

# **Отчет о выполнении лабораторной работы**

**Лабораторная работа №12**

Филиппьева Ксения Дмитриевна

# Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	13
5	Ответы на вопросы	14

## Список иллюстраций

3.1	создание папки . . . . .	7
3.2	создание файла . . . . .	8
3.3	код задания . . . . .	8
3.4	выдача прав . . . . .	8
3.5	создание файла . . . . .	9
3.6	код для задания . . . . .	9
3.7	работоспособность кода №2 . . . . .	9
3.8	создание файла . . . . .	10
3.9	код для задания . . . . .	10
3.10	работоспособность кода №3 . . . . .	11
3.11	создание файла . . . . .	11
3.12	код для задания . . . . .	12
3.13	работоспособность кода №4 . . . . .	12

## **Список таблиц**

# 1 Цель работы

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

## 2 Задание

Приобрести и отработать уже имеющиеся навыки программирования в оболочке ОС Linux.

### 3 Выполнение лабораторной работы

Создадим папку в которую будут сохраняться бэк-апы для первого задания.  
(рис. 3.1).

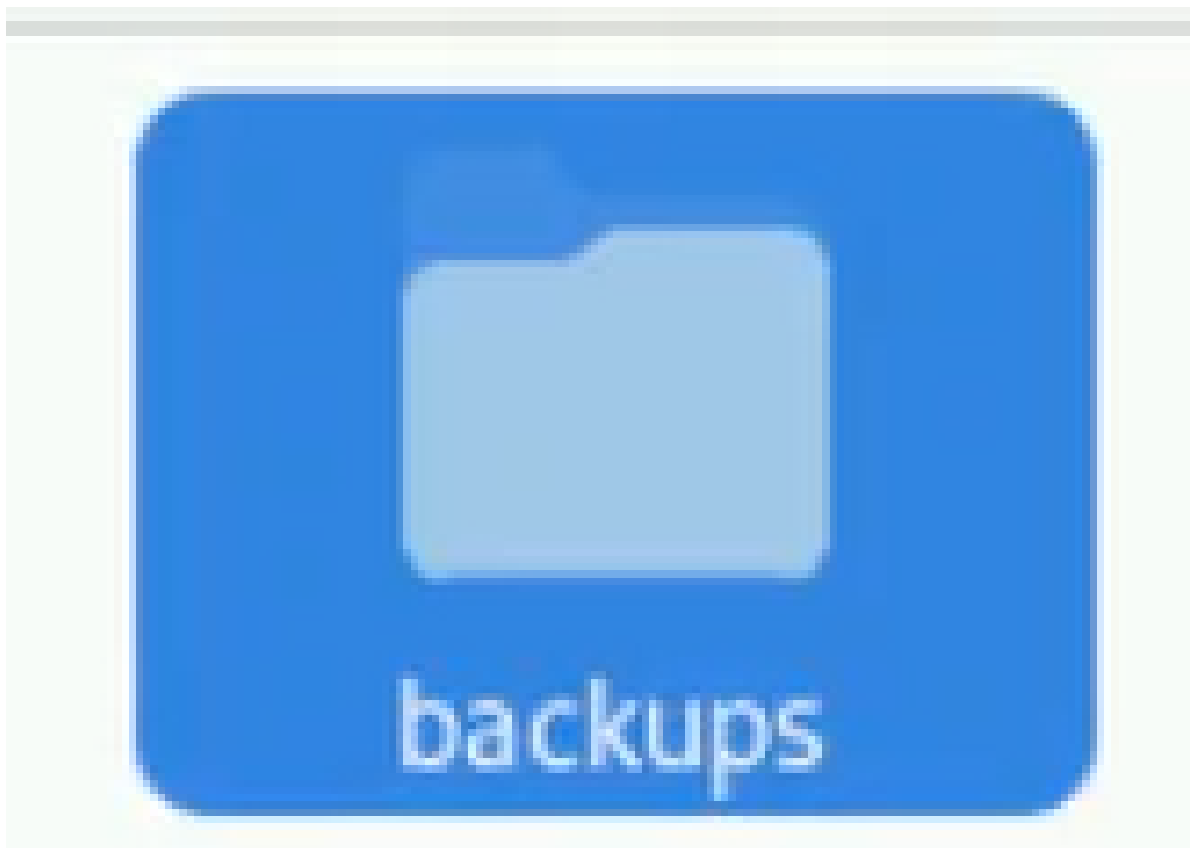


Рис. 3.1: создание папки

Создадим файл для первого задания и откроем его (рис. 3.2).

```
[kdfilipjeva@kdfilipjeva ~]$ touch t1.sh
[kdfilipjeva@kdfilipjeva ~]$ nano t1.sh
```

Рис. 3.2: создание файла

Впишем в него код, который позволит выполнять нам поставленные задачи (рис. 3.3).

```
GNU nano 7.2      t1.sh      Изменён
tar -cvf ~/backups/t1.tar $0
```

Рис. 3.3: код задания

Выдадим все права на файл и выполним его (тут я забыла сделать скрин вывода, но архив создался в папке) (рис. 3.4).

```
[kdfilipjeva@kdfilipjeva ~]$ chmod 777 t1.sh
[kdfilipjeva@kdfilipjeva ~]$ ./t1.sh
./t1.sh
[kdfilipjeva@kdfilipjeva ~]$
```

Рис. 3.4: выдача прав

Создадим файл для второго задания, выдадим все права и откроем его для редактирования (рис. 3.5).



```
[kdfilipjeva@kdfilipjeva ~]$ touch t2.sh
[kdfilipjeva@kdfilipjeva ~]$ chmod 777 t2.sh
[kdfilipjeva@kdfilipjeva ~]$ nano t2.sh
```

Рис. 3.5: создание файла

Впишем в него код, позволяющий выполнить поставленное задание (рис. 3.6).

```
GNU nano 7.2      t2.sh      Изменён
for i in "$@"
do echo ${i}
done
```

Рис. 3.6: код для задания

Работоспособность кода (рис. 3.7).

```
[kdfilipjeva@kdfilipjeva ~]$ ./t2.sh asd df
jgh fdoigure 234 dflkgj
asd
dfjgh
fdoigure
234
dflkgj
[kdfilipjeva@kdfilipjeva ~]$
```

Рис. 3.7: работоспособность кода №2

Создадим файл для третьего задания, выдадим все права и откроем его для редактирования (рис. 3.8).

```
[kdfilipjeva@kdfilipjeva ~]$ touch t3.sh
[kdfilipjeva@kdfilipjeva ~]$ chmod 777 t3.sh
[kdfilipjeva@kdfilipjeva ~]$
```

Рис. 3.8: создание файла

Впишем в него код, позволяющий выполнить поставленное задание (рис. 3.9).

```
GNU nano 7.2      t3.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. 3.9: код для задания

Работоспособность кода (рис. 3.10).

```
[kdfilipjeva@kdfilipjeva ~]$ ./t3.sh ~  
/home/kdfilipjeva/ drwx-----  
/home/kdfilipjeva/abc1 drwx-----  
/home/kdfilipjeva/australia drwx-----  
/home/kdfilipjeva/backups drwx-----  
/home/kdfilipjeva/bin drwx-----  
/home/kdfilipjeva/cconf.txt drwx-----  
/home/kdfilipjeva/conf.txt drwx-----  
/home/kdfilipjeva/Desktop drwx-----  
/home/kdfilipjeva/Downloads drwx-----  
/home/kdfilipjeva/feathers drwx-----  
/home/kdfilipjeva/file.txt drwx-----  
/home/kdfilipjeva/git-extended drwx-----  
/home/kdfilipjeva/lab071.sh drwx-----  
/home/kdfilipjeva/lab072.sh drwx-----
```

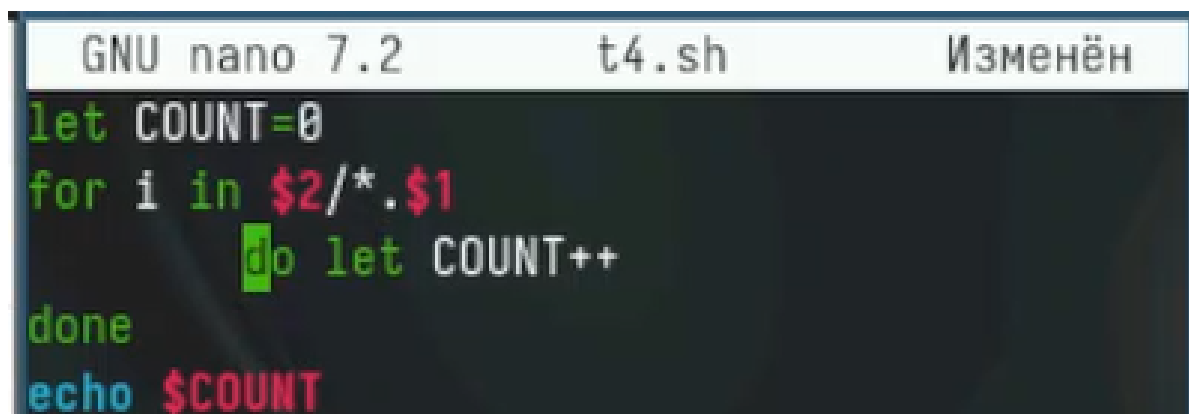
Рис. 3.10: работоспособность кода №3

Создадим файл для четвертого задания, выдадим все права и откроем его для редактирования (рис. 3.11).

```
[kdfilipjeva@kdfilipjeva ~]$ touch t4.sh  
[kdfilipjeva@kdfilipjeva ~]$ chmod 777 t4.s  
h  
[kdfilipjeva@kdfilipjeva ~]$ nano t4.sh
```

Рис. 3.11: создание файла

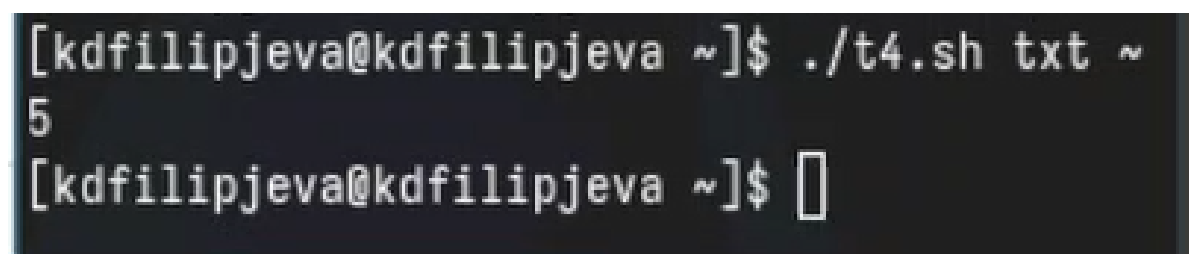
Впишем в него код, позволяющий выполнить поставленное задание (рис. 3.12).



```
GNU nano 7.2          t4.sh          Изменён
let COUNT=0
for i in $2/*. $1
do let COUNT++
done
echo $COUNT
```

Рис. 3.12: код для задания

Работоспособность кода (рис. 3.13).



```
[kdfilipjeva@kdfilipjeva ~]$ ./t4.sh txt ~
5
[kdfilipjeva@kdfilipjeva ~]$
```

Рис. 3.13: работоспособность кода №4

## 4 Выводы

Мы получили новые и отработали уже имеющиеся навыки программирования в оболочке ОС Linux.

## 5 Ответы на вопросы

1. Командная оболочка (shell) - это программа, которая обеспечивает интерфейс для взаимодействия пользователя с операционной системой. Примеры: Bash (Bourne Again Shell), Zsh (Z shell), Fish (Friendly Interactive Shell). Они отличаются синтаксисом, встроенными функциями, расширенными возможностями автодополнения и настройками.
2. POSIX (Portable Operating System Interface) - это набор стандартов, определяющих интерфейсы между операционной системой и прикладными программами для обеспечения переносимости.
3. Переменные в Bash определяются как имя=значение. Массивы определяются как имя=(значение1 значение2 ... ) или имя[индекс]=значение.
4. let позволяет выполнять арифметические операции, а read считывает ввод пользователя и сохраняет его в переменной.
5. Bash поддерживает основные арифметические операции: сложение (+), вычитание (-), умножение (\*), деление (/), остаток от деления (%), возведение в степень (\*\*).
6. Операция (( )) используется для выполнения арифметических операций и сравнений.
7. Стандартные переменные: HOME (домашний каталог), PATH (список каталогов для поиска команд), USER (имя текущего пользователя) и др.
8. Метасимволы - это специальные символы, имеющие особое значение для командной оболочки (например, \*, ?, [, ]).
9. Метасимволы экранируются с помощью обратного слэша \ или заключаются в кавычки.

10. Командные файлы создаются в текстовом редакторе и сохраняются с расширением `.sh`. Запуск: `bash имя_файла.sh` или `./имя_файла.sh` (при наличии прав на исполнение).
11. Функции в Bash определяются как `имя_функции() { команды; }`.
12. Для проверки типа файла используется команда `test` или `[ : [ -d файл ]` (каталог) или `[ -f файл ]` (обычный файл).
13. `set` устанавливает опции командной оболочки, `typeset` объявляет переменные и их атрибуты, `unset` удаляет переменные или функции.
14. Параметры передаются в командные файлы через аргументы командной строки, доступные как `$1`, `$2` и т.д.
15. Специальные переменные Bash: `$0` (имя скрипта), `$#` (количество аргументов), `$@` (все аргументы в виде отдельных слов), `$*` (все аргументы в виде одной строки), `$$` (PID текущего процесса) и др.