

# **Лабораторная работа №12**

**Отчет**

Зубов Иван Александрович

# 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>Контрольные вопросы</b>	<b>11</b>

## Список иллюстраций

3.1	Создаем файл . . . . .	7
3.2	Пишем скрипт . . . . .	7
3.3	Создаем файл . . . . .	8
3.4	Пишем скрипт . . . . .	8
3.5	Создаем файл . . . . .	8
3.6	Пишем скрипт . . . . .	9
3.7	Создаем файл . . . . .	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

Создаем файл `task1.sh` и пишем 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`.

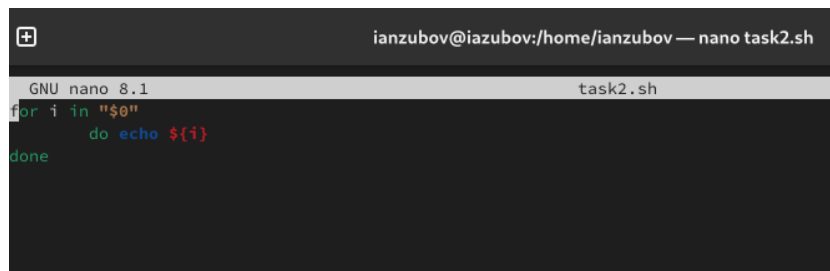
```
bash: /usr/libexec/mc/mc.sh: Нет такого файла или каталога
ianzubov@ianzubov:~$ touch task1.sh
ianzubov@ianzubov:~$ chmod 777 task1.sh
ianzubov@ianzubov:~$ nano task1.sh
ianzubov@ianzubov:~$
```

Рис. 3.1: Создаем файл

```
ianzubov@ianzubov:/home/ianzubov — nano task1.sh
GNU nano 8.1 task1.sh
tar -cvf ~/backup/task1.tar $0
```

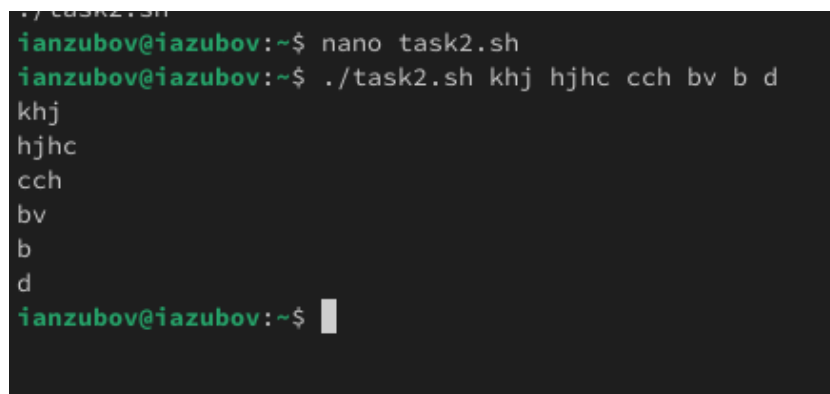
Рис. 3.2: Пишем скрипт

Создаем файл `task2.sh`, даем ему права доступа и пишем пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять.



```
ianzubov@ianzubov:/home/ianzubov — nano task2.sh
GNU nano 8.1 task2.sh
for i in "$@"
do echo ${i}
done
```

Рис. 3.3: Создаем файл



```
ianzubov@ianzubov:~$ nano task2.sh
ianzubov@ianzubov:~$ ./task2.sh khj hjhc cch bv b d
khj
hjhc
cch
bv
b
d
ianzubov@ianzubov:~$
```

Рис. 3.4: Пишем скрипт

Создаем файл task3.sh, даем ему права доступа и пишем 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.



```
ianzubov@ianzubov:/home/ianzubov — nano task3.sh
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. 3.5: Создаем файл



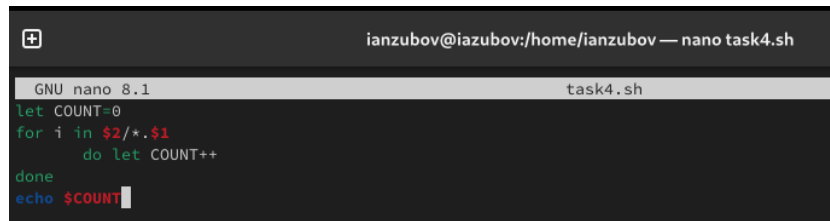
```

anzubov@ianzubov:~$ ./task3.sh ~
home/ianzubov/ drwx-----
home/ianzubov/#1# -rw-r--r--
home/ianzubov/#2# -rw-r--r--
home/ianzubov/#3# -rw-r--r--
home/ianzubov/#4# -rw-r--r--
home/ianzubov/australia drwxr--r--
home/ianzubov/backup drwxr-xr-x
home/ianzubov/conf.txt -rw-r--r--
home/ianzubov/Documents drwxr-xr-x
home/ianzubov/Downloads drwxr-xr-x
home/ianzubov/file.txt -rw-r--r--
home/ianzubov/#lab07.sh# -rw-r--r--
home/ianzubov/#lab7.sh# -rw-r--r--
home/ianzubov/LICENSE -rw-r--r--
home/ianzubov/monthly drwxr-xr-x
home/ianzubov/monthly1 drwx--x--x
home/ianzubov/Pictures drwxr-xr-x
home/ianzubov/play drwx--x--x
home/ianzubov/reports drwxr-xr-x
home/ianzubov/ski.plases drwxr-xr-x
home/ianzubov/task1.sh -rwxrwxrwx
home/ianzubov/task2.sh -rwxrwxrwx
home/ianzubov/task3.sh -rwxrwxrwx
home/ianzubov/work drwxr-xr-x
home/ianzubov/Видео drwxr-xr-x
home/ianzubov/Документы drwxr-xr-x
home/ianzubov/Загрузки drwxr-xr-x
home/ianzubov/Изображения drwxr-xr-x
home/ianzubov/Музыка drwxr-xr-x
home/ianzubov/Общедоступные drwxr-xr-x
home/ianzubov/Рабочий стол drwxr-xr-x
home/ianzubov/Шаблоны drwxr-xr-x
anzubov@ianzubov:~$ ./task3.sh

```

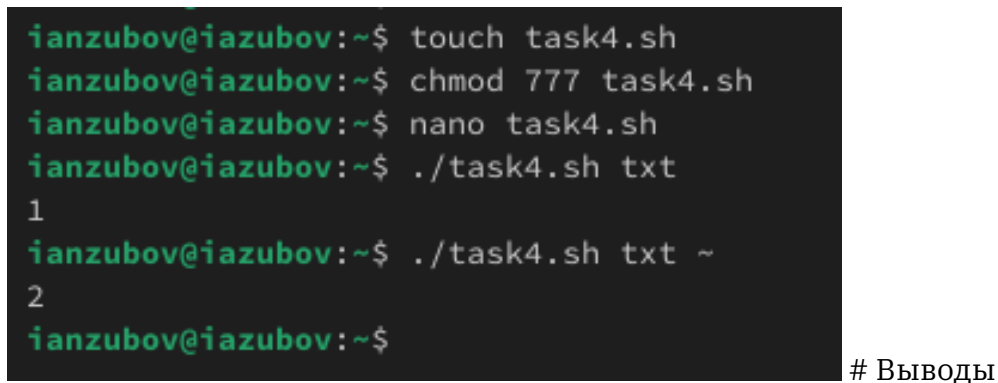
Рис. 3.6: Пишем скрипт

Создаем файл task4.sh, даем ему права доступа и пишем 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.



```
ianzubov@iazubov:/home/ianzubov — nano task4.sh
GNU nano 8.1 task4.sh
let COUNT=0
for i in $2/*. $1
do let COUNT++
done
echo $COUNT
```

Рис. 3.7: Создаем файл



```
ianzubov@iazubov:~$ touch task4.sh
ianzubov@iazubov:~$ chmod 777 task4.sh
ianzubov@iazubov:~$ nano task4.sh
ianzubov@iazubov:~$ ./task4.sh txt
1
ianzubov@iazubov:~$ ./task4.sh txt ~
2
ianzubov@iazubov:~$
```

# Выводы

Я научился писать небольшие командные файлы

## 4 Контрольные вопросы

1. Командная оболочка (shell) - это интерфейс между пользователем и ядром ОС, позволяющий запускать программы и управлять системой с помощью текстовых команд.
  - Примеры: bash, zsh, sh, csh, fish.
  - Отличия: Синтаксис, набор встроенных команд, возможности настройки, интерактивные функции.
2. POSIX - набор стандартов IEEE, описывающих интерфейсы операционной системы, обеспечивающие переносимость программ между различными Unix-подобными системами.
3. Переменные: `variable=value` (без пробелов). Массивы: `array=(item1 item2 item3)`.
4. `let`: Выполняет арифметические операции. `read`: Считывает ввод пользователя.
5. Арифметические операции: `+`, `-`, `,`, `/`, `%`, (возведение в степень).
6. `(( ))`: Окружает арифметическое выражение для вычисления.
7. Стандартные переменные: `HOME`, `PATH`, `USER`, `PWD`, `UID`, `SHELL`, `PS1` (prompt).
8. Метасимволы - символы, имеющие специальное значение для командной оболочки (например, `*`, `?`, `[]`, `|`, `>`, `<`).

9. Экранирование: Обратный слеш , одинарные кавычки ' ', двойные кавычки " " (частичное экранирование).
10. Создание: Создать текстовый файл с командами. Запуск: `bash script.sh` или `./script.sh` (если файл исполняемый).
11. Определение функции: `function function_name() { # commands }`
12. Проверка типа файла: `if [ -d "filename" ]; then echo "Directory"; fi` или `if [ -f "filename" ]; then echo "Regular file"; fi`
13. Назначение команд:
  - `set`: Устанавливает или снимает различные параметры командной оболочки, также отображает все переменные.
  - `typeset`: (или `declare`) Объявляет переменные и задает их атрибуты (например, `integer`, `readonly`).
  - `unset`: Удаляет переменную.
14. Передача параметров: Аргументы командной строки передаются в командный файл как `$1`, `$2`, `$3...` `$9`, `${10}`, `${11}` и т.д. Все аргументы содержатся в переменной `$@`.
15. Специальные переменные:
  - `$0`: Имя скрипта.
  - `$1`, `$2`, ... `$9`, `${10}` и т.д. : Аргументы командной строки.
  - `$#`: Количество аргументов командной строки.
  - `$@`: Все аргументы командной строки (как отдельные слова).
  - `$*`: Все аргументы командной строки (как одна строка).
  - `$?`: Код возврата последней выполненной команды.
  - `$$`: PID текущего процесса.
  - `$_`: PID последнего запущенного фонового процесса.