Отчет по лабораторной работе №10

Алмазова Елизавета Андреевна

21.05.2022

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №10

Цель работы и задание

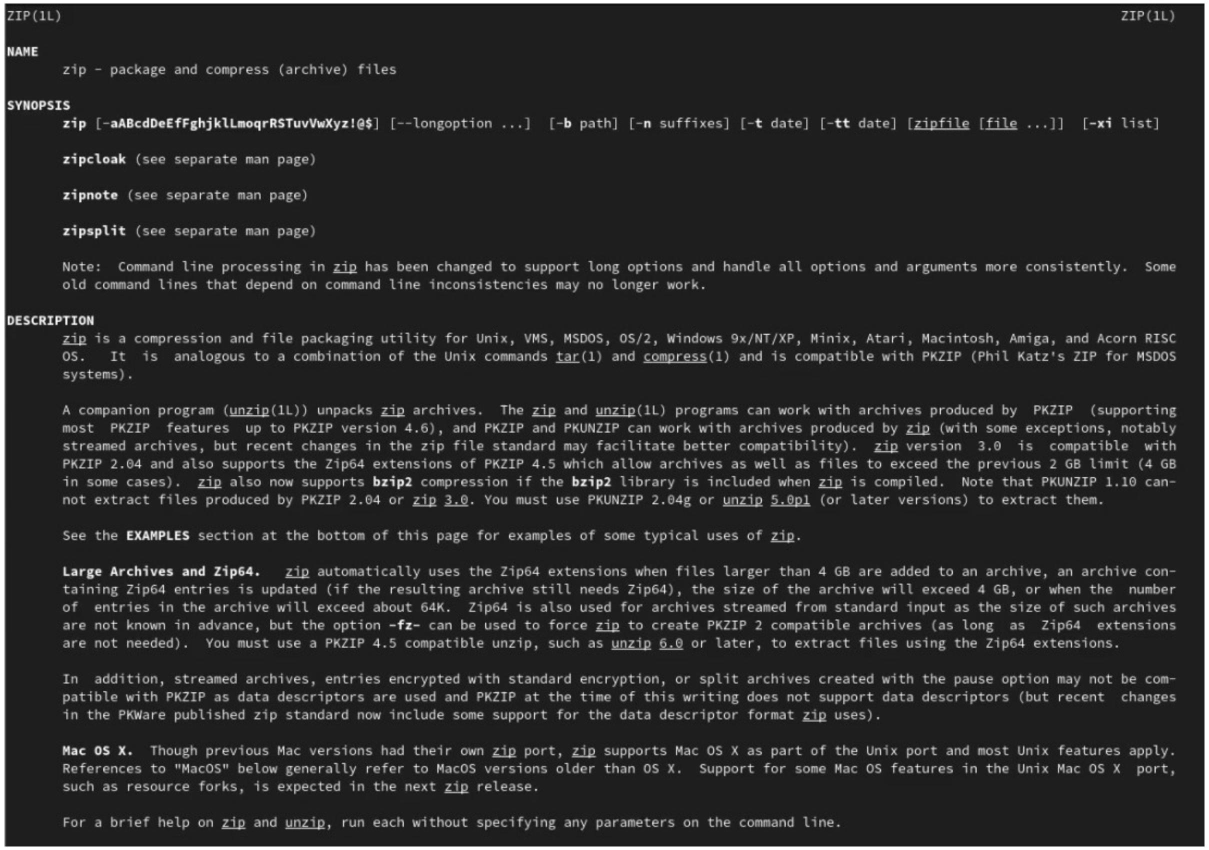
Цель данной лабораторной работы - изучить основы программирования в оболочке ОС UNIX/Linux, научиться писать небольшие командные файлы.

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

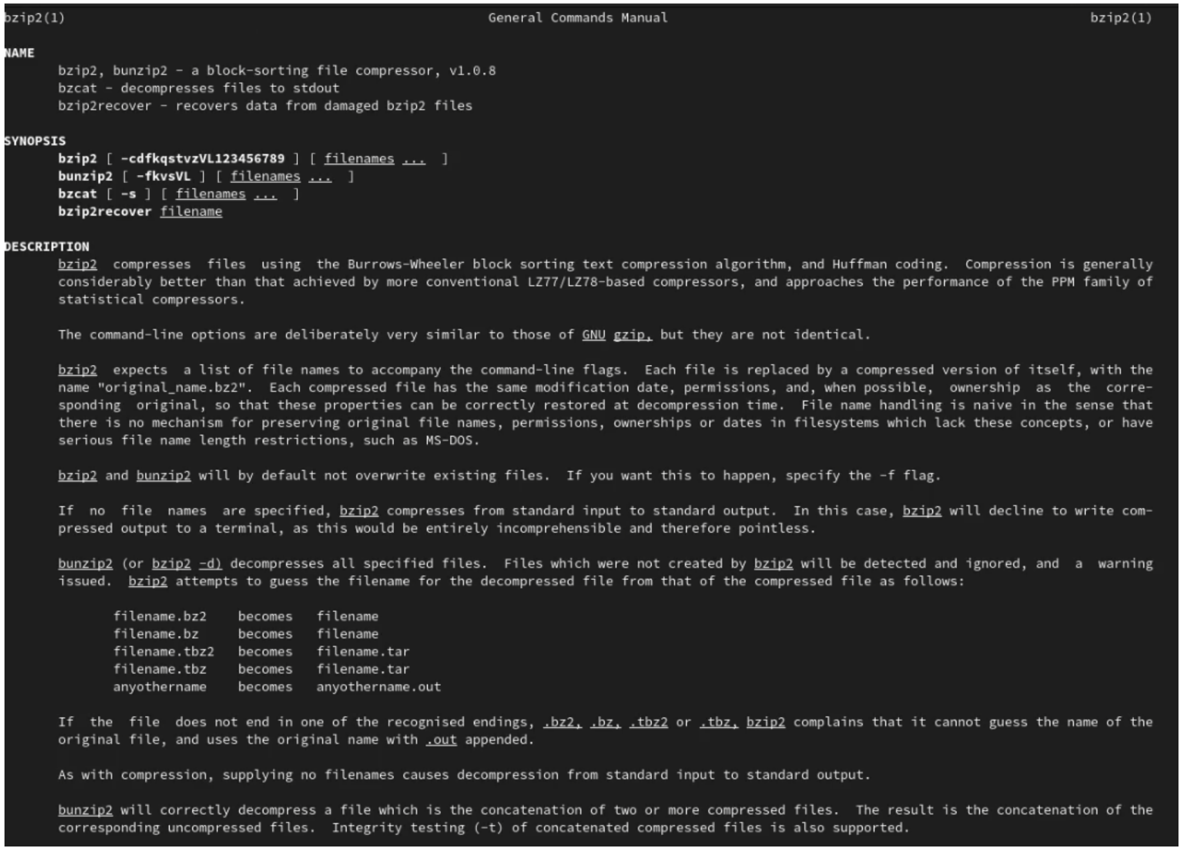
Ход работы

1. С помощью команды man изучала справку по командам архивации: man zip, man bzip2, man tar (рис.1,2,3).



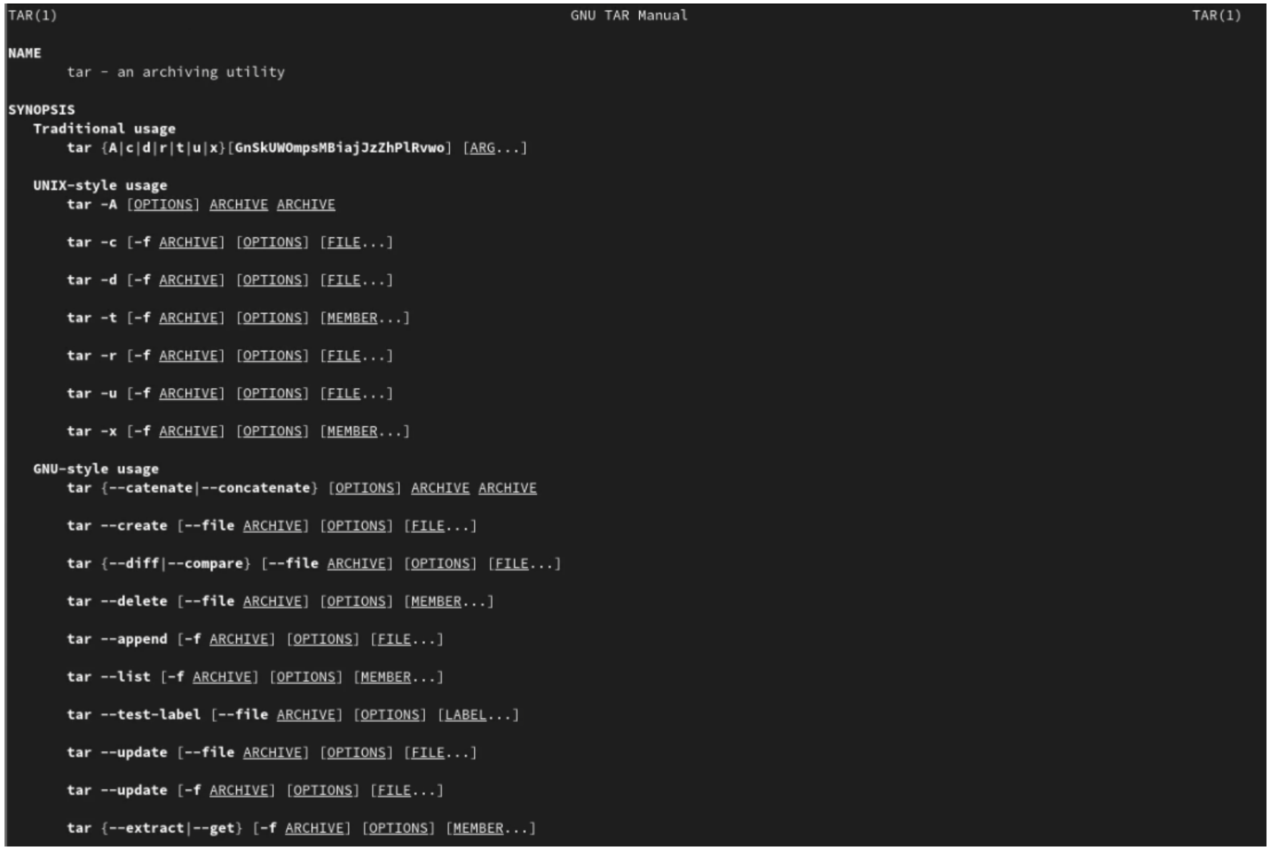
```
ZIP(1L)                                                                                                          ZIP(1L)

NAME
       zip - package and compress (archive) files

SYNOPSIS
       zip [-aABcdDeEfFghjklLmoqrRSTuvVwXyz!@$] [--longoption ...]  [-b path] [-n suffixes] [-t date] [-tt date] [zipfile [file ...]]  [-xi list]

       zipcloak (see separate man page)

       zipnote (see separate man page)

       zipsplit (see separate man page)

       Note:  Command line processing in zip has been changed to support long options and handle all options and arguments more consistently.  Some
       old command lines that depend on command line inconsistencies may no longer work.

DESCRIPTION
       zip is a compression and file packaging utility for Unix, VMS, MSDOS, OS/2, Windows 9x/NT/XP, Minix, Atari, Macintosh, Amiga, and Acorn RISC
       OS.   It  is  analogous to a combination of the Unix commands tar(1) and compress(1) and is compatible with PKZIP (Phil Katz's ZIP for MSDOS
       systems).

       A companion program (unzip(1L)) unpacks zip archives.  The zip and unzip(1L) programs can work with archives produced by  PKZIP  (supporting
       most  PKZIP  features  up to PKZIP version 4.6), and PKZIP and PKUNZIP can work with archives produced by zip (with some exceptions, notably
       streamed archives, but recent changes in the zip file standard may facilitate better compatibility).  zip version  3.0  is  compatible  with
       PKZIP 2.04 and also supports the Zip64 extensions of PKZIP 4.5 which allow archives as well as files to exceed the previous 2 GB limit (4 GB
       in some cases).  zip also now supports bzip2 compression if the bzip2 library is included when zip is compiled.  Note that PKUNZIP 1.10 can-
       not extract files produced by PKZIP 2.04 or zip 3.0. You must use PKUNZIP 2.04g or unzip 5.0p1 (or later versions) to extract them.

       See the EXAMPLES section at the bottom of this page for examples of some typical uses of zip.

       Large Archives and Zip64.   zip automatically uses the Zip64 extensions when files larger than 4 GB are added to an archive, an archive con-
       taining Zip64 entries is updated (if the resulting archive still needs Zip64), the size of the archive will exceed 4 GB, or when the  number
       of  entries in the archive will exceed about 64K.  Zip64 is also used for archives streamed from standard input as the size of such archives
       are not known in advance, but the option -fz- can be used to force zip to create PKZIP 2 compatible archives (as long  as  Zip64  extensions
       are not needed).  You must use a PKZIP 4.5 compatible unzip, such as unzip 6.0 or later, to extract files using the Zip64 extensions.

       In  addition, streamed archives, entries encrypted with standard encryption, or split archives created with the pause option may not be com-
       patible with PKZIP as data descriptors are used and PKZIP at the time of this writing does not support data descriptors (but recent  changes
       in the PKWare published zip standard now include some support for the data descriptor format zip uses).

       Mac OS X.  Though previous Mac versions had their own zip port, zip supports Mac OS X as part of the Unix port and most Unix features apply.
       References to "MacOS" below generally refer to MacOS versions older than OS X.  Support for some Mac OS features in the Unix Mac OS X  port,
       such as resource forks, is expected in the next zip release.

       For a brief help on zip and unzip, run each without specifying any parameters on the command line.
```

Рисунок 1 - man zip.



```
bzip2(1)                          General Commands Manual                          bzip2(1)

NAME
       bzip2, bunzip2 - a block-sorting file compressor, v1.0.8
       bzcat - decompresses files to stdout
       bzip2recover - recovers data from damaged bzip2 files

SYNOPSIS
       bzip2 [ -cdfkqstvzVL123456789 ] [ filenames ...  ]
       bunzip2 [ -fkvsVL ] [ filenames ...  ]
       bzcat [ -s ] [ filenames ...  ]
       bzip2recover filename

DESCRIPTION
       bzip2  compresses  files  using  the Burrows-Wheeler block sorting text compression algorithm, and Huffman coding.  Compression is generally
       considerably better than that achieved by more conventional LZ77/LZ78-based compressors, and approaches the performance of the PPM family of
       statistical compressors.

       The command-line options are deliberately very similar to those of GNU gzip, but they are not identical.

       bzip2  expects  a list of file names to accompany the command-line flags.  Each file is replaced by a compressed version of itself, with the
       name "original_name.bz2".  Each compressed file has the same modification date, permissions, and, when possible,  ownership  as  the  corre-
       sponding  original, so that these properties can be correctly restored at decompression time.  File name handling is naive in the sense that
       there is no mechanism for preserving original file names, permissions, ownerships or dates in filesystems which lack these concepts, or have
       serious file name length restrictions, such as MS-DOS.

       bzip2 and bunzip2 will by default not overwrite existing files.  If you want this to happen, specify the -f flag.

       If  no  file  names  are specified, bzip2 compresses from standard input to standard output.  In this case, bzip2 will decline to write com-
       pressed output to a terminal, as this would be entirely incomprehensible and therefore pointless.

       bunzip2 (or bzip2 -d) decompresses all specified files.  Files which were not created by bzip2 will be detected and ignored, and  a  warning
       issued.  bzip2 attempts to guess the filename for the decompressed file from that of the compressed file as follows:

              filename.bz2    becomes   filename
              filename.bz     becomes   filename
              filename.tbz2   becomes   filename.tar
              filename.tbz    becomes   filename.tar
              anyothername    becomes   anyothername.out

       If  the  file  does not end in one of the recognised endings, .bz2, .bz, .tbz2 or .tbz, bzip2 complains that it cannot guess the name of the
       original file, and uses the original name with .out appended.

       As with compression, supplying no filenames causes decompression from standard input to standard output.

       bunzip2 will correctly decompress a file which is the concatenation of two or more compressed files.  The result is the concatenation of the
       corresponding uncompressed files.  Integrity testing (-t) of concatenated compressed files is also supported.
```

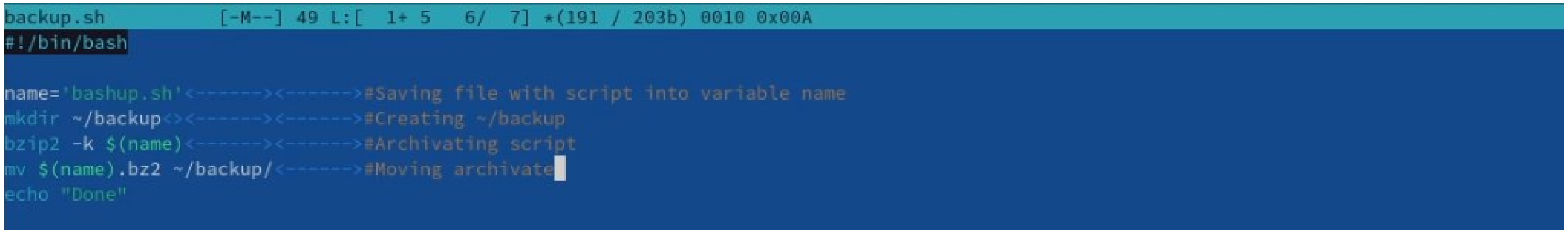
Рисунок 2 - man bzip2.



```
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...]

   GNU-style usage
       tar {--catenate|--concatenate} [OPTIONS] ARCHIVE ARCHIVE

       tar --create [--file ARCHIVE] [OPTIONS] [FILE...]

       tar {--diff|--compare} [--file ARCHIVE] [OPTIONS] [FILE...]

       tar --delete [--file ARCHIVE] [OPTIONS] [MEMBER...]

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

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

       tar --test-label [--file ARCHIVE] [OPTIONS] [LABEL...]

       tar --update [--file ARCHIVE] [OPTIONS] [FILE...]

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

       tar {--extract|--get} [-f ARCHIVE] [OPTIONS] [MEMBER...]
```

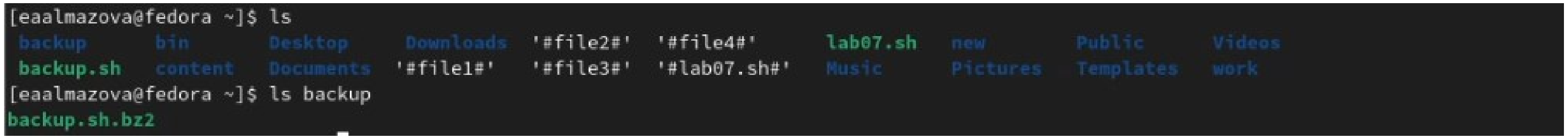
Рисунок 3 - man tar.

2. Написала скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию backup в моем домашнем каталоге, при этом файл архивируется bzip2. Сделала файл исполняемым и проверила работу скрипта(рис.4,5).



```
backup.sh          [-M--] 49 L:[  1+ 5   6/  7] *(191 / 203b) 0010 0x00A
#!/bin/bash

name='bashup.sh'<------><------>#Saving file with script into variable name
mkdir ~/backup<><------><------>#Creating ~/backup
bzip2 -k $(name)<------><------>#Archivating script
mv $(name).bz2 ~/backup/<------>#Moving archivate
echo "Done"
```

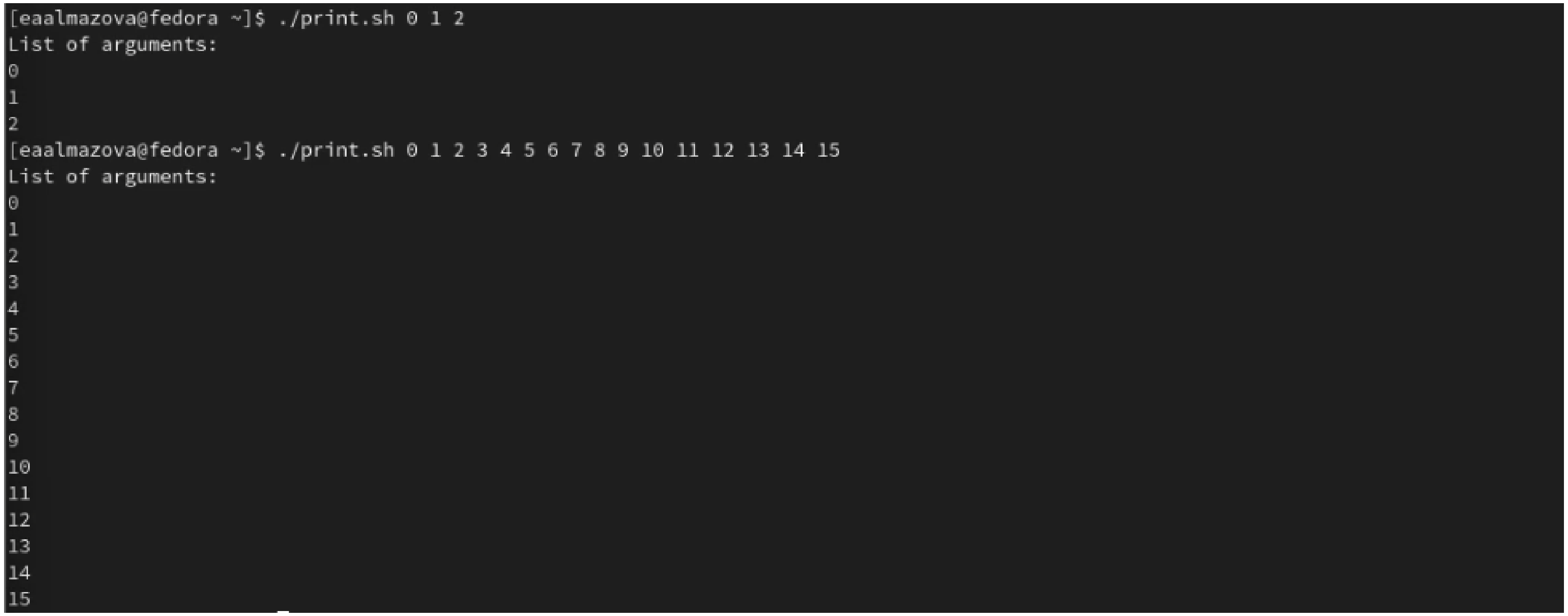
Рисунок 4 - Редактирование файла backup.sh.



```
[eaalmazova@fedora ~]$ ls
 backup      bin        Desktop     Downloads  '#file2#'  '#file4#'      lab07.sh   new        Public     Videos
 backup.sh   content    Documents  '#file1#'   '#file3#'  '#lab07.sh#'   Music      Pictures   Templates  work
[eaalmazova@fedora ~]$ ls backup
backup.sh.bz2
```

Рисунок 5 - Результат работы скрипта.

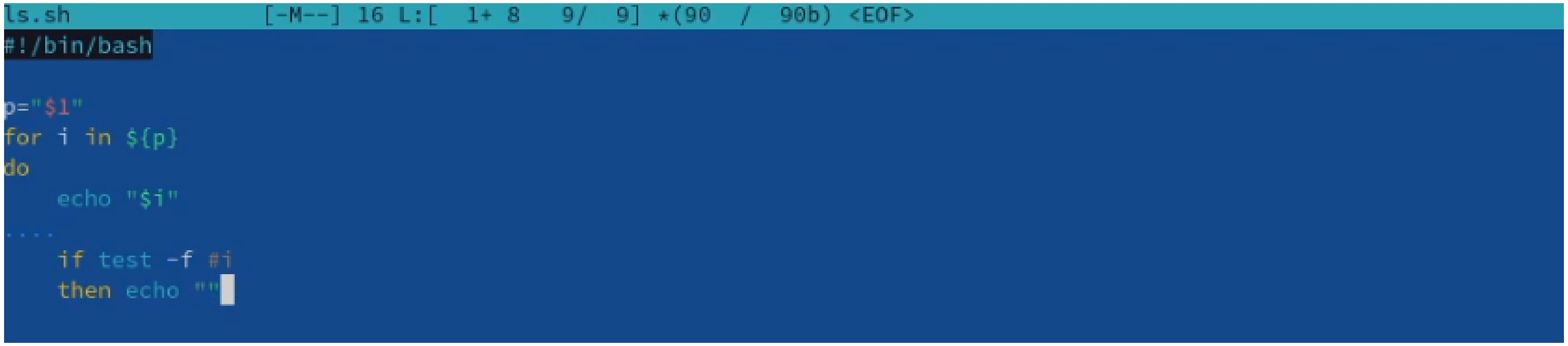
3. Написала пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять: скрипт может последовательно распечатывать значения всех переданных аргументов. Сделала файл исполняемым и проверила работу скрипта(рис.6).



```
[eaalmazova@fedora ~]$ ./print.sh 0 1 2
List of arguments:
0
1
2
[eaalmazova@fedora ~]$ ./print.sh 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
List of arguments:
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

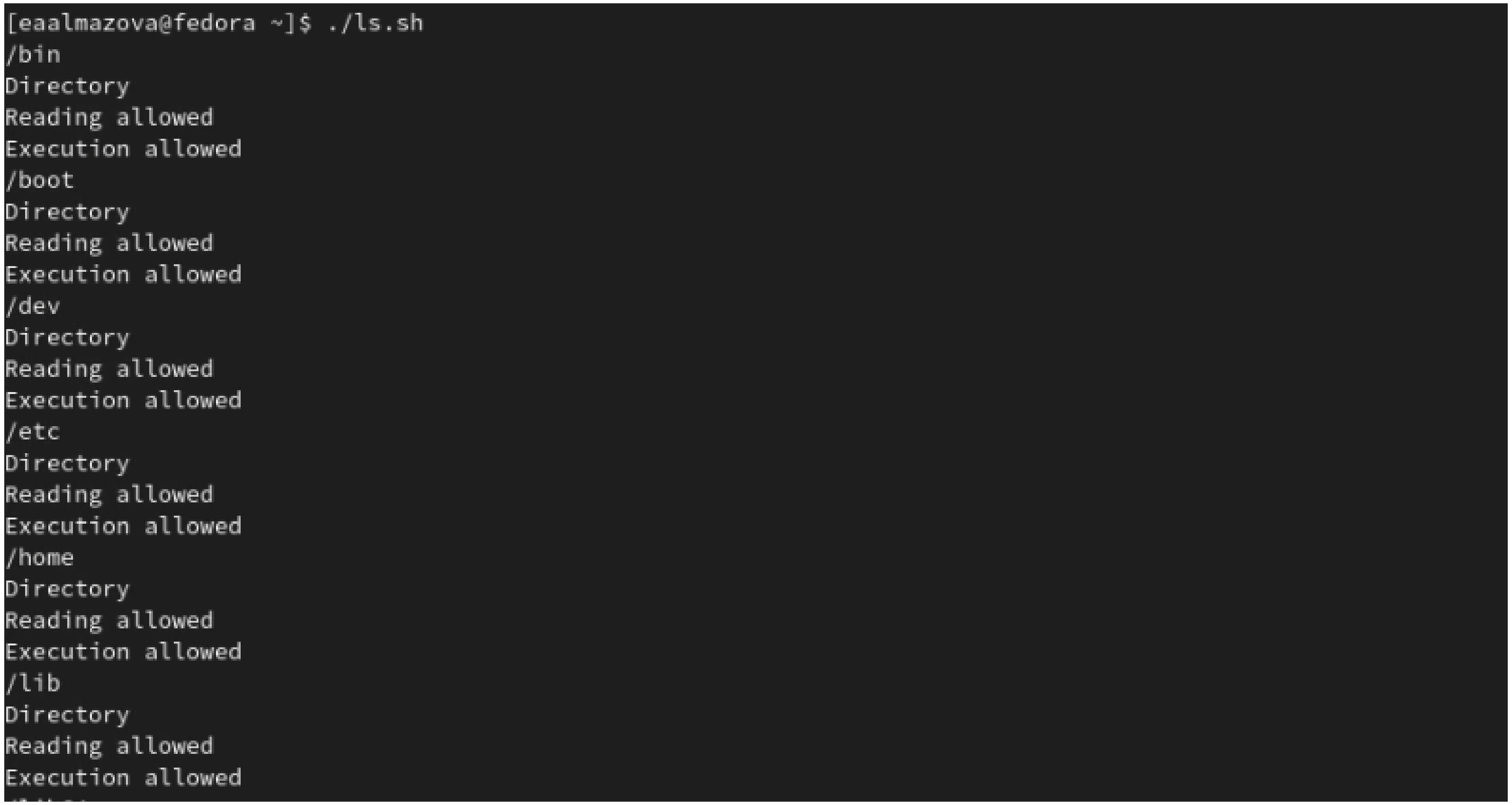
Рисунок 6 - Результат работы скрипта.

4. Написала командный файл — аналог команды ls (без использования самой этой команды и команды dir), чтобы он выдавал информацию о нужном каталоге и выводил информацию о возможностях доступа к файлам этого каталога. Сделала файл исполняемым и проверила работу скрипта (рис.7,8).



```
ls.sh                [-M--] 16 L:[  1+ 8   9/  9] *(90  /  90b) <EOF>
#!/bin/bash

p="$1"
for i in ${p}
do
    echo "$i"
....
    if test -f #i
    then echo ""
```

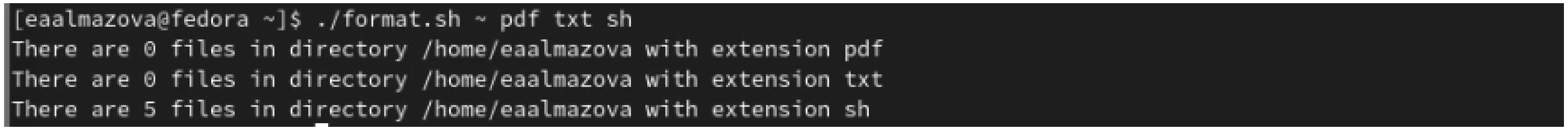
Рисунок 7 - Редактирование файла ls.sh.



```
[eaalmazova@fedora ~]$ ./ls.sh
/bin
Directory
Reading allowed
Execution allowed
/boot
Directory
Reading allowed
Execution allowed
/dev
Directory
Reading allowed
Execution allowed
/etc
Directory
Reading allowed
Execution allowed
/home
Directory
Reading allowed
Execution allowed
/lib
Directory
Reading allowed
Execution allowed
```

Рисунок 8 - Результат работы скрипта.

5. Написала 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. Сделала файл исполняемым и проверила работу скрипта(рис.9).



```
[eaalmazova@fedora ~]$ ./format.sh ~ pdf txt sh
There are 0 files in directory /home/eaalmazova with extension pdf
There are 0 files in directory /home/eaalmazova with extension txt
There are 5 files in directory /home/eaalmazova with extension sh
```

Рисунок 9 - Подсчет файлов домашнего каталога с расширениями pdf, txt, sh.

Выводы

В ходе выполнения данной лабораторной работы я изучила основы программирования в оболочке ОС UNIX/Linux, научилась писать небольшие командные файлы.

Спасибо за внимание!