

# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

ОТЧЕТ

ЛАБОРАТОРНОЙ РАБОТЕ № 11

## 1. дисциплина: Операционные системы

Студент: Мохаммад Амин Шоаибуллах

Группа: НПИбд-02-20

### Цель работы:

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

### Введение

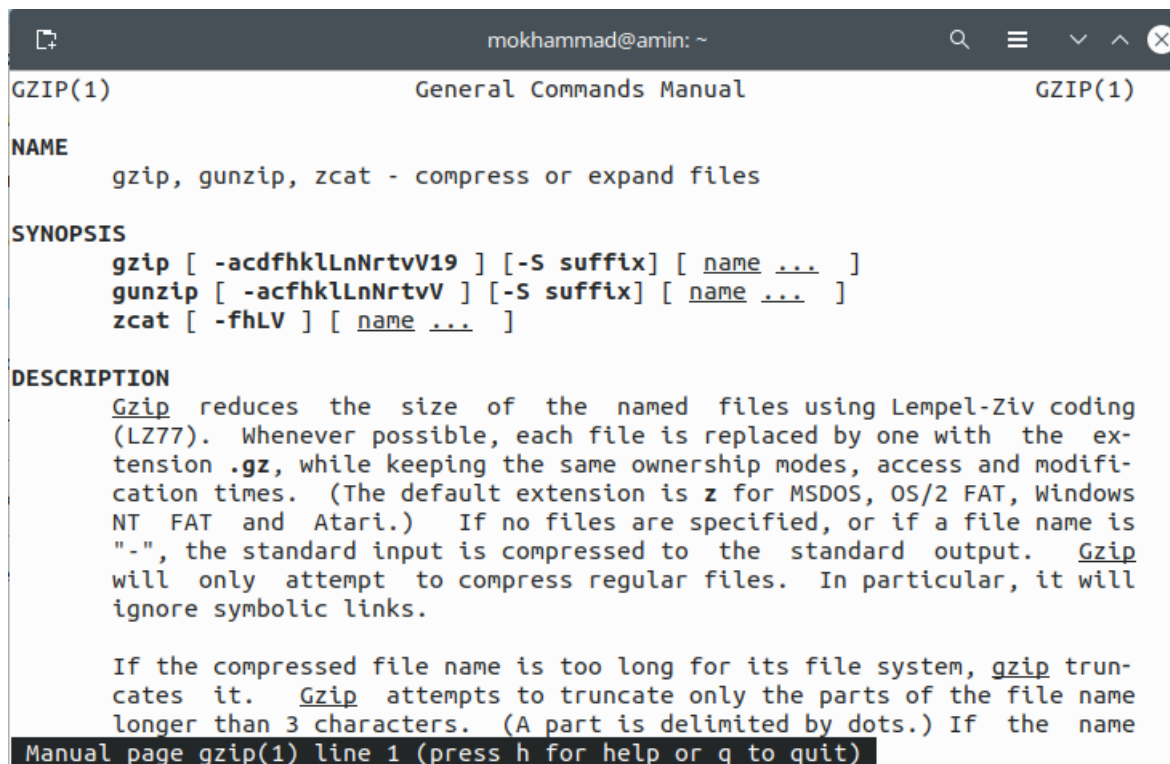
Командные процессоры или оболочки – это программы, позволяющие пользователю взаимодействовать с компьютером. Их можно рассматривать как настоящие интерпретируемые языки, которые воспринимают команды пользователя и обрабатывают их. Поэтому командные процессоры также называют интерпретаторами команд. На языках оболочек можно писать программы и выполнять их подобно любым другим программам. UNIX обладает большим количеством оболочек. Наиболее популярными являются следующие четыре оболочки: \* оболочка Борна (Bourne) – первоначальная командная оболочка UNIX: базовый, но полный набор функций; \* C-оболочка – добавка университета Беркли к коллекции оболочек: она надстраивается над оболочкой Борна, используя C-подобный синтаксис команд, и сохраняет историю выполненных команд; \* оболочка Корна – напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна; \* BASH – сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation).

### Последовательность выполнения работы

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.

### Ход работы:

1. Изучил опции команды, и саму команду *gzip*, с помощью команды *man*.



```
GZIP(1)                                General Commands Manual                                GZIP(1)

NAME
    gzip, gunzip, zcat - compress or expand files

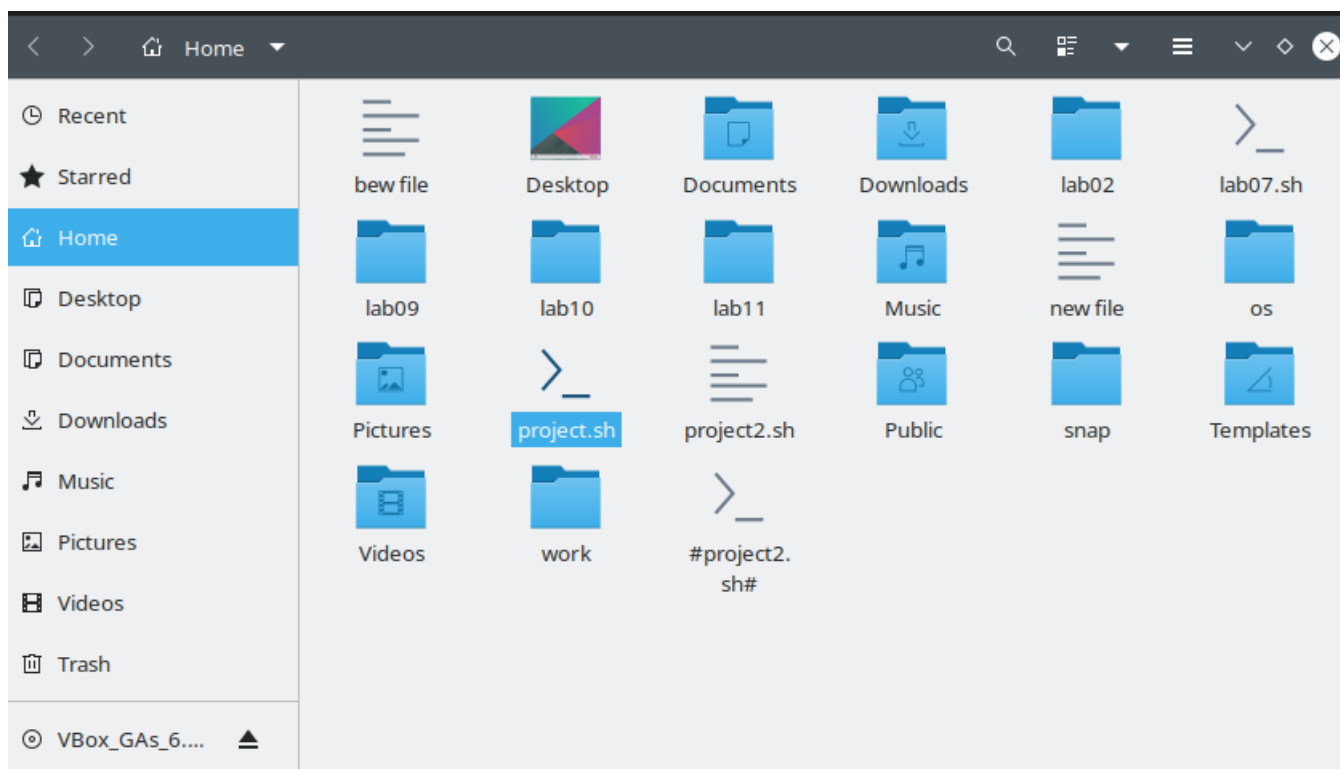
SYNOPSIS
    gzip [ -acdfhklLnNrtvV19 ] [-S suffix] [ name ... ]
    gunzip [ -acfhklLnNrtvV ] [-S suffix] [ name ... ]
    zcat [ -fhLV ] [ name ... ]

DESCRIPTION
    Gzip reduces the size of the named files using Lempel-Ziv coding
    (LZ77). Whenever possible, each file is replaced by one with the ex-
    tension .gz, while keeping the same ownership modes, access and modifi-
    cation times. (The default extension is z for MSDOS, OS/2 FAT, Windows
    NT FAT and Atari.) If no files are specified, or if a file name is
    "-", the standard input is compressed to the standard output. Gzip
    will only attempt to compress regular files. In particular, it will
    ignore symbolic links.

    If the compressed file name is too long for its file system, gzip trun-
    cates it. Gzip attempts to truncate only the parts of the file name
    longer than 3 characters. (A part is delimited by dots.) If the name
    Manual page gzip(1) line 1 (press h for help or q to quit)
```

- Создал текстовый файл *Project.sh*, используя команду *touch*. Открываю текстовый редактор *emacs*.

```
mokhammad@amin: ~  
mokhammad@amin:~$ man gzip  
mokhammad@amin:~$ touch project.sh  
mokhammad@amin:~$ chmod +x project.sh
```



- Написал пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Например, скрипт может последовательно распечатывать значения всех переданных аргументов.

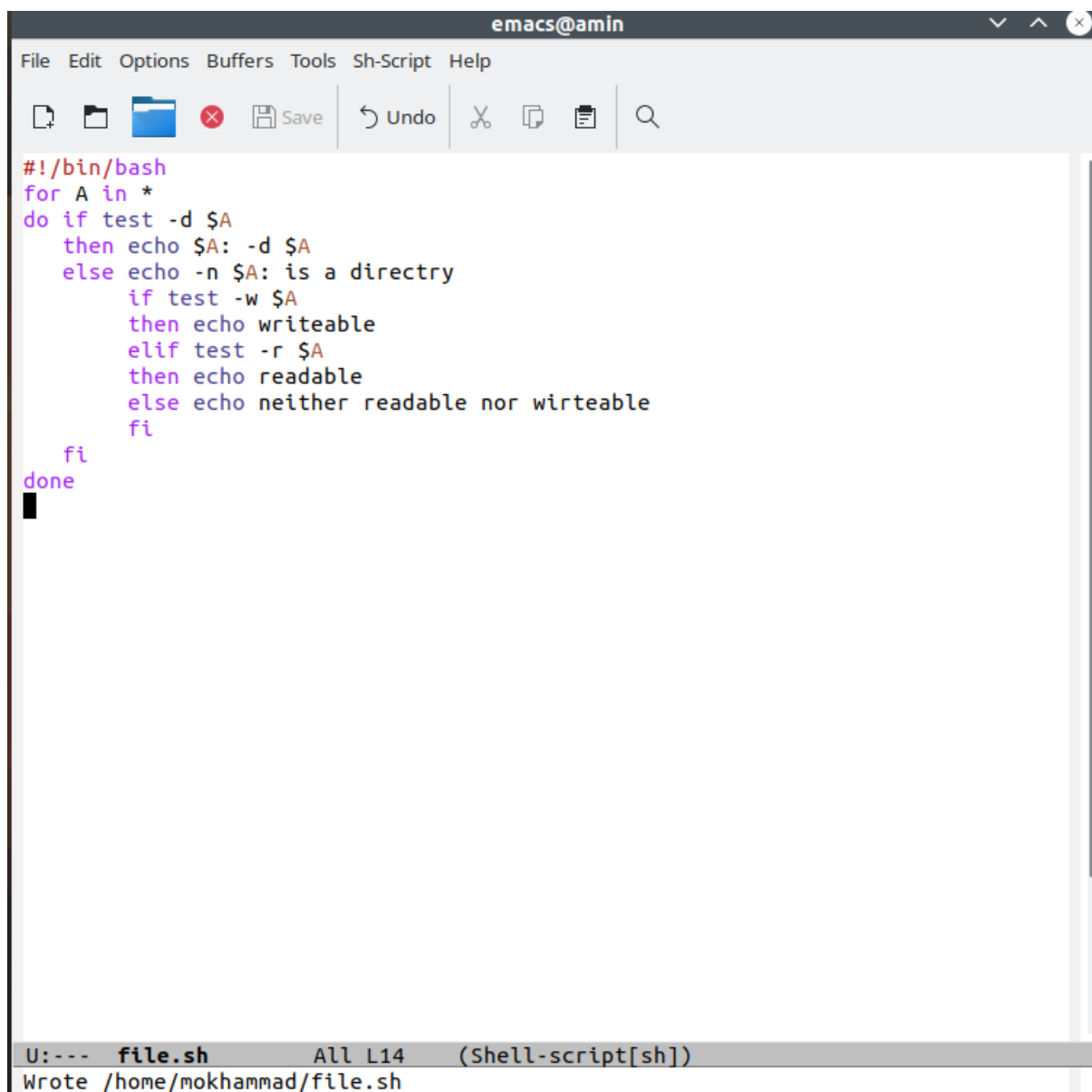
The screenshot shows the Emacs editor interface. The title bar reads 'emacs@amin'. The menu bar includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'Sh-Script', and 'Help'. The toolbar contains icons for file operations (new, open, save, close), editing (undo, redo, cut, copy, paste), and search. The main text area contains a shell script:

```
#!/bin/bash
echo "Enter value"
head -1
```

Below the script, a status bar indicates the current buffer: 'U: \*- project2.sh All L4 (Shell-script[sh])'. Below the status bar, the text 'End of buffer' is visible. The bottom part of the screenshot shows the terminal output of running the script:

```
mokhammad@amin:~$ chmod +x project2.sh
mokhammad@amin:~$ ./project2.sh
Enter value
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
mokhammad@amin:~$
```

- Написал 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.



The image shows a screenshot of an Emacs editor window. The title bar at the top reads "emacs@amin". Below the title bar is a menu bar with the following items: "File", "Edit", "Options", "Buffers", "Tools", "Sh-Script", and "Help". Underneath the menu bar is a toolbar containing icons for opening a file, saving a file, undo, redo, cut, copy, paste, and a search icon. The main editing area contains a shell script with the following content:

```
#!/bin/bash
for A in *
do if test -d $A
  then echo $A: -d $A
  else echo -n $A: is a directry
    if test -w $A
    then echo writeable
    elif test -r $A
    then echo readable
    else echo neither readable nor wirteable
    fi
  fi
done
```

The script is a bash script that iterates over all files and directories in the current directory. For each item, it checks if it is a directory. If it is, it prints the name followed by "-d". If it is not a directory, it prints the name followed by "is a directry". It then checks if the item is writeable, readable, or neither. The script has several typos: "directry" instead of "directory" and "wirteable" instead of "writeable".

At the bottom of the window, there is a status bar that displays the following information: "U:--- file.sh All L14 (Shell-script[sh])". Below the status bar, there is a message that says "Wrote /home/mokhammad/file.sh".

```
mokhammad@amin: ~  
mokhammad@amin:~$ chmod +x file.sh  
mokhammad@amin:~$ ./file.sh  
./file.sh: line 3: test: bew: binary operator expected  
bew file: is a directry./file.sh: line 6: test: bew: binary operator expected  
./file.sh: line 8: test: bew: binary operator expected  
neither readable nor wirteable  
Desktop: -d Desktop  
Documents: -d Documents  
Downloads: -d Downloads  
#file.sh#: is a directrywriteable  
file.sh: is a directrywriteable  
file.sh~: is a directrywriteable  
lab02: -d lab02  
lab07.sh: is a directrywriteable  
lab07.sh~: is a directrywriteable  
lab09: -d lab09  
lab10: -d lab10  
lab11: -d lab11  
Music: -d Music  
./file.sh: line 3: test: new: binary operator expected  
new file: is a directry./file.sh: line 6: test: new: binary operator expected  
./file.sh: line 8: test: new: binary operator expected  
neither readable nor wirteable  
os: -d os  
Pictures: -d Pictures  
project11.sh~: is a directrywriteable  
project2.sh: is a directrywriteable  
project2.sh~: is a directrywriteable  
project.sh: is a directrywriteable  
project.sh~: is a directrywriteable  
Public: -d Public  
snap: -d snap  
Templates: -d Templates  
Videos: -d Videos  
work: -d work  
mokhammad@amin:~$
```

- Написал 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.

```
emacs@amin
File Edit Options Buffers Tools Sh-Script Help
[Icons: New File, Open File, Save, Undo, Cut, Copy, Paste, Find]

#!/bin/bash
format=""
direct=""
echo "point out format"
read format
echo "point out direct"
read direct
find "$direct" -name ".$format" -type f | wc -l
ls
```

U:--- **project3.sh** All L9 (Shell-script[sh])  
Wrote /home/mokhammad/project3.sh

```
mokhammad@amin: ~
./project3.sh
point out format
.sh
point out direct
/home
0
'bew file'    file.sh    lab09    os    project3.sh    snap
Desktop      file.sh~  lab10    Pictures    project3.sh~  Templates
Documents    lab02     lab11    project11.sh~ project.sh    Videos
Downloads    lab07.sh  Music    project2.sh  project.sh~   work
'#file.sh#'  lab07.sh~ 'new file' project2.sh~  Public
mokhammad@amin:~$
```

## Выводы

Я изучил основы программирования в оболочке ОС UNIX/Linux, научился писать небольшие командные файлы.