"
        for file in "$dir"/*; do
            if [ -e "$file" ]; then
                if [ -d "$file" ]; then
                    # Если файл - каталог, рекурсивно выводим информацию о нем
                    print_dir_info "$file"
                else
                    # Если файл - обычный файл, выводим информацию о нем
                    print_file_info "$file"
                fi
            fi
        done
    else
        echo "Error: $dir does not exist"
    fi
}

# Обработка аргументов командной строки
if [ $# -eq 0 ]; then
    # Если аргументов нет, выводим информацию о текущем каталоге
    print_dir_info "."
else
    # Если аргументы есть, выводим информацию о каждом из них
    for arg in "$@"; do
        if [ -d "$arg" ]; then
            # Если аргумент - каталог, выводим информацию о нем
            print_dir_info "$arg"
        else
            # Если аргумент - файл, выводим информацию о нем
            print_file_info "$arg"
        fi
    done
fi
```

Рис. 2.5: myls.sh

```
vladimir@fadin:~/os_fadin/os-intro/labs/lab09
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ ./mys.sh ./path
Error: ./path does not exist
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ mkdir path
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ touch path/file.txt
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ ./mys.sh ./path
Directory: ./path
644 vladimir:vladimir 0 2024-05-16 19:47:14.796491684 +0300 ./path/file.txt
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ mkdir path/to
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ mkdir path/to/directory
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ touch path/to/directory/newfile.txt
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ ./mys.sh ./path/
file.txt to/
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ ./mys.sh ./path/to/directory
Directory: ./path/to/directory
644 vladimir:vladimir 0 2024-05-16 19:49:09.183200213 +0300 ./path/to/directory/newfile.txt
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ nano
backup_self.sh  myls.sh      path/      presentation/  process_args.sh  report/
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ nano myls.sh
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$
```

Рис. 2.6: myls.sh

2.4 Написать командный файл, который получает в качестве аргумента командной строки формат файла (.txt, .doc, .jpg, .pdf и т.д.) и вычисляет количество таких файлов в указанной директории. Путь к директории также передаётся в виде аргумента командной строки

В этом примере мы используем команду `find` для поиска файлов с расширением в указанной директории. Опция `-type f` указывает, что мы ищем только файлы, а не каталоги. Опция `-name "*. $file_extension"` указывает, что мы ищем файлы с именем, заканчивающимся указанным расширением. Результат поиска приводится в команду `wc -l`, которая вычисляет количество строк в выводе, что в данном случае равно количеству файлов с расширениями запроса.



```
vladimir@fadin:~/os_fadin/os-intro/labs/lab09 — nano count_files.sh
GNU nano 7.2 count_files.sh
#!/bin/bash

# Проверка наличия аргументов
if [ $# -ne 2 ]; then
    echo "Usage: $0 <directory> <file_extension>"
    exit 1
fi

# Получение аргументов
directory=$1
file_extension=$2

# Вычисление количества файлов с указанным расширением
count=$(find "$directory" -type f -name ".*$file_extension" | wc -l)

# Вывод результата
echo "Number of files with extension .$file_extension in $directory: $count"
```

Рис. 2.7: myls.sh

```
vladimir@fadin:~/os_fadin/os-intro/labs/lab09
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ ls path/to/directory/
newfile2.txt  newfile.txt
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ ./count_files.sh ./path/to/directory txt
Number of files with extension .txt in ./path/to/directory: 2
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ ls path/
file.txt  to
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$ ./count_files.sh ./path txt
Number of files with extension .txt in ./path: 3
vladimir@fadin:~/os_fadin/os-intro/labs/lab09$
```

Рис. 2.8: myls.sh

3 Выводы

Изучили основы программирования в оболочке ОС UNIX/Linux. Научились писать небольшие командные файлы.

4 Ответы на контрольные вопросы

1. **Что такое командная оболочка? Приведите примеры командных оболочек. Чем они отличаются?** Командная оболочка — это программа, которая позволяет пользователям взаимодействовать с операционной системой путем выполнения команд, сценариев и программ. Примеры командных оболочек: Bash, pdksh, tcsh и zsh. Они различаются синтаксисом, функциональностью и возможностями, такими как завершение команд, история и возможности создания сценариев.
2. **Что такое POSIX?** POSIX (Portable Operating System Interface) — это набор стандартов для операционных систем, включая Unix и Linux, определяющий общий API для взаимодействия с операционной системой. Это обеспечивает совместимость и переносимость программного обеспечения между различными системами.
3. **Как определяются переменные и массивы в Bash?** В Bash переменные определяются с помощью оператора =, а массивы определяются с помощью круглых скобок () и запятых , для разделения элементов.
4. **Какова цель операторов let и read?** Оператор let используется для выполнения арифметических операций, а оператор read используется для чтения ввода от пользователя или файла.
5. **Какие арифметические операции можно выполнять в Bash?** Bash поддерживает базовые арифметические операции, такие как сложение, вычитание, умножение и деление, а также более сложные операции, такие как по модулю и возведение в степень.

6. **Что означает операция (())?** Операция (()) используется для выполнения арифметических операций и вычисления выражений в Bash.
7. **Какие стандартные имена переменных вы знаете?** Стандартные имена переменных в Bash включают, среди прочего, SHELL, PATH, HOME и USER.
8. **Что такое метасимволы?** Метасимволы — это специальные символы в Bash, имеющие определенное значение, например *, ? и [, которые используются для сопоставления с образцом и подстановки под шаблон.
9. **Как избежать метасимволов?** Метасимволы можно экранировать с помощью обратной косой черты \ или заключая их в кавычки.
10. **Как создавать и запускать командные файлы?** Командные файлы, также известные как сценарии, можно создавать с помощью текстового редактора и запускать с использованием нотации ./, за которой следует имя сценария.
11. **Как определяются функции в Bash?** Функции в Bash определяются с помощью ключевого слова function, за которым следует имя функции и аргументы в круглых скобках.
12. **Как определить, является ли файл каталогом или обычным файлом?** Вы можете использовать команду test или оператор [, чтобы определить, является ли файл каталогом или обычным файлом, используя параметры -d и -f соответственно.
13. **Какова цель команд set, typeset и unset?** Команда set используется для установки параметров оболочки, typeset используется для объявления переменных, а unset используется для удаления переменных или функций.
14. **Как параметры передаются в командные файлы?** Параметры можно передавать в командные файлы с использованием синтаксиса \$1, \$2 и т. д., который представляет первый, второй и т. д. аргумент командной строки.
15. **Что такое специальные переменные в Bash и каково их назначение?** К специальным переменным в Bash относятся, среди прочего, SHELL,

PATH, HOME и USER, которые используются для хранения информации об оболочке и среде пользователя.