

# **Отчет по лабораторной работе № 12**

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

Скобеева Алиса Алексеевна

# Содержание

|          |                                       |           |
|----------|---------------------------------------|-----------|
| <b>1</b> | <b>Цель работы</b>                    | <b>5</b>  |
| <b>2</b> | <b>Задание</b>                        | <b>6</b>  |
| <b>3</b> | <b>Выполнение лабораторной работы</b> | <b>7</b>  |
| <b>4</b> | <b>Выводы</b>                         | <b>12</b> |
| <b>5</b> | <b>Ответы на контрольные вопросы</b>  | <b>13</b> |

## Список иллюстраций

|     |                                      |    |
|-----|--------------------------------------|----|
| 3.1 | Пишем скрипт . . . . .               | 7  |
| 3.2 | Ввод необходимых команд . . . . .    | 7  |
| 3.3 | Архив создан . . . . .               | 8  |
| 3.4 | Пишем скрипт . . . . .               | 8  |
| 3.5 | Все работает корректно . . . . .     | 9  |
| 3.6 | Пишем скрипт . . . . .               | 9  |
| 3.7 | Файл выводит все корректно . . . . . | 10 |
| 3.8 | Пишем скрипт . . . . .               | 10 |
| 3.9 | Все работает корректно . . . . .     | 11 |

## **Список таблиц**

# 1 Цель работы

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

## 2 Задание

Написать несколько командных файлов.

### 3 Выполнение лабораторной работы

Создаем файл task1.sh. Пишем скрипт, который при запуске будет делать резервную копию самого себя в другую директорию backup. При этом файл архивируется.

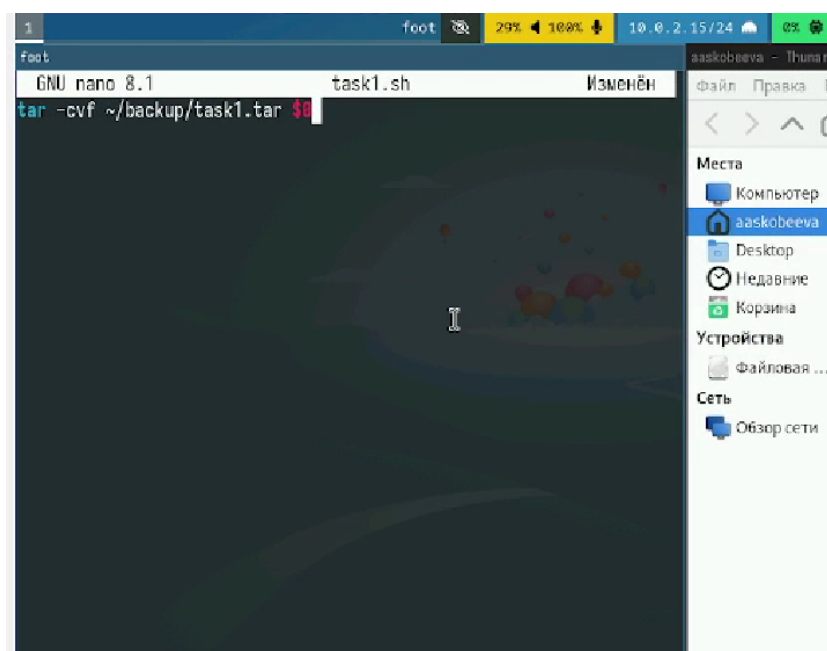


Рис. 3.1: Пишем скрипт

Запускаем файл

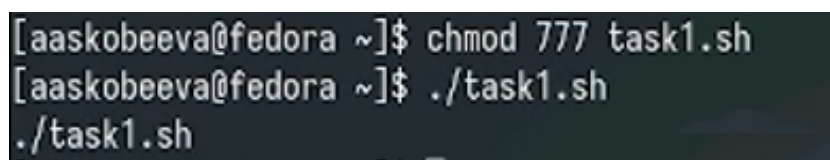


Рис. 3.2: Ввод необходимых команд

Проверяем корректность работы файла

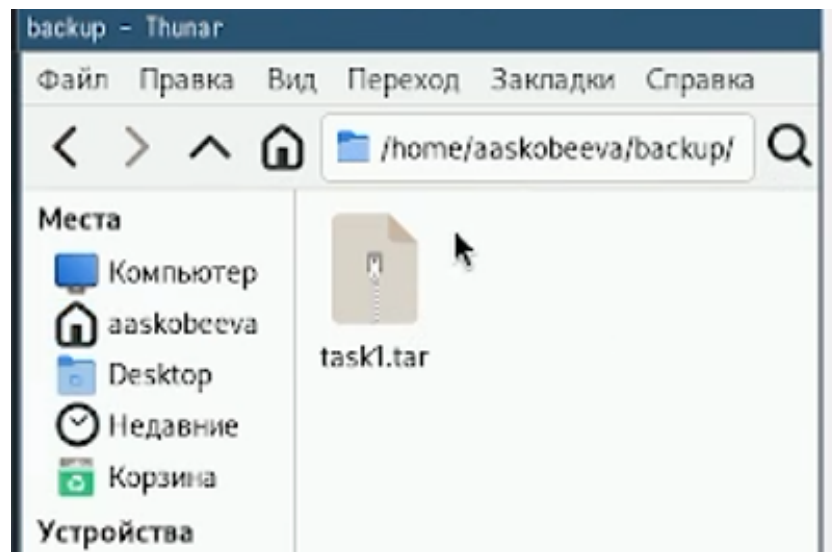


Рис. 3.3: Архив создан

Создаем файл task2.sh Пишем командный файл, обрабатывающий любое число аргументов командной строки. Скрипт последовательно распечатывает значения всех переданных аргументов

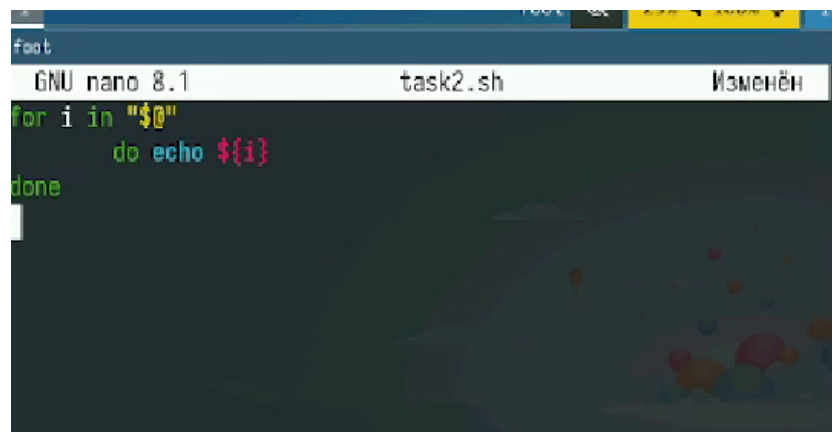


Рис. 3.4: Пишем скрипт

Проверяем корректность работы файла



```
[aaskobeeva@fedora ~]$ ./task2.sh apple cat house dog flower
apple
cat
house
dog
flower
[aaskobeeva@fedora ~]$
```

Рис. 3.5: Все работает корректно

Создаем файл task3.sh и пишем аналог команды ls

```
foot
GNU nano 8.1 task3.sh Изменён
echo "$1/ " | tr -d "\n";
stat --printf "%A" "$1/";
echo
for i in $1/*
do echo "${i} " | tr -d "\n";
stat --printf "%A" "${i}";
echo
done
```

Рис. 3.6: Пишем скрипт

Проверяем корректность работы файла

```
[aaskobeeva@fedora ~]$ ./task3.sh ~
/home/aaskobeeva/ drwx-----
/home/aaskobeeva/#1# -rw-r--r--
/home/aaskobeeva/#2# -rw-r--r--
/home/aaskobeeva/#3# -rw-r--r--
/home/aaskobeeva/#4# -rw-r--r--
/home/aaskobeeva/abc1 -rw-r--r--
/home/aaskobeeva/australia -rwxr-xr--
/home/aaskobeeva/backup drwxr-xr-x
/home/aaskobeeva/conf.txt -rw-r--r--
/home/aaskobeeva/Desktop drwxr-xr-x
/home/aaskobeeva/Documents drwxr-xr-x
/home/aaskobeeva/Downloads drwxr-xr-x
/home/aaskobeeva/feathers -rw-rw-r--
/home/aaskobeeva/file.txt -rw-r--r--
/home/aaskobeeva/git-extended drwxr-xr-x
/home/aaskobeeva/lab07.sh -rw-r--r--
/home/aaskobeeva/lab07.sh~ -rw-r--r--
/home/aaskobeeva/laba71 drwxr-xr-x
```

Рис. 3.7: Файл выводит все корректно

Создаем файл task4.sh и пишем скрипт, который вычисляет кол-во файлов указанного формата в указанной директории

```
GNU nano 8.1 task4.sh
let COUNT=0
for i in $2/*. $1
do let COUNT++
done
echo $COUNT
```

Рис. 3.8: Пишем скрипт

Проверяем работу

```
[aaskobeeva@fedora ~]$ ./task4.sh txt ~  
3  
[aaskobeeva@fedora ~]$
```

Рис. 3.9: Все работает корректно

## 4 Выводы

Мы научились писать небольшие команды и успешно выполнили все задания лабораторной работы.

## 5 Ответы на контрольные вопросы

1. Командная оболочка: Интерфейс между пользователем и ядром ОС. Примеры: bash, zsh, fish. Отличаются синтаксисом, функциональностью, и наличием дополнительных возможностей.
2. POSIX: Семейство стандартов, определяющих интерфейсы операционных систем. Гарантирует переносимость программ.
3. Переменные/Массивы в bash:
  - Переменные: var="value"
  - Массивы: array=(item1 item2 item3)
4. let/read:
  - let: Вычисление арифметических выражений.
  - read: Чтение ввода пользователя.
5. Арифметические операции в bash: +, -, \*, /, % (возведение в степень).
6. (( )): Обозначает арифметическое выражение.
7. Стандартные переменные: HOME, PATH, USER, PWD, SHELL, PS1 (строка приглашения).
8. Метасимволы: Символы, имеющие специальное значение для оболочки (например, \*, ?, |, >, <).
9. Экранирование: Обратный слеш перед метасимволом или заключение в кавычки.
10. Создание/Запуск командных файлов:

- Создание: `touch script.sh`, редактирование с помощью текстового редактора.
- Запуск: `bash script.sh` или `./script.sh` (если файл исполняемый).

#### 11. Функции в bash:

```
Bash function function_name() { # Команды }
```

#### 12. Проверка типа файла: `if [ -d "file" ]` (каталог) или `if [ -f "file" ]` (обычный файл).

#### 13. `set/typeset/unset`:

- `set`: Устанавливает или отображает переменные оболочки.
- `typeset`: Объявляет тип переменной (например, `integer`).
- `unset`: Удаляет переменную.

#### 14. Передача параметров: `./script.sh arg1 arg2`, параметры доступны как `$1`, `$2` и т.д.

#### 15. Специальные переменные:

- `$0`: Имя скрипта
- `$#`: Количество параметров
- `$@`: Все параметры
- `$?`: Код возврата последней команды
- `$$`: PID текущего процесса
- `$_`: PID последнего запущенного фонового процесса