

# Отчет

Лабораторная работа №10

Павлова Варвара Юрьевна НПМбд-02-21

# Содержание

Цель работы	5
Задание	6
Теоретическое введение	7
Выполнение лабораторной работы	8
Выводы	14
Список литературы	15

## Список иллюстраций

0.1	вызов справки . . . . .	8
0.2	man zip . . . . .	8
0.3	man bzip2 . . . . .	9
0.4	man tar . . . . .	9
0.5	создание файла . . . . .	10
0.6	написание скрипта . . . . .	10
0.7	проверка первого файла . . . . .	10
0.8	создание файла . . . . .	10
0.9	написание скрипта . . . . .	11
0.10	проверка второго файла . . . . .	11
0.11	создание файла . . . . .	11
0.12	написание скрипта . . . . .	12
0.13	проверка третьего файла . . . . .	12
0.14	создание файла . . . . .	13
0.15	написание скрипта . . . . .	13
0.16	проверка четвертого файла . . . . .	13

## Список таблиц

## Цель 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 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.
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.

# Теоретическое введение

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 оболочке Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд; - оболочка Корна (или ksh) — 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). 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. Рассмотрим основные элементы программирования в оболочке bash. В других оболочках большинство команд будет совпадать с описанными ниже.

# Выполнение лабораторной работы

1. Изучаю информацию о командах архивации zip(рис. [-@fig:002]), bzip2 (рис. [-@fig:003]) и tar (рис. [-@fig:004]) с помощью команды man.(рис. [-@fig:001])

```
[vypavlova@fedora ~]$ man zip
[vypavlova@fedora ~]$ man bzip2
[vypavlova@fedora ~]$ man tar
[vypavlova@fedora ~]$
```

Рис. 0.1: вызов справки

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

Рис. 0.2: man zip



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

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

SYNOPSIS
    bzip2 [ -cdfkqstvzVL123456789 ] [ filenames ... ]
    bunzip2 [ -fkvsVL ] [ 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.
```

Рис. 0.3: 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}[GnSkUW0mpsMBiajJzZhPlRvwo] [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)
```

Рис. 0.4: man tar

2. Создаю первый исполняемый файл backup.sh и открываю редактор emacs. (рис. [-@fig:005])

```
[vypavlova@fedora ~]$ touch backup.sh
[vypavlova@fedora ~]$ emacs &
[1] 2933
```

Рис. 0.5: создание файла

- Пишу скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию backup в домашнем каталоге. При этом файл архивируется архиватором bzip2. (рис. [-@fig:006])

```
#!/bin/bash
name='backup.sh'
mkdir ~/backup
bzip2 -k ${name}
mv ${name}.bz2 ~/backup/
echo "done"
```

Рис. 0.6: написание скрипта

- Добавляю право на выполнение файла и проверяю его работу. (рис. [-@fig:007])

```
[vypavlova@fedora ~]$ chmod +x *.sh
[vypavlova@fedora ~]$ ./backup.sh
done
[vypavlova@fedora ~]$ cd backup
[vypavlova@fedora backup]$ ls
backup.sh.bz2
[vypavlova@fedora backup]$ bunzip2 -c backup.sh.bz2
#!/bin/bash

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

Рис. 0.7: проверка первого файла

- Создаю второй исполняемый файл second.sh и открываю редактор emacs. (рис. [-@fig:008])

```
[vypavlova@fedora ~]$ touch second.sh
[vypavlova@fedora ~]$ emacs &
```

Рис. 0.8: создание файла

6. Пишу командный файл, обрабатывающий любое произвольное число аргументов командной строки, в том числе превышающее десять. Скрипт последовательно распечатывает значения всех переданных аргументов. (рис. [-@fig:009])

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

Рис. 0.9: написание скрипта

7. Добавляю право на выполнение файла и проверяю его работу. (рис. [-@fig:010])

```
[vypavlova@fedora ~]$ chmod +x *.sh
[vypavlova@fedora ~]$ ./second.sh 1 2 3 4
numbers
1
2
3
4
[vypavlova@fedora ~]$ ./second.sh 1 2 3 4 5 6 7 8 9 10 11
numbers
1
2
3
4
5
6
7
8
9
10
11
```

Рис. 0.10: проверка второго файла

8. Создаю третий исполняемый файл third.sh и открываю редактор emacs. (рис. [-@fig:011])

```
[vypavlova@fedora ~]$ touch third.sh
[vypavlova@fedora ~]$ emacs &
```

Рис. 0.11: создание файла

9. Пишу командный файл — аналог команды ls (без использования самой этой команды и команды dir). Скрипт выдает информацию о нужном каталоге и

выводит информацию о возможностях доступа к файлам этого каталога. (рис. [-@fig:012])

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

    if test -f $i
    then echo "regular file"
    fi

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

    if test -r $i
    then echo "reading is allowed"
    fi

    if test -w $i
    then echo "writing is allowed"
    fi
done
```

Рис. 0.12: написание скрипта

10. Добавляю право на выполнение файла и проверяю его работу. (рис. [-@fig:013])

```
[vypavlova@fedora ~]$ chmod +x *.sh
[vypavlova@fedora ~]$ ./third.sh ~
/home/vypavlova/backup
directory
reading is allowed
writing is allowed
/home/vypavlova/backup.sh
regular file
reading is allowed
writing is allowed
/home/vypavlova/backup.sh~
regular file
reading is allowed
writing is allowed
/home/vypavlova/bin
directory
reading is allowed
writing is allowed
/home/vypavlova/Desktop
directory
reading is allowed
writing is allowed
/home/vypavlova/Documents
```

Рис. 0.13: проверка третьего файла

11. Создаю четвертый исполняемый файл fourth.sh и открываю редактор emacs. (рис. [-@fig:014])

```
[vypavlova@fedora ~]$ touch fourth.sh
[vypavlova@fedora ~]$ emacs &
[1] 5210
```

Рис. 0.14: создание файла

12. Пишу 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. (рис. [-@fig:015])

```
#!/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 $a files in $b directory"
done
```

Рис. 0.15: написание скрипта

13. Добавляю право на выполнение файла и проверяю его работу. (рис. [-@fig:016])

```
[vypavlova@fedora ~]$ chmod +x *.sh
[vypavlova@fedora ~]$ ls
backup Desktop fourth.sh~ opsystemlab second.sh third.sh~
backup.sh Documents lab07.sh Pictures second.sh~ Videos
backup.sh~ Downloads Music Public Templates work
bin fourth.sh my_os reports third.sh
[vypavlova@fedora ~]$ touch 1.txt 2.txt 3.txt
[vypavlova@fedora ~]$ touch 24.pdf u.pdf
[vypavlova@fedora ~]$ ./fourth.sh ~ txt pdf sh
3 txt files in /home/vypavlova directory
2 pdf files in /home/vypavlova directory
5 sh files in /home/vypavlova directory
[vypavlova@fedora ~]$ ls
1.txt backup Desktop fourth.sh~ opsystemlab second.sh third.sh~
24.pdf backup.sh Documents lab07.sh Pictures second.sh~ u.pdf
2.txt backup.sh~ Downloads Music Public Templates Videos
3.txt bin fourth.sh my_os reports third.sh work
```

Рис. 0.16: проверка четвертого файла

## Выводы

Выполняя данную лабораторную работу я изучила основы программирования в оболочке ОС UNIX/Linux и научилась писать небольшие командные файлы.

## Список литературы