

# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

## Факультет физико-математических и естественных наук

### Кафедра прикладной информатики и теории вероятностей

## ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 11

### *дисциплина: Операционные системы*

Студент: Ким Реачна

Группа: НПИбд-02-20

Москва

2021г.

### Цель работы:

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

### Теоретическое введение:

#### Командные процессоры (оболочки):

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 Сподобный синтаксис команд с возможностью сохранения истории выполнения команд;
- *оболочка Корна (или ksh)* — напоминает оболочку С, но операторы управления программой совместимы с операторами оболочки Борна;
- *BASH* — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек С и Корна (разработка компании Free Software Foundation).

*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 Корна.

### **Переменные в языке программирования bash:**

Командный процессор bash обеспечивает возможность использования переменных типа строка символов. Имена переменных могут быть выбраны пользователем.

Пользователь имеет возможность присвоить переменной значение некоторой строки символов. Например, команда

```
mark=/usr/andy/bin
```

присваивает значение строки символов `/usr/andy/bin` переменной `mark` типа *строка символов*.

Значение, присвоенное некоторой переменной, может быть впоследствии использовано. Для этого в соответствующем месте командной строки должно быть употреблено имя этой переменной, которому предшествует метасимвол `$`. Например, команда

```
mv afile ${mark}
```

Команда `echo` в Linux используется для отображения строки текста/строки, которые передаются в качестве аргумента. Это встроенная команда, которая в основном используется в сценариях оболочки и пакетных файлах для вывода текста состояния на экран или в файл.

```
echo [string]
```

Команда `read` принимает ввод с клавиатуры и присваивает его переменной.

```
read [options] [name...]
```

## Выполнение работы:

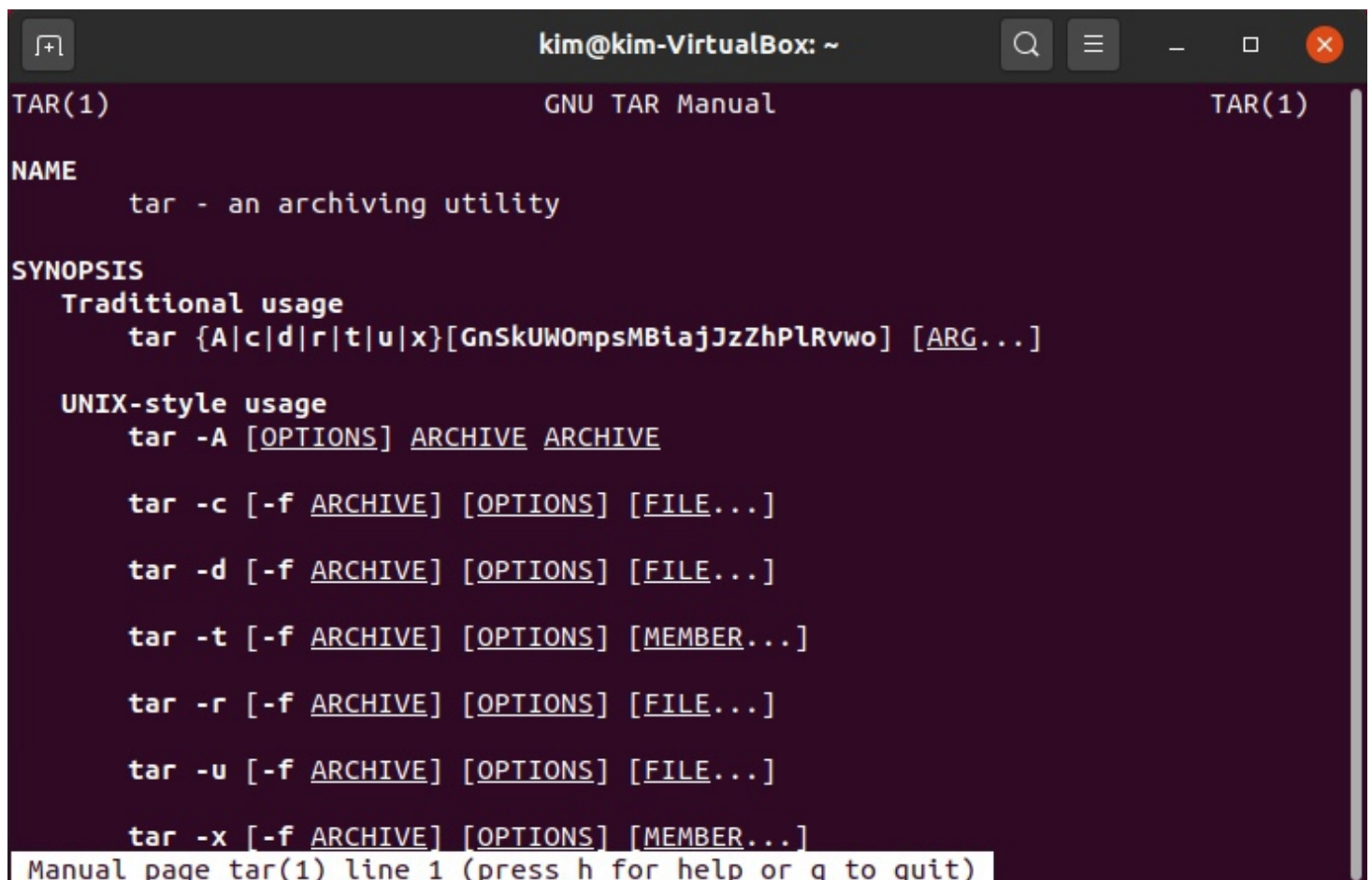
**Задание1:** Написать скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию `backup` в вашем домашнем каталоге. При этом файл должен архивироваться одним из архиваторов на выбор `zip`, `bzip2` или `tar`. Способ использования команд архивации необходимо узнать, изучив справку.

- Сначала мы изучаем команду `tar` с помощью командной `man tar` (Рисунок 1-2)

Рисунок 1: изучаем команду `tar`

```
kim@kim-VirtualBox:~$ man tar
kim@kim-VirtualBox:~$
```

Рисунок 2: изучаем команду `tar`

A screenshot of a terminal window titled "kim@kim-VirtualBox: ~". The terminal displays the GNU TAR Manual. The window has a dark theme with a purple background. The manual text is white and yellow. The text shows the NAME, SYNOPSIS, and various usage examples for the tar command. The terminal window has standard Linux window controls (minimize, maximize, close) in the top right corner. The bottom of the terminal shows the prompt "Manual page tar(1) line 1 (press h for help or q to quit)".

```
TAR(1) GNU TAR Manual TAR(1)

NAME
    tar - an archiving utility

SYNOPSIS
    Traditional usage
        tar {A|c|d|r|t|u|x}[GnSkUWompsMBiajJzZhPlRvwo] [ARG...]

    UNIX-style usage
        tar -A [OPTIONS] ARCHIVE ARCHIVE

        tar -c [-f ARCHIVE] [OPTIONS] [FILE...]

        tar -d [-f ARCHIVE] [OPTIONS] [FILE...]

        tar -t [-f ARCHIVE] [OPTIONS] [MEMBER...]

        tar -r [-f ARCHIVE] [OPTIONS] [FILE...]

        tar -u [-f ARCHIVE] [OPTIONS] [FILE...]

        tar -x [-f ARCHIVE] [OPTIONS] [MEMBER...]

Manual page tar(1) line 1 (press h for help or q to quit)
```

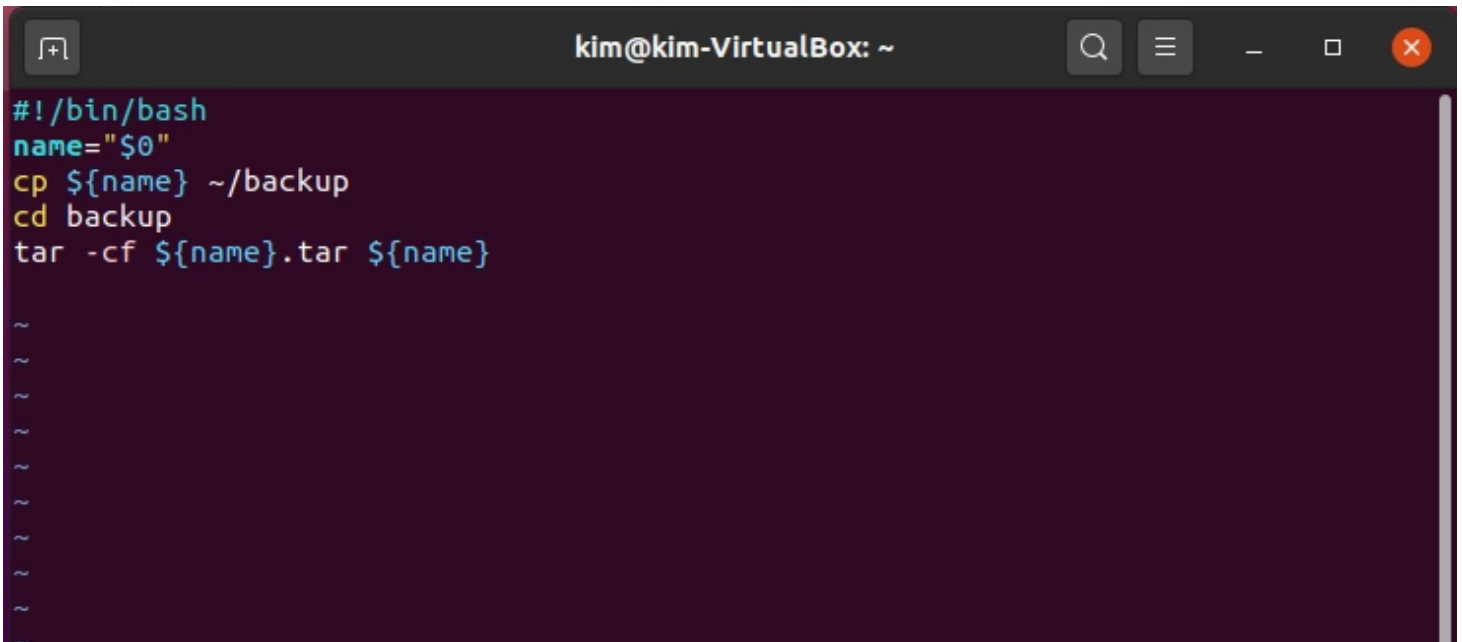
- В этом процессе мы сначала переходим в домашний каталог с помощью команды `cd` , затем создаем `backup` каталога в домашнем каталоге с помощью команды `mkdir` , а затем создаем сценарий, который мы собираемся записать с помощью редактора `vi` который называется `scr1.sh` (Рисунок 3)

Рисунок 3: Создание файла и каталога

```
kim@kim-VirtualBox:~$ man tar
kim@kim-VirtualBox:~$ cd
kim@kim-VirtualBox:~$ mkdir backup
kim@kim-VirtualBox:~$ vi scr1.sh
```

- В редакторе `scr1.sh` мы пишем командный файл , который мы начнем с `#!/bin/bash` в первой строке, затем используем команду `cp` для перемещения или копирования файла в резервную копию каталога, затем меняем каталог на резервную копию, а затем с помощью команды `tar` объединяем несколько файлов и или каталогов вместе в один файл.(Рисунок 4)

Рисунок 4: Создать командный файл

A screenshot of a terminal window titled "kim@kim-VirtualBox: ~". The window shows the contents of a script file named scr1.sh. The script starts with a shebang line "#!/bin/bash", followed by "name=\"\$0\"", "cp \${name} ~/backup", "cd backup", and "tar -cf \${name}.tar \${name}". There are several tilde characters (~) at the bottom of the terminal, indicating the prompt character. The terminal window has standard Linux window controls (search, menu, zoom, close) in the top right corner.

```
#!/bin/bash
name="$0"
cp ${name} ~/backup
cd backup
tar -cf ${name}.tar ${name}

~
~
~
~
~
~
~
~
~
~
```

- Затем переключитесь в командный режим с помощью клавиши `Esc` , затем `:` затем `wq` , чтобы записать изменения в файл перед выходом из редактора (Рисунок 5), затем он вернется обратно в наш терминал, а затем мы используем команду `bash scr1.sh` для выполнения команды из файла (Рисунок 6)

Рисунок 5: Вставить текст

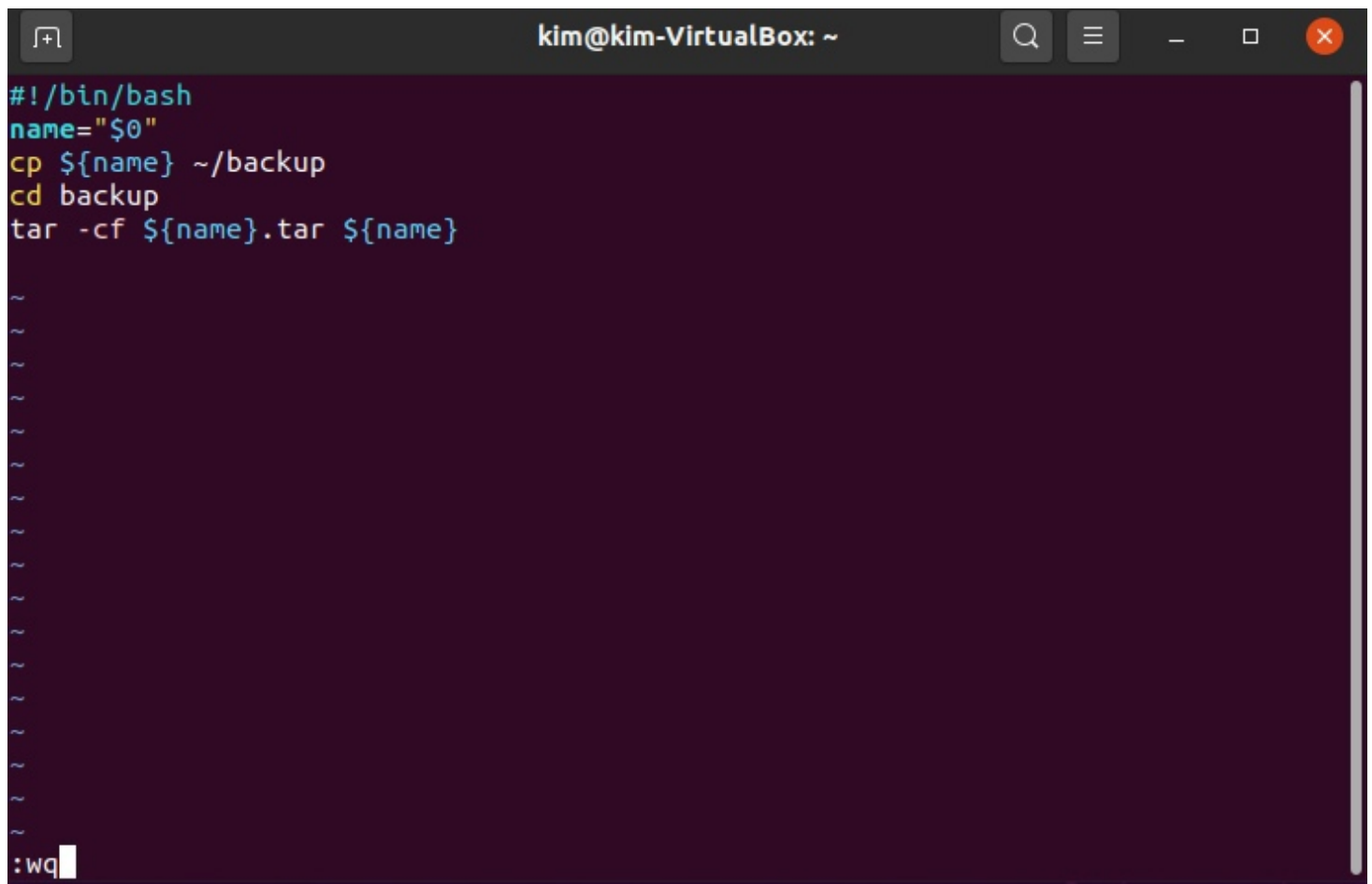


Рисунок 6: Запустите командный файл

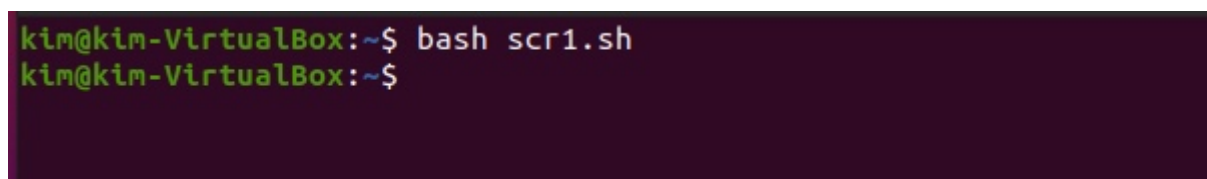


Рисунок 7: проверка файлов

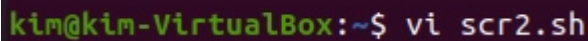


Как мы видим, он успешно создает.

**Задание 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.

- Для этого сначала мы создаем новый командный файл с помощью редактора `vi` снова `vi scr2.sh` и сценарий-это вызов `scr2.sh` (Рисунок 8)

Рисунок 8: Создать новый командный файл



```
kim@kim-VirtualBox:~$ vi scr2.sh
```

- Для этого мы используем массив, называемый числами. Сначала мы объявляем `-a numbers` с помощью команды `declare`, а затем для отображения строковых элементов ввода, которые передали аргумент с помощью команды `echo "Enter numbers "`. Затем мы используем команду `read` для чтения элементов массива с клавиатуры, затем снова используем команду `echo` для отображения строки ваших элементов, а затем элементов массива с помощью `echo ${numbers[@]}` затем сохраните и закройте файл (Рисунок 9).

Рисунок 9: Вставить командный файл `scr2.sh`



```
kim@kim-VirtualBox: ~
#!/bin/bash
declare -a numbers
echo "Enter elements:"
read -a numbers
echo "Your elements:"
echo ${numbers[@]}

~
~
~
~
~
:wq
```

- Мы проверяем нашу работу с помощью команды `bash scr2.sh` и нажмите клавишу `Enter` после чего мы сможем ввести любые элементы в наш массив (Рисунок 10)

Рисунок 10: Результаты командного файла `scr2.sh`

```
kim@kim-VirtualBox:~$ bash scr2.sh
Enter elements:
1 2 3 4 4 6 8 2 87 13 23 1333 5643 23445 123456
Your elements:
1 2 3 4 4 6 8 2 87 13 23 1333 5643 23445 123456
kim@kim-VirtualBox:~$
```

**Задание 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.

Рисунок 11: Создать новый файл `scr3.sh`

```
kim@kim-VirtualBox:~$ vi scr3.sh
kim@kim-VirtualBox:~$
```

- Пишем текст командного файла. Сначала выведем сообщение о вводе имени каталога, который мы хотим рассмотреть, `echo`. Команда `read` позволит нам считать введенную с клавиатуры директорию в переменную `name`. Выводим имя директории и переходим в заданный каталог: `cd ${name}`. Выведем строку-сообщение о выводе файлов каталога и прав доступа к ним командой вывода `echo`. Выведем содержимое текущего каталога командой `stat`: `stat -c '%A %n' *`. Где `-c` является ключом, который выведет наши файлы построчно, `%A` - вывод прав доступа в формате, читаемом для человека, а не машины, `%n` - названия файлов, `*` - указывает на текущий каталог (Рисунок 12)

Рисунок 12: Вставить командный файл

```
kim@kim-VirtualBox: ~
#!/bin/bash
echo "Enter name of Directory:"
read name
echo "Directory name: $name"
cd ${name}
echo "File information and access rights:"
stat -c '%A %n' *
```

-- INSERT -- 9,1 All

- Здесь мы тестируем нашу работу , сначала используя `bash src3.sh` затем используйте клавишу `Enter` ; теперь вы можете увидеть строку ввода имени каталога, в котором вы хотите получить информацию, и их права доступа. (Рисунок 13)

Рисунок 13: Результаты выполнения задания 3

```
kim@kim-VirtualBox:~$ vi src3.sh
kim@kim-VirtualBox:~$ bash src3.sh
Enter name of Directory:
backup
Directory name: backup
File information and access rights:
-rw-rw-r-- src1.sh
-rw-rw-r-- src1.sh.tar
kim@kim-VirtualBox:~$
```

**Задание 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.

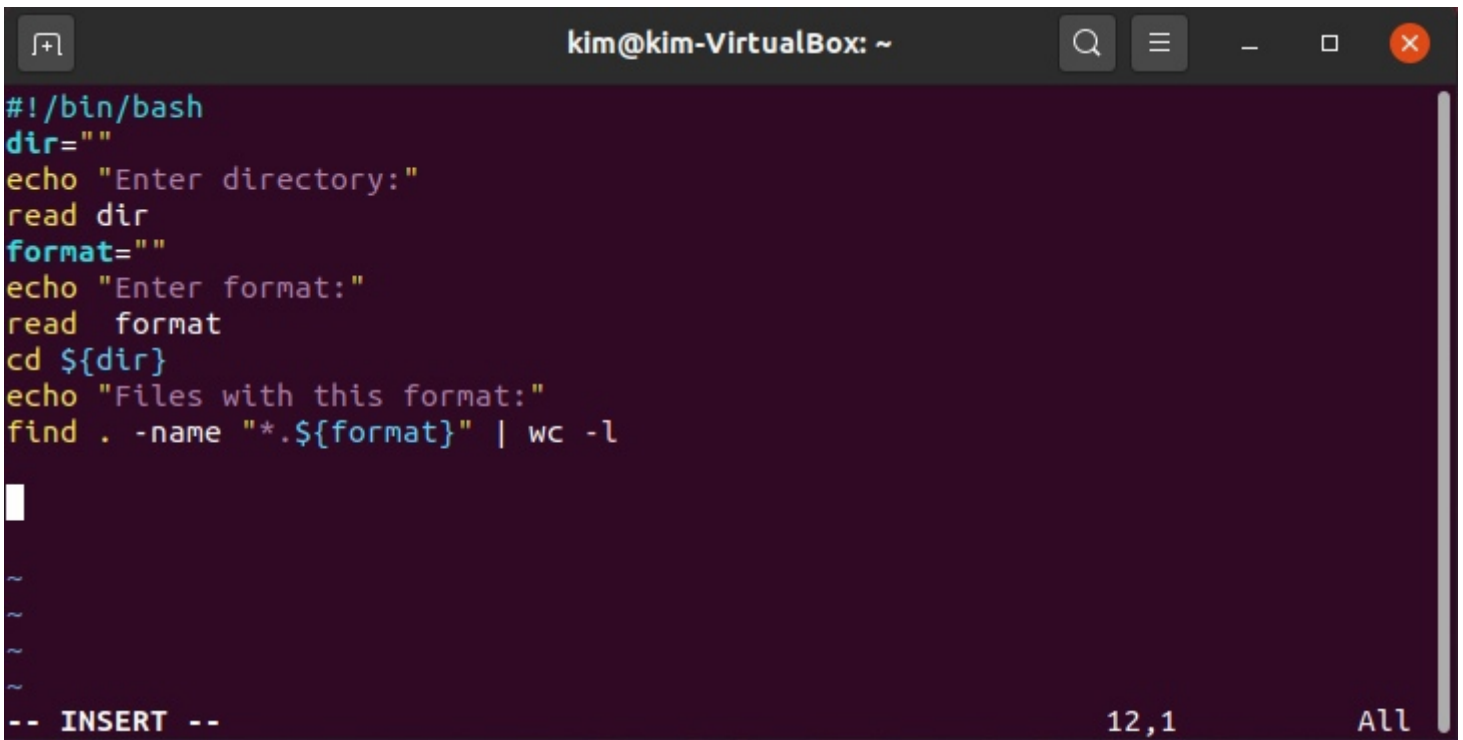
Рисунок 14: Создать новый файл `src4.sh`

```
kim@kim-VirtualBox:~$ vi src4.sh
```

- Давайте напишем сам командный файл. Мы введем обозначения двух переменных: `dir` , в котором мы запишем рассматриваемый каталог, и `format` , в котором мы запишем нужный формат файла. Они сопровождаются двумя эхо-выходами, которые информируют пользователя о том, что именно необходимо ввести в данный момент. `cd ${dir}` - перейдите в нужный каталог. Мы ищем (команда `find` ) в нем ( `" . "` - текущий каталог) файлы по имени ( `- name` ), в которых мы найдем введенный формат. Мы используем конвейер для чтения нереализованного вывода и используем команду `wc -l` для подсчета его строк, файлов, найденных в этом каталоге и соответствующих требованиям.

Рисунок 15: Вставить командный файл

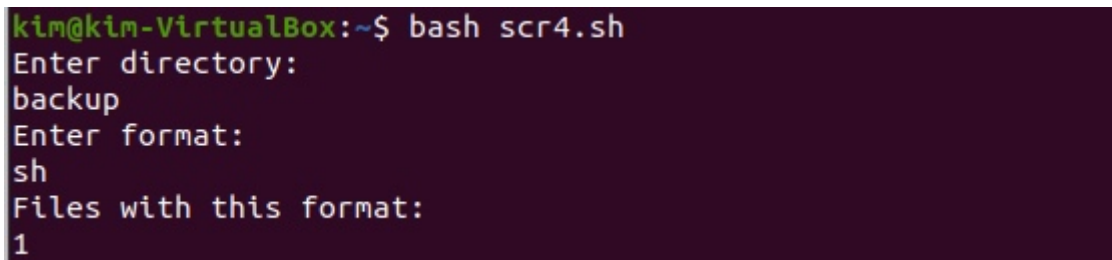


A terminal window titled 'kim@kim-VirtualBox: ~' with standard window controls. It shows a script being executed line by line. The script prompts for a directory and a file format, then uses 'find' to locate files and 'wc -l' to count them. The output shows 12 files found in the current directory with the specified format.

```
#!/bin/bash
dir=""
echo "Enter directory:"
read dir
format=""
echo "Enter format:"
read format
cd ${dir}
echo "Files with this format:"
find . -name "*.${format}" | wc -l

~
~
~
~
-- INSERT --                               12,1      All
```

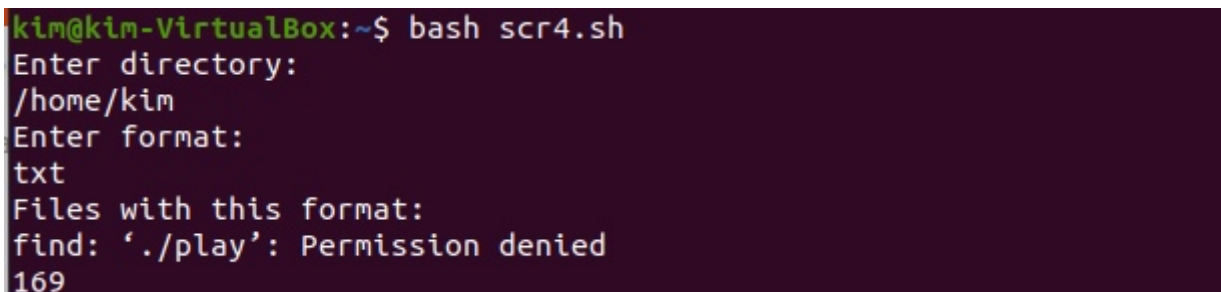
Рисунок 16: Тестирование и результаты задание 4

A terminal window showing the execution of 'scr4.sh'. The user enters 'backup' for the directory and 'sh' for the format. The script outputs that 1 file was found with this format.

```
kim@kim-VirtualBox:~$ bash scr4.sh
Enter directory:
backup
Enter format:
sh
Files with this format:
1
```

Как мы видим, здесь мы тестируем нашу работу, чтобы найти файлы или каталог с форматом `sh` .(Рисунок 16)

Рисунок 17: Тестирование и результаты задание 4

A terminal window showing the execution of 'scr4.sh'. The user enters '/home/kim' for the directory and 'txt' for the format. The script outputs that 169 files were found, but there is a 'Permission denied' error for the './play' directory.

```
kim@kim-VirtualBox:~$ bash scr4.sh
Enter directory:
/home/kim
Enter format:
txt
Files with this format:
find: './play': Permission denied
169
```

Далее мы ищем файлы и каталоги, которые имеют формат `txt`

Рисунок 18: Тестирование и результаты задание 4

```
kim@kim-VirtualBox:~$ bash scr4.sh
Enter directory:
/home/kim/Pictures
Enter format:
png
Files with this format:
32
```

И последнее, что мы тестируем, чтобы найти файлы и каталоги, которые имеют формат `png` .

## Вывод:

Я изучила основы программирования в оболочке ОС UNIX/Linux. Научилась писать небольшие командные файлы.

## Библиография:

[1]:[Лаборатория №11](#)

[2]:[read and echo](#)

[3]:[Команда bash](#)