

Лабораторная работа №12

**Программирование в командном процессоре ОС UNIX. Командные
файлы**

Полякова Юлия Александровна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Контрольные вопросы	11
5	Вывод	16

Список иллюстраций

3.1	Листинг script1.sh	7
3.2	Запуск script1.sh	7
3.3	Архив в backup	8
3.4	Листинг script2.sh	8
3.5	Запуск script2.sh	8
3.6	Листинг script3.sh	9
3.7	Запуск script3.sh	9
3.8	Листинг script4.sh	10
3.9	Запуск script4.sh	10

Список таблиц

1 Цель работы

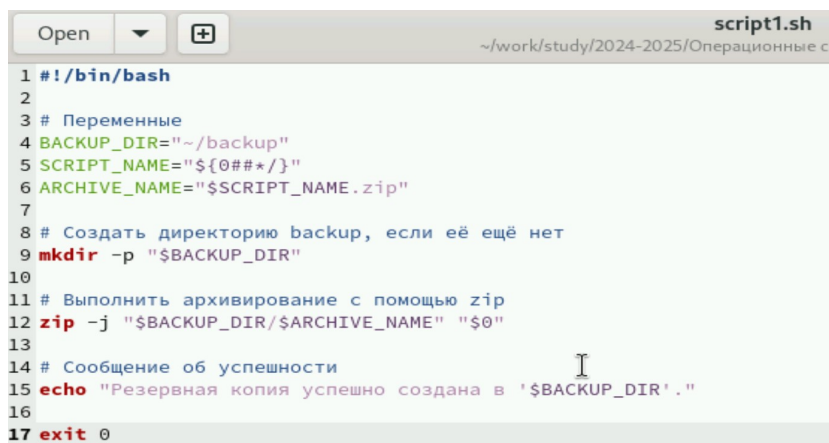
Изучить основы программирования в оболочке ОС UNIX/Linux. Научиться писать небольшие командные файлы.

2 Задание

1. Ознакомиться с теоретическим материалом.
2. Написать 4 командных файла по заданию.
3. Ответить на контрольные вопросы.

3 Выполнение лабораторной работы

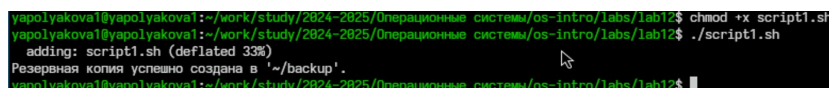
1. Для начала командой `touch` создаем 4 файла с расширением `.sh`. Рассмотрим первый скрипт. При запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию `backup`. Архивируется архиватором `zip`. (рис. 3.1).



```
1 #!/bin/bash
2
3 # Переменные
4 BACKUP_DIR=~/.backup
5 SCRIPT_NAME="${0##*/}"
6 ARCHIVE_NAME="$SCRIPT_NAME.zip"
7
8 # Создать директорию backup, если её ещё нет
9 mkdir -p "$BACKUP_DIR"
10
11 # Выполнить архивирование с помощью zip
12 zip -j "$BACKUP_DIR/$ARCHIVE_NAME" "$0"
13
14 # Сообщение об успешности
15 echo "Резервная копия успешно создана в '$BACKUP_DIR'."
16
17 exit 0
```

Рис. 3.1: Листинг script1.sh

2. Чтобы запустить файл нужно дать ему доступ на исполняемость. Как видно по сообщению, резервная копия успешно создалась. (рис. 3.2)



```
yapolyakova1@yapolyakova1:~/work/study/2024-2025/Операционные системы/os-intro/labs/lab12$ chmod +x script1.sh
yapolyakova1@yapolyakova1:~/work/study/2024-2025/Операционные системы/os-intro/labs/lab12$ ./script1.sh
adding: script1.sh (deflated 33%)
Резервная копия успешно создана в '~/.backup'.
yapolyakova1@yapolyakova1:~/work/study/2024-2025/Операционные системы/os-intro/labs/lab12$
```

Рис. 3.2: Запуск script1.sh

3. Файл сархивировался в этот каталог. (рис. 3.3)

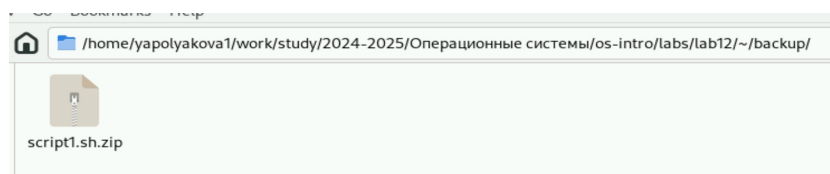


Рис. 3.3: Архив в backup

4. Рассмотрим второй скрипт. Обрабатывает любое произвольное число аргументов командной строки, в том числе превышающее десять. Последовательно распечатывает значения всех переданных аргументов в цикле (рис. 3.4)

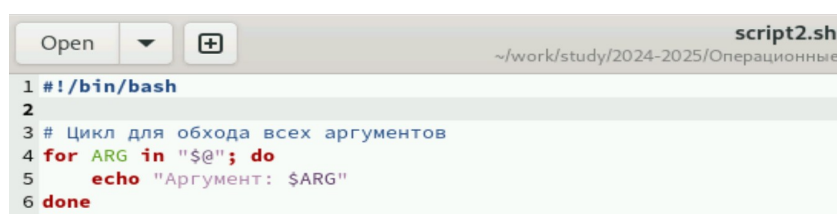


Рис. 3.4: Листинг script2.sh

5. Даем доступ на исполняемость и запускаем с большим кол-вом аргументов (рис. 3.5)

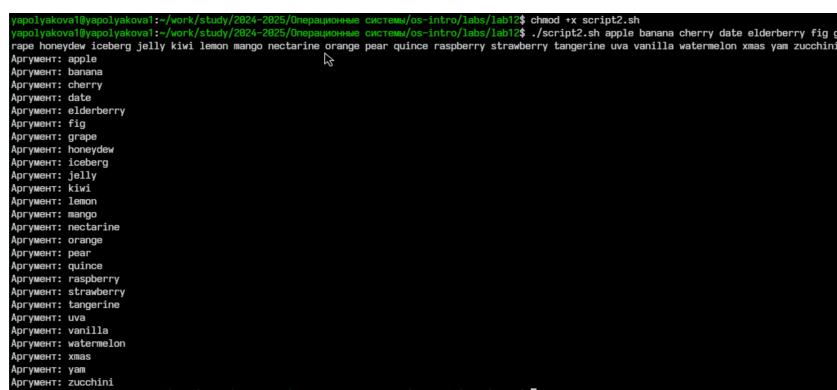
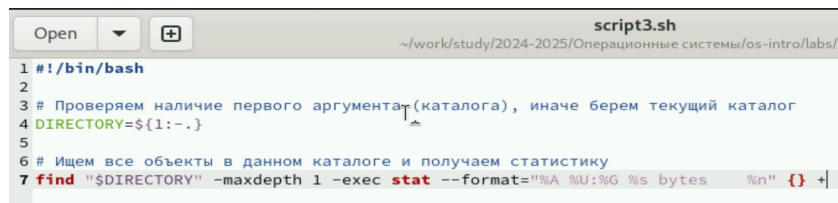


Рис. 3.5: Запуск script2.sh

6. Рассмотрим третий скрипт. Это командный файл — аналог команды ls (без использования самой этой команды и команды dir). Выдает информацию о

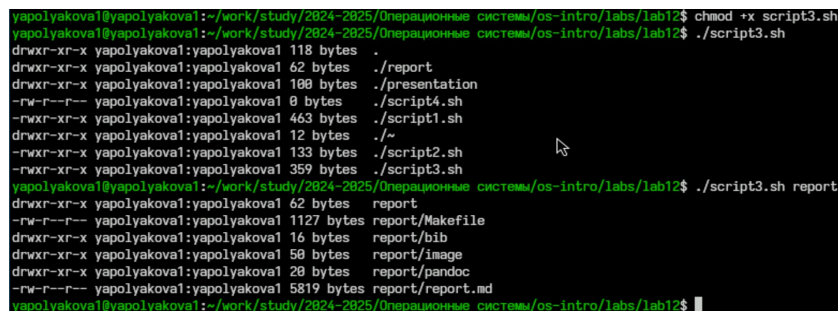
нужном каталоге и выводит информацию о возможностях доступа к файлам этого каталога (рис. 3.6)



```
script3.sh
~/work/study/2024-2025/Операционные системы/os-intro/labs/l
1 #!/bin/bash
2
3 # Проверяем наличие первого аргумента (каталога), иначе берем текущий каталог
4 DIRECTORY=${1:-.}
5
6 # Ищем все объекты в данном каталоге и получаем статистику
7 find "$DIRECTORY" -maxdepth 1 -exec stat --format="%A %U:%G %s bytes" %n" {} + |
```

Рис. 3.6: Листинг script3.sh

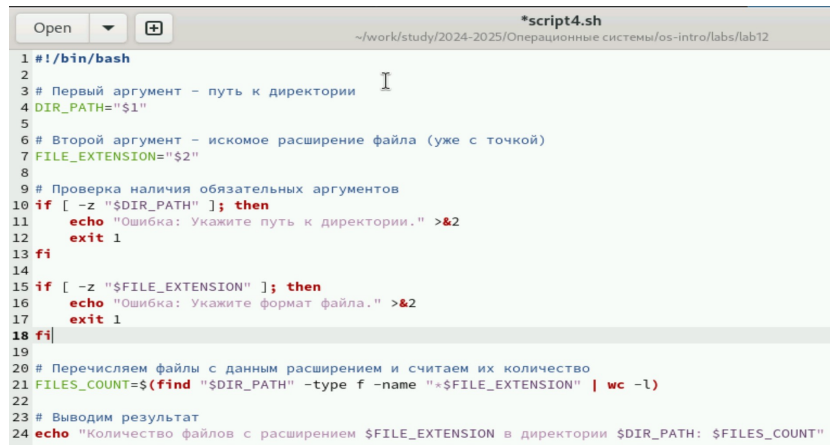
7. Даем доступ к исполнению. Если не указать аргумент, то выведутся данные текущего каталога, если указать, то указанного (рис. 3.7)



```
yapolyakova1@yapolyakova1:~/work/study/2024-2025/Операционные системы/os-intro/labs/lab12$ chmod +x script3.sh
yapolyakova1@yapolyakova1:~/work/study/2024-2025/Операционные системы/os-intro/labs/lab12$ ./script3.sh
drwxr-xr-x yapolyakova1:yapolyakova1 118 bytes .
drwxr-xr-x yapolyakova1:yapolyakova1 62 bytes ./report
drwxr-xr-x yapolyakova1:yapolyakova1 100 bytes ./presentation
-rw-r--r-- yapolyakova1:yapolyakova1 8 bytes ./script4.sh
-rwxr-xr-x yapolyakova1:yapolyakova1 463 bytes ./script1.sh
drwxr-xr-x yapolyakova1:yapolyakova1 12 bytes ./~
-rwxr-xr-x yapolyakova1:yapolyakova1 133 bytes ./script2.sh
-rwxr-xr-x yapolyakova1:yapolyakova1 359 bytes ./script3.sh
yapolyakova1@yapolyakova1:~/work/study/2024-2025/Операционные системы/os-intro/labs/lab12$ ./script3.sh report
drwxr-xr-x yapolyakova1:yapolyakova1 62 bytes report
-rw-r--r-- yapolyakova1:yapolyakova1 1127 bytes report/Makefile
drwxr-xr-x yapolyakova1:yapolyakova1 16 bytes report/bib
drwxr-xr-x yapolyakova1:yapolyakova1 50 bytes report/image
drwxr-xr-x yapolyakova1:yapolyakova1 20 bytes report/pandoc
-rw-r--r-- yapolyakova1:yapolyakova1 5819 bytes report/report.md
yapolyakova1@yapolyakova1:~/work/study/2024-2025/Операционные системы/os-intro/labs/lab12$
```

Рис. 3.7: Запуск script3.sh

8. Рассмотрим четвертый скрипт. Командный файл, который получает в качестве аргумента командной строки формат файла (.txt, .doc, .jpg, .pdf и т.д., но сразу с точкой) и вычисляет количество таких файлов в указанной директории. Путь к директории также передаётся в виде аргумента командной строки. Также я решила добавить проверку на наличие аргументов. (рис. 3.8)



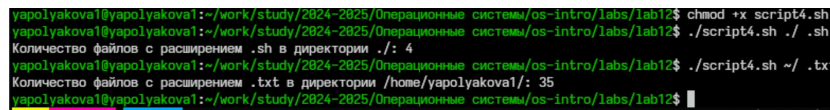
```

1 #!/bin/bash
2
3 # Первый аргумент - путь к директории
4 DIR_PATH="$1"
5
6 # Второй аргумент - искомое расширение файла (уже с точкой)
7 FILE_EXTENSION="$2"
8
9 # Проверка наличия обязательных аргументов
10 if [ -z "$DIR_PATH" ]; then
11     echo "Ошибка: Укажите путь к директории." >&2
12     exit 1
13 fi
14
15 if [ -z "$FILE_EXTENSION" ]; then
16     echo "Ошибка: Укажите формат файла." >&2
17     exit 1
18 fi
19
20 # Перечисляем файлы с данным расширением и считаем их количество
21 FILES_COUNT=$(find "$DIR_PATH" -type f -name "$FILE_EXTENSION" | wc -l)
22
23 # Выводим результат
24 echo "Количество файлов с расширением $FILE_EXTENSION в директории $DIR_PATH: $FILES_COUNT"

```

Рис. 3.8: Листинг script4.sh

9. Даем возможность исполнения, запускаем с различными форматами и каталогами. (рис. 3.9)



```

yarolyakova1@yarolyakova1:~/work/study/2024-2025/Операционные системы/os-intro/labs/lab12$ chmod +x script4.sh
yarolyakova1@yarolyakova1:~/work/study/2024-2025/Операционные системы/os-intro/labs/lab12$ ./script4.sh ./ .sh
Количество файлов с расширением .sh в директории ./: 4
yarolyakova1@yarolyakova1:~/work/study/2024-2025/Операционные системы/os-intro/labs/lab12$ ./script4.sh ~/ .txt
Количество файлов с расширением .txt в директории /home/yarolyakova1/: 35
yarolyakova1@yarolyakova1:~/work/study/2024-2025/Операционные системы/os-intro/labs/lab12$

```

Рис. 3.9: Запуск script4.sh

4 Контрольные вопросы

1. Объясните понятие командной оболочки. Приведите примеры командных оболочек. Чем они отличаются?

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек:

- оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций;
- C-оболочка (или csh) — надстройка на оболочкой Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд;
- оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна;
- BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation).

Основные отличия:

- Синтаксис и особенности: Каждая оболочка имеет уникальный набор возможностей и особенностей, такие как поддержка встроенных функций, автозавершение команд, управление историей команд и другие.

- Совместимость: Некоторые оболочки строго следуют стандартам POSIX, тогда как другие добавляют собственные расширения.
- Производительность: Различные оболочки могут иметь разную производительность при выполнении определённых операций

2. Что такое POSIX?

POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Керна.

3. Как определяются переменные и массивы в языке программирования bash?

- Переменная определяется следующим образом: `variable=value`
- Массив объявляется следующим образом: `array=(value1 value2 ...)`
- Доступ к элементам массива осуществляется через индекс: `echo ${array[index]}`

4. Каково назначение операторов `let` и `read`?

Оператор `let` используется для вычисления выражений и присваивания результата переменной:

`let variable=expression` или `((variable = expression))`

Оператор `read` предназначен для чтения строки из стандартного ввода и сохранения её в переменную: `read variable`

5. Какие арифметические операции можно применять в языке программирования bash?

- сложение (+)
- вычитание (-)
- умножение (*)
- деление (/)
- остаток от деления (%)
- возведение в степень (**)

Пример: `result=$((a + b))`

6. Что означает операция (())?

Операция (()) позволяет выполнять арифметическое выражение внутри скобок и автоматически возвращает результат.

Например: `((result = a * b))`

Это эквивалентно использованию оператора `let`, но удобнее и нагляднее.

7. Какие стандартные имена переменных Вам известны?

- `$HOME`: домашний каталог текущего пользователя.
- `$PATH`: список путей для поиска исполняемых файлов.
- `$USER`: имя текущего пользователя.
- `$SHELL`: используемая командная оболочка.
- `$PWD`: текущий рабочий каталог.
- `$PS1`: строка приглашения.

8. Что такое метасимволы?

Метасимволы — это символы, имеющие специальное значение в командной строке. Они используются для обозначения шаблонов имен файлов, перенаправлений ввода-вывода и специальных действий. Примеры метасимволов:

- `*`
- `?`

- []
- < > | &;

9. Как экранировать метасимволы?

Экранирование выполняется с использованием обратного слэша () или двойных кавычек ("). Пример:

```
ls file\* # выводит все файлы начинающиеся с 'file'
```

10. Как создавать и запускать командные файлы?

Сначала нужно создать файл с расширением .sh, затем добавить код и сделать файл исполняемым:

```
chmod +x script.sh
```

Чтобы запустить из текущего каталога: ./script.sh, если в другом, то полный путь.

11. Как определяются функции в языке программирования bash?

```
function_name() {
    commands
}
```

Пример:

```
hello_world() {
    echo "Hello World!"
}
```

12. Каким образом можно выяснить, является файл каталогом или обычным файлом?

Использовать команду test или оператор [[]]. Например:

```
if [[ -d "$filename" ]]; then
    echo "Файл является каталогом"
fi
```

13. Каково назначение команд set, typeset и unset?

- Команда set устанавливает значения глобальных переменных окружения.
- Команда typeset (аналогична declare) создаёт локальные переменные и задаёт атрибуты переменным.
- Команда unset удаляет переменные или функции.

14. Как передаются параметры в командные файлы?

Параметры передаются следующим образом:

```
./script.sh arg1 arg2 ...
```

Доступ к параметрам осуществляется через позиционные переменные: \$1, \$2, ..., \$n

15. Назовите специальные переменные языка bash и их назначение.

- \$#: количество аргументов.
- \$@: аргументы команды как отдельные элементы.
- \$*: аргументы команды как одна строка.
- \$\$: PID процесса оболочки.
- \$!: PID последнего запущенного фона.
- \$? : статус завершения последней выполненной команды.

5 Вывод

Были изучены основы программирования в оболочке ОС UNIX/Linux. Также мы научились писать небольшие командные файлы.