

# **Лабораторная работа №12**

**Отчет**

Устинова Виктория Вадимовна

# 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>11</b>
4.1	Ответы на контрольные вопросы . . . . .	11

## Список иллюстраций

3.1	Написали скрипт . . . . .	7
3.2	Вводим команду ./task1.sh . . . . .	7
3.3	Перешли с созданную папку backup и разархивировали файл . . .	8
3.4	Редактор nano . . . . .	8
3.5	Пишем код . . . . .	8
3.6	У нас выводит последовательно, как и требуется . . . . .	9
3.7	Пишем код . . . . .	9
3.8	Выводит необходимое . . . . .	9
3.9	Пишем код . . . . .	10
3.10	Выводит необходимое, я проверила, ровно 4 файла txt находятся у меня в домашней директории . . . . .	10

## **Список таблиц**

# 1 Цель работы

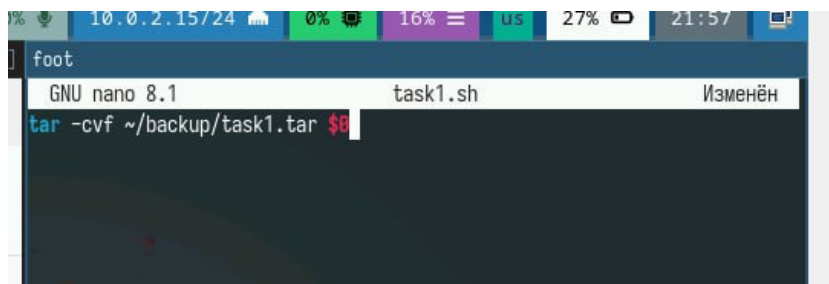
Изучить основы программирования в оболочке ОС UNIX/Linux. Научиться писать небольшие командные файлы.

## 2 Задание

1. Написать скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию backup в вашем домашнем каталоге. При этом файл должен архивироваться одним из архиваторов на выбор zip, bzip2 или tar. Способ использования команд архивации необходимо узнать, изучив справку.
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.
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.
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.

### 3 Выполнение лабораторной работы

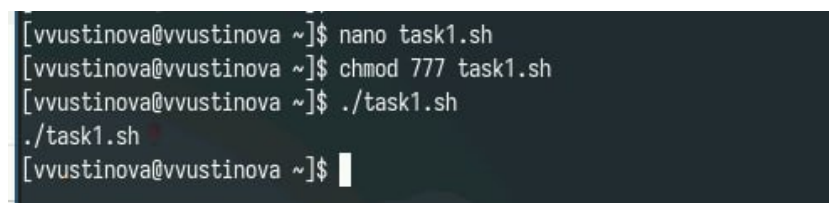
Написать скрипт, который будет делать резервную копию самого себя(рис. 3.1).



```
foot
GNU nano 8.1 task1.sh Изменён
tar -cvf ~/backup/task1.tar $0
```

Рис. 3.1: Написали скрипт

Запускаем файл(рис. 3.2).



```
[vvustinova@vvustinova ~]$ nano task1.sh
[vvustinova@vvustinova ~]$ chmod 777 task1.sh
[vvustinova@vvustinova ~]$ ./task1.sh
./task1.sh
[vvustinova@vvustinova ~]$
```

Рис. 3.2: Вводим команду ./task1.sh

При этом файл должен архивироваться одним из архиваторов на выбор(рис. 3.3).

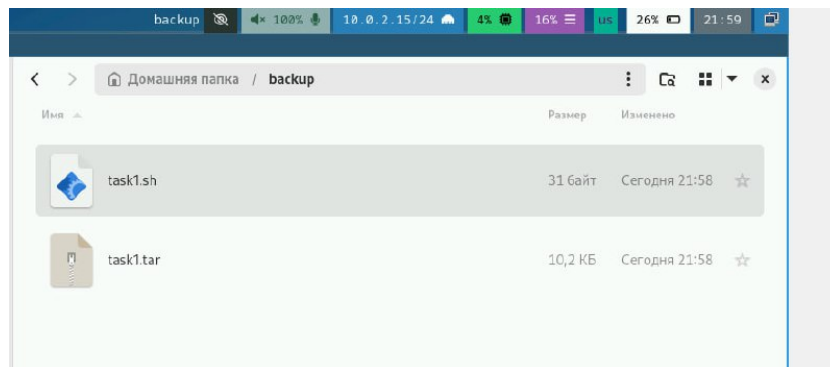


Рис. 3.3: Перешли с созданную папку backup и разархивировали файл

Создаем новый файл для второго задания и открываем редактор(рис. 3.4).

```
[vvustinova@vvustinova ~]$ touch task2.sh
[vvustinova@vvustinova ~]$ chmod 777 task2.sh
[vvustinova@vvustinova ~]$ nano task2.sh
[vvustinova@vvustinova ~]$
```

Рис. 3.4: Редактор nano

Написать пример командного файла, обрабатывающего любое произвольное число аргументов командной строки(рис. 3.5).

```
GNU nano 8.1 task2.sh
for i in "$@"
do echo ${i}
done
```

Рис. 3.5: Пишем код

Скрипт может последовательно распечатывать значения всех переданных аргументов(рис. 3.6).



```

[vvustinova@vvustinova ~]$ ./task2.sh sssl html apple pear king
ssssl
html
apple
pear
king
[vvustinova@vvustinova ~]$

```

Рис. 3.6: У нас выводит последовательно, как и требуется

Написать командный файл — аналог команды `ls` (без использования самой этой команды и команды `dir`)(рис. 3.7).

```

foot
GNU nano 8.1 task3.sh
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. 3.7: Пишем код

Требуется, чтобы он выдавал информацию о нужном каталоге и выводил информацию о возможностях доступа к файлам этого каталога(рис. 3.8).

```

[vvustinova@vvustinova ~]$ nano task3.sh
[vvustinova@vvustinova ~]$ nano task3.sh
[vvustinova@vvustinova ~]$ ./task3.sh ~
/home/vvustinova/drwx-----
/home/vvustinova/1 -rw-r--r--
/home/vvustinova/#2# -rw-r--r--
/home/vvustinova/#3# -rw-r--r--
/home/vvustinova/#4# -rw-r--r--
/home/vvustinova/abc1 -rw-rw-r--
/home/vvustinova/australia drwxr--r--
/home/vvustinova/backup drwxr-xr-x
/home/vvustinova/conf.txt -rw-r--r--
/home/vvustinova/dconf.txt drwxr-xr-x
/home/vvustinova/Desktop drwxr-xr-x
/home/vvustinova/Documents drwxr-xr-x

```

Рис. 3.8: Выводит необходимое

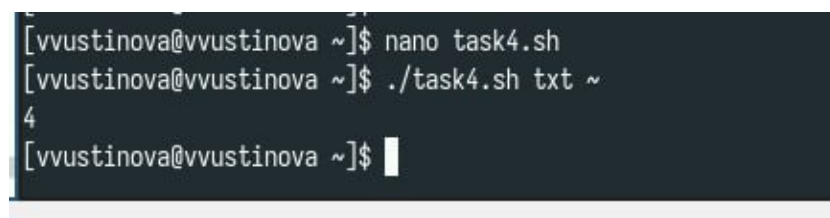
Написать командный файл, который получает в качестве аргумента командной строки формат файла (.txt, .doc, .jpg, .pdf и т.д.) и вычисляет количество таких файлов в указанной директории. (рис. 3.9).

A screenshot of a terminal window showing the nano text editor editing a file named task4.sh. The editor's title bar shows 'GNU nano 8.1' and 'task4.sh' with a status 'Изменён'. The script content is as follows:

```
let COUNT=0
for i in $(ls *.txt); do
  do let COUNT++
done
echo $COUNT
```

Рис. 3.9: Пишем код

Проверяем выполнение(рис. ??).

A screenshot of a terminal window showing the execution of the script. The user runs 'nano task4.sh' and then './task4.sh txt ~'. The output of the script is '4'.

```
[vvustinova@vvustinova ~]$ nano task4.sh
[vvustinova@vvustinova ~]$ ./task4.sh txt ~
4
[vvustinova@vvustinova ~]$
```

Рис. 3.10: Выводит необходимое, я проверила, ровно 4 файла txt находятся у меня в домашней директории

## 4 Выводы

Мы успешно изучили основы программирования в оболочке ОС UNIX/Linux. Научились писать небольшие командные файлы.

### 4.1 Ответы на контрольные вопросы

1. Командная оболочка: Интерфейс для взаимодействия с ОС через команды. Примеры: `bash`, `zsh`, `fish`. Отличаются синтаксисом, возможностями.
2. POSIX: Семейство стандартов, определяющих совместимость ОС Unix-типа.
3. Переменные: `var=value`, Массивы: `array=(item1 item2)`.
4. `let`: Для арифметических операций. `read`: Для чтения ввода от пользователя.
5. Арифметика: `+`, `-`, `,`, `/`, `%`, `\*`.
6. `(( ))`: Арифметические выражения и вычисления.
7. Стандартные переменные: `PATH`, `HOME`, `USER`, `PWD`, `SHELL`.
8. Метасимволы: Символы, имеющие специальное значение для оболочки (`*`, `?`, `[]`).
9. Экранирование: Обратный слэш `\`, кавычки (одинарные/двойные).
10. Командные файлы: Создать текстовый файл с командами, дать права на выполнение (`chmod +x`), запустить `./filename`.
11. Функции: `function_name() { commands; }`.
12. Тип файла: Команды `test -d` (каталог), `test -f` (обычный файл).
13. Назначение: `set`: Установить опции оболочки/переменные, `typeset`: Объявить атрибуты переменных, `unset`: Удалить переменную.

14. Параметры: Передаются при запуске скрипта: `./script arg1 arg2`. Доступны как `$1`, `$2` и т.д.
15. Специальные переменные: `$?`: Код возврата последней команды, `$$`: PID текущего процесса, `$@`: Все аргументы.