

Laboratory №11

Krupennikova V.

MAY-2021

RUDN University, Moscow, Russian Federation

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

1. Ознакомиться с теоретическим материалом.
2. Изучить основы программирования в оболочке ОС UNIX/Linux.
3. Выполнить упражнения.
4. Ответить на контрольные вопросы.

- 1) Для начала я изучила команды архивации, используя команды «man zip», «man bzip2», «man tar»

```
vakrupennikova@dk4n60 ~ $ man zip  
vakrupennikova@dk4n60 ~ $ man bzip2  
vakrupennikova@dk4n60 ~ $ man tar
```

Figure 1: Команды архивации

Синтаксис команды zip для архивации файла: zip [опции] [имя файла.zip] [файлы или папки, которые будем архивировать]

Синтаксис команды zip для разархивации/распаковки файла: unzip[опции][файл_архива.zip][файлы]-x[исключить]-d[папка]

```
File: /usr/share/doc/zip/zip.1.gz
ZIP(1L)

NAME
    zip - package and compress (archive) files

SYNOPSIS
    zip [-aABcdDeffghjklmnpqRStuvWxyz015] [--longoption ...] [-b path] [-n suffixes] [-t date] [-tt date] [zipfile [file ...]] [-x 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 ar-
    chives, 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 cannot extract files produced by
    PKZIP 2.04 or zip 1.0. You must use PKUNZIP 2.04g or unzip 5.0n1 (or later versions) to extract them.

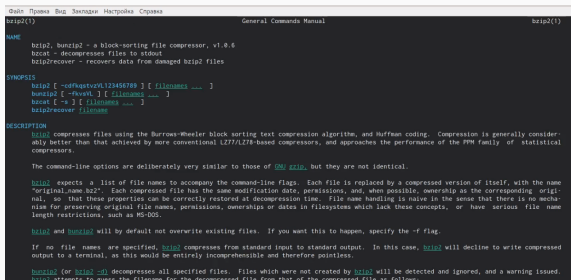
    See the EXAMPLES section at the bottom of this page for examples of some typical uses of zip.

    Large Archives and Zip64. zip automatically uses the Zip64 extensions when files larger than 4 GB are added to an archive, an archive containing
    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 65,000. Zip64 is also used for archives streamed from standard input as the size of such archives is not known in ad-
```

Figure 2: Синтаксис команды zip

Синтаксис команды bzip2 для архивации файла: bzip2 [опции] [имена файлов]

Синтаксис команды bzip2 для разархивации/распаковки файла: bunzip2[опции] [архивы.bz2]



```
Файл Правка Вых Загрузки Настройка Справка
bzip2(1)                                     General Commands Manual      bzip2(1)

NAME
    bzip2, bunzip2 - a block-sorting file compressor, v1.0.6
    bzcat - decompresses files to stdout
    bzip2recover - recovers data from damaged bzip2 files

SYNOPSIS
    bzip2 [ -cdRqstvzVL123456789 ] [ filenames ... ]
    bunzip2 [ -fkvvVL ] [ filenames ... ]
    bzcat [ -s ] [ filenames ... ]
    bzip2recover filenames

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 corresponding 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 compressed 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:
```

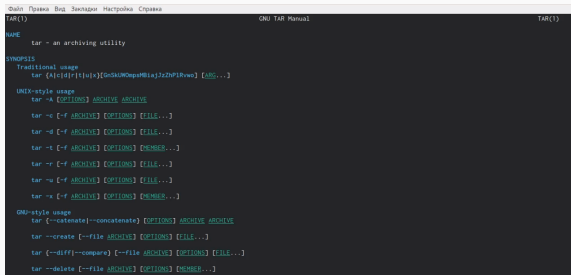
Figure 3: Синтаксис команды bzip2

Синтаксис команды tar для архивации файла:

tar[опции][архив.tar][файлы_для_архивации]

Синтаксис команды tar для разархивации/распаковки файла:

tar[опции][архив.tar]

A screenshot of a terminal window displaying the GNU TAR Manual. The window has a title bar with buttons for 'Ок', 'Правка', 'Выг', 'Закрыть', 'Настройка', and 'Справка'. The manual content is as follows:

```
TAR(1) GNU TAR Manual TAR(1)
NAME
    tar - an archiving utility
SYNOPSIS
    Traditional usage
    tar {A|c|d|r|t|u|x}[[G|S|M|W|p|H|a|j|z|Z|P|R|w|o] [dbs...]]
    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 [--concatenate|--concatenate] [OPTIONS] ARCHIVE ARCHIVE
    tar --create [--file ARCHIVE] [OPTIONS] [FILE...]
    tar [--diff|--compare] [--file ARCHIVE] [OPTIONS] [FILE...]
    tar --delete [--file ARCHIVE] [OPTIONS] [MEMBER...]
```

Figure 4: Синтаксис команды tar

- 2) Далее я создала файл, в котором буду писать первый скрипт, и открыла его в редакторе emacs, используя клавиши «Ctrl-x» и «Ctrl-f»(команды «touchbackup.sh» и «emacs&»)

```
vakrupennikova@dk4n60 ~ $ touch backup.sh  
vakrupennikova@dk4n60 ~ $ emacs &
```

Figure 5: Создание файла

- 3) После 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 вашем домашнем каталоге. При этом файл должен архивироваться одним из архиваторов на выбор zip, bzip2 или tar. При написании скрипта использовала архиватор bzip2.

```
File Edit Options Buffers Tools Sh-Sc  
#!/bin/bash
```


4) Проверила работу скрипта (команда «./backup.sh»), предварительно добавив для него право на выполнение (команда «chmod+x*.sh»). Проверила, появился ли каталог backup/, перейдя в него (команда «cd backup/»), посмотрела его содержимое (команда «ls») и просмотрела содержимое архива (команда «bunzip2 -c backup.sh.bz2»). Скрипт работает корректно.

```
vakrupennikova@dk4n00 ~ $ ls
2021-05-28 13:18-18_ekv' backup.sh  GNUstep  labOS    public_html  Видео  Загрузки  Музыка  проекты  Шаблоны
backup.sh      docs    image    public    tar         Документы  Изображения  Общедоступные  'Рабочий стол'
vakrupennikova@dk4n00 ~ $ chmod +x *.sh
vakrupennikova@dk4n00 ~ $ ./backup.sh
Выполнено
vakrupennikova@dk4n00 ~ $ cd backup/
vakrupennikova@dk4n00 ~/backup $ ls
backup.sh.bz2
```

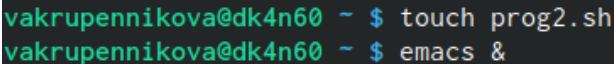
Figure 7: Проверка работы скрипта

```
vakrupennikova@dk4n60 ~/backup $ bunzip2 -c backup.sh.bz2
#!/bin/bash

name='backup.sh'
mkdir ~/backup
bzip2 -k ${name}
mv ${name}.bz2 ~/backup/
echo "Выполнено"
```

Figure 8: Вывод скрипта

- 5) Создала файл, в котором буду писать второй скрипт, и открыла его в редакторе emacs, используя клавиши «Ctrl-x» и «Ctrl-f» (команды «touchprog2.sh» и «emacs&»)

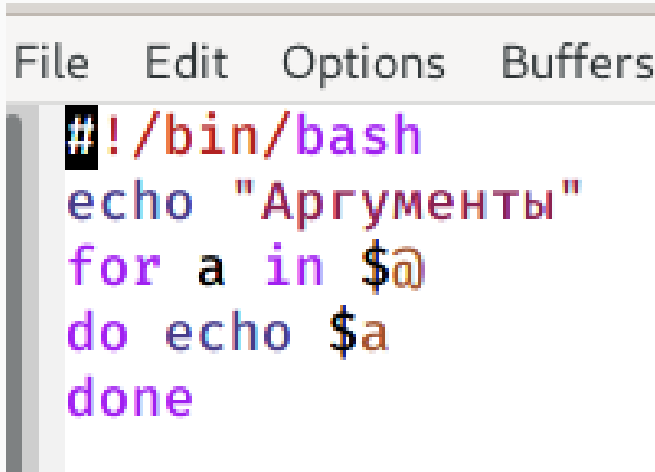


```
vakrupennikova@dk4n60 ~ $ touch prog2.sh  
vakrupennikova@dk4n60 ~ $ emacs &
```

A terminal window with a dark background. The prompt is 'vakrupennikova@dk4n60 ~ \$'. The first command is 'touch prog2.sh' and the second is 'emacs &'. The output is not visible.

Figure 9: Создание файла

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

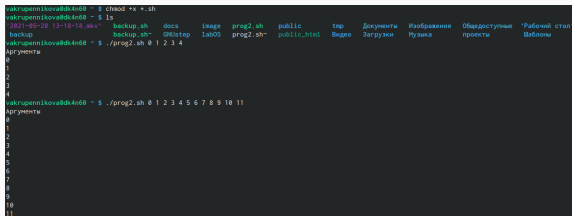
A screenshot of a text editor window. The title bar at the top contains the text "File Edit Options Buffers". The editor area displays a shell script with the following lines:

```
#!/bin/bash  
echo "Аргументы"  
for a in $@  
do echo $a  
done
```

 The script is written in a monospaced font with syntax highlighting: the shebang line is black, "echo" is blue, "Аргументы" is red, "for" is blue, "a" is black, "in" is blue, "\$@" is black, "do" is blue, "echo" is blue, "\$a" is black, and "done" is blue.

Figure 10: Второй скрипт

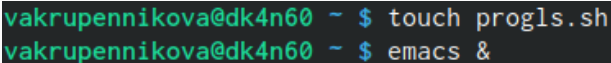
7) Проверила работу написанного скрипта (команды «./prog2.sh 0 1 2 3 4» и «./prog2.sh 0 1 2 3 45 6 7 8 9 10 11»), предварительно добавив для него право на выполнение (команда «chmod+x*.sh»). Вводила аргументы, количество которых меньше 10 и больше 10. Скрипт работает корректно.



```
akr@krupennikova@dk4n60 ~$ chmod +x *.sh
akr@krupennikova@dk4n60 ~$ ls
2021-05-28 13-18-19.mkv  backup.sh  docs  image  prog2.sh  public  tmp  Документы  Изображения  Общедоступные  Рабочий стол
backup                  backup.sh  Gksstep  lab03  prog2.sh~  public_html  Видео  Загрузки  Музыка  проекты  Шаблоны
akr@krupennikova@dk4n60 ~$ ./prog2.sh 0 1 2 3 4
Аргументы
0
1
2
3
4
akr@krupennikova@dk4n60 ~$ ./prog2.sh 0 1 2 3 4 5 6 7 8 9 10 11
Аргументы
0
1
2
3
4
5
6
7
8
9
10
11
```

Figure 11: Проверка работы скрипта

- 8) Создала файл, в котором буду писать третий скрипт, и открыла его в редакторе emacs, используя клавиши «Ctrl-x» и «Ctrl-f» (команды «touchprogl.sh» и «emacs&»)

A terminal window with a dark background. The prompt is 'vakrupennikova@dk4n60 ~ \$'. The first command entered is 'touch progl.sh' and the second is 'emacs &'.

```
vakrupennikova@dk4n60 ~ $ touch progl.sh
vakrupennikova@dk4n60 ~ $ emacs &
```

Figure 12: Создание файла

- 9) Написала командный файл –аналог команды ls (без использования самой этой команды и команды dir). Он должен выдавать информацию о нужном каталоге и выводить информацию о возможностях доступа к файлам этого каталога

File Edit Options Buffers Tools Sh-Script Help

```
#!/bin/bash
a="$1"
for i in ${a}/*
do
    echo "$1"

    if test -f $i
    then echo "Обычный файл"
    fi

    if test -d $i
    then echo "Каталог"
    fi

    if test -r $i
    then echo "Чтение разрешено"
    fi

    if test -w $i
    then echo "Запись разрешена"
    fi

    if test -x $i
```

- 10) Далее проверила работу скрипта (команда «./progl.sh~»), предварительно добавив для него право на выполнение (команда «chmod+x*.sh»). Скрипт работает корректно.

```
vakrupennikova@dk4n60 ~ $ chmod +x *.sh
[1] Завершён emacs
[2]- Завершён emacs
vakrupennikova@dk4n60 ~ $ ls
'2021-05-28 13:18-18.mkv'  backup.sh~  image      prog2.sh~  public      Видео      Изображения  проекты
backup                    docs        lab05      progl.sh  public_html  Документы  Музыка      'Рабочий стол'
backup.sh                 GNUstep     prog2.sh~  progl.sh~  tmp          Загрузки    Общиедупные  Шаблоны
vakrupennikova@dk4n60 ~ $ ./progl.sh ~
/afs/.dk.sci.pfu.edu.ru/home/v/a/vakrupennikova
./progl.sh: строка 7: test: /afs/.dk.sci.pfu.edu.ru/home/v/a/vakrupennikova/2021-05-28: ожидается бинарный оператор
./progl.sh: строка 11: test: /afs/.dk.sci.pfu.edu.ru/home/v/a/vakrupennikova/2021-05-28: ожидается бинарный оператор
./progl.sh: строка 15: test: /afs/.dk.sci.pfu.edu.ru/home/v/a/vakrupennikova/2021-05-28: ожидается бинарный оператор
./progl.sh: строка 19: test: /afs/.dk.sci.pfu.edu.ru/home/v/a/vakrupennikova/2021-05-28: ожидается бинарный оператор
./progl.sh: строка 23: test: /afs/.dk.sci.pfu.edu.ru/home/v/a/vakrupennikova/2021-05-28: ожидается бинарный оператор
/afs/.dk.sci.pfu.edu.ru/home/v/a/vakrupennikova
Каталог
Чтение разрешено
Запись разрешена
Выполнение разрешено
/afs/.dk.sci.pfu.edu.ru/home/v/a/vakrupennikova
Обычный файл
Чтение разрешено
Запись разрешена
Выполнение разрешено
/afs/.dk.sci.pfu.edu.ru/home/v/a/vakrupennikova
Обычный файл
Чтение разрешено
Запись разрешена
Выполнение разрешено
/afs/.dk.sci.pfu.edu.ru/home/v/a/vakrupennikova
Каталог
Чтение разрешено
Запись разрешена
Выполнение разрешено
/afs/.dk.sci.pfu.edu.ru/home/v/a/vakrupennikova
Обычный файл
Чтение разрешено
Запись разрешена
Выполнение разрешено
/afs/.dk.sci.pfu.edu.ru/home/v/a/vakrupennikova
Каталог
Чтение разрешено
Запись разрешена
Выполнение разрешено
/afs/.dk.sci.pfu.edu.ru/home/v/a/vakrupennikova
```

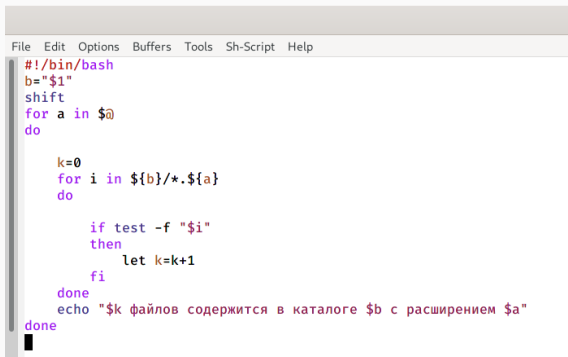
Figure 14: Проверка работы скрипта

- 11) Для четвертого скрипта также создала файл (команда «touchformat.sh»)и открыла его в редакторе emacs, используя клавиши «Ctrl-x» и «Ctrl-f» (команда «emacs&»)

```
vakrupennikova@dk4n60 ~ $ touch format.sh  
vakrupennikova@dk4n60 ~ $ emacs &
```

Figure 15: Создание файла

- 12) Написала командный файл, который получает в качестве аргумента командной строки формат файла (.txt, .doc, .jpg, .pdfи т.д.) и вычисляет количество таких файлов в указанной директории. Путь к директории также передаётся в виде аргумента командной строки



```
#!/bin/bash
b="$1"
shift
for a in $@
do
    k=0
    for i in ${b}/*.${a}
    do
        if test -f "$i"
        then
            let k=k+1
        fi
    done
    echo "$k файлов содержится в каталоге $b с расширением $a"
done
```

Figure 16: Четвертый скрипт

13) Проверила работу написанного скрипта (команда «./format.sh-pdfshtxt.doc»), предварительно добавив для него право на выполнение (команда «chmod+x*.sh»), а также создав дополнительные файлы с разными расширениями (команда «touchfile.pdf file1.doc file2.doc»). Скрипт работает корректно.

```
vakrurpennikov@sk4n68 ~$ chmod +x *.sh
vakrurpennikov@sk4n68 ~$ touch file.pdf file1.doc file2.doc
vakrurpennikov@sk4n68 ~$ ls
2021-05-20 12:10:10 .bakup backup.sh file2.doc format.sh labOS progls.sh public.html Документы Музыка 'Рабочий стол'
backup docs file1.pdf GMailstep prog1.sh progls.sh tar шаблоны
backup.sh file1.doc format.sh image prog2.sh public Видео
vakrurpennikov@sk4n68 ~$ ./format.sh - pdf sh txt doc
1 файл(ов) содержится в каталоге /afs/.dk.sci.pfu.edu.ru/home/v/a/vakrurpennikova с расширением pdf
4 файл(ов) содержится в каталоге /afs/.dk.sci.pfu.edu.ru/home/v/a/vakrurpennikova с расширением sh
3 файл(ов) содержится в каталоге /afs/.dk.sci.pfu.edu.ru/home/v/a/vakrurpennikova с расширением txt
2 файл(ов) содержится в каталоге /afs/.dk.sci.pfu.edu.ru/home/v/a/vakrurpennikova с расширением doc
vakrurpennikov@sk4n68 ~$
```

Figure 17: Проверка работы скрипта

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

The end.