

# Отчёт по лабораторной работе №10

Операционные системы

Балханова Алтана Юрьевна

# Содержание

Цель работы	5
Выполнение лабораторной работы	6
Контрольные вопросы	17
Выводы	21

## Список иллюстраций

0.1	Создание файла . . . . .	6
0.2	Справка команды <code>zip</code> . . . . .	7
0.3	Справка команды <code>bzip2</code> . . . . .	7
0.4	Справка команды <code>tar</code> . . . . .	8
0.5	Резервное копирование . . . . .	9
0.6	Работа скрипта <code>backup.sh</code> . . . . .	10
0.7	Печать аргументов . . . . .	11
0.8	Работа командного файла . . . . .	12
0.9	<code>ls</code> . . . . .	13
0.10	Работа аналога <code>ls</code> . . . . .	14
0.11	Командный файл для подсчёта нужных файлов . . . . .	15
0.12	Работа . . . . .	16

## Список таблиц

## Цель работы

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

# Выполнение лабораторной работы

1. Создала файл backup.sh, используя комбинации клавиш C-x C-f (рис. 0.1).

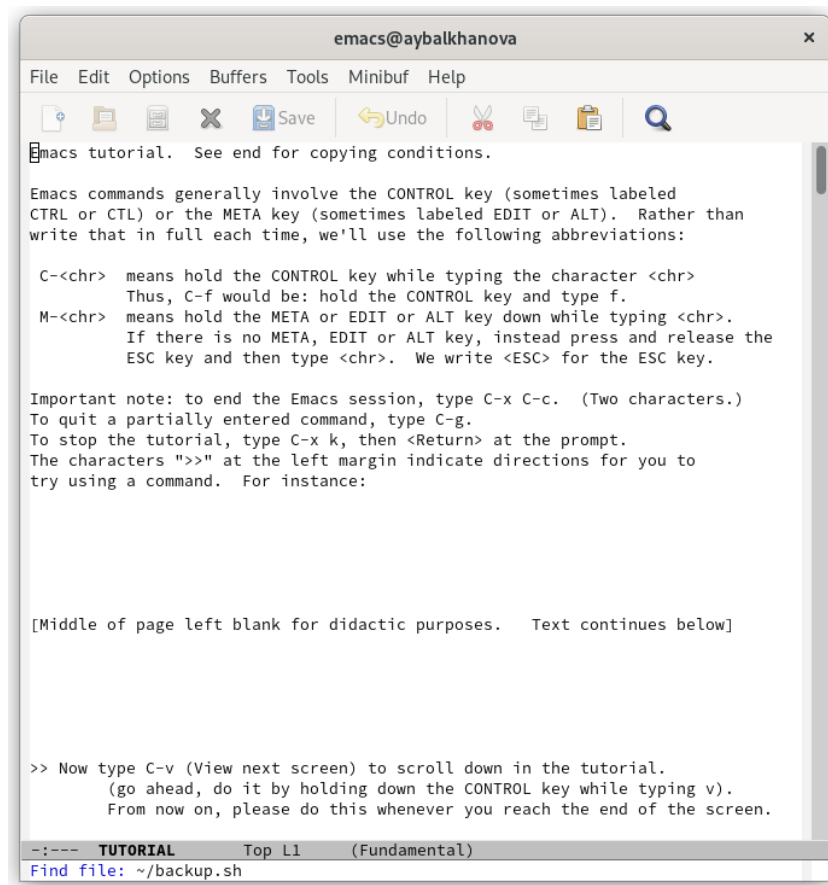


Рис. 0.1: Создание файла

2. Прочитала справки архиваторов zip, bzip2, tar (рис. 0.2, 0.3, 0.4).

```
aybalkhanova@aybalkhanova:~ — man zip
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: Справка команды zip

```
aybalkhanova@aybalkhanova:~ — man bzip2
bzip2(1) General Commands Manual bzip2(1)

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

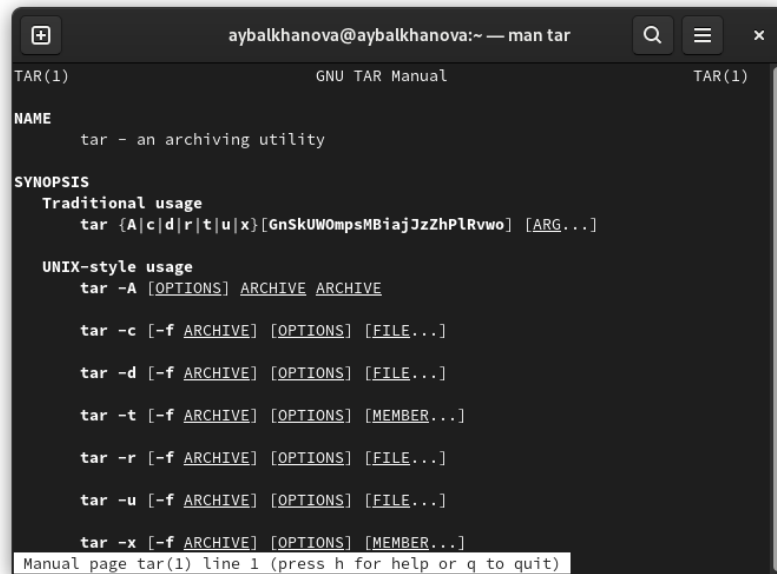
SYNOPSIS
    bzip2 [-cdfkqstzVL123456789] [filenames ...]
    bunzip2 [-fkvsVL] [filenames ...]
    bzip2recover [-s] [filenames ...]
    bzip2recover filename

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.

    Manual page bzip2(1) line 1 (press h for help or q to quit)
```

Рис. 0.3: Справка команды 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...]
Manual page tar(1) line 1 (press h for help or q to quit)
```

Рис. 0.4: Справка команды tar

3. Написала скрипт, который делает резервную копию самого себя и отправляет его в директорию backup (рис. 0.5).



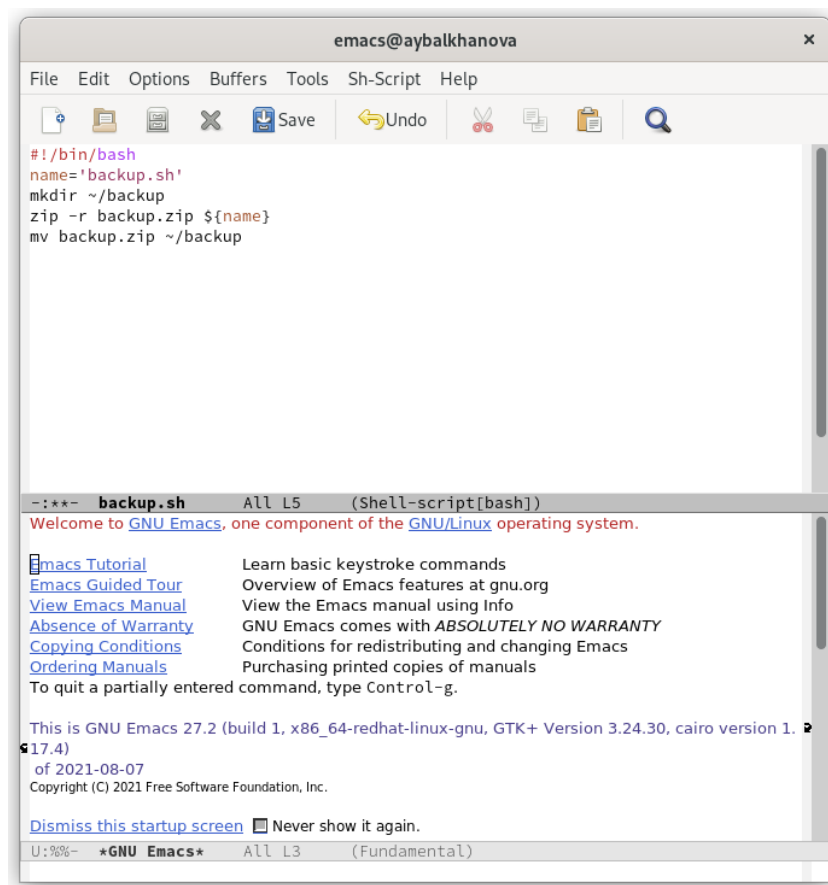
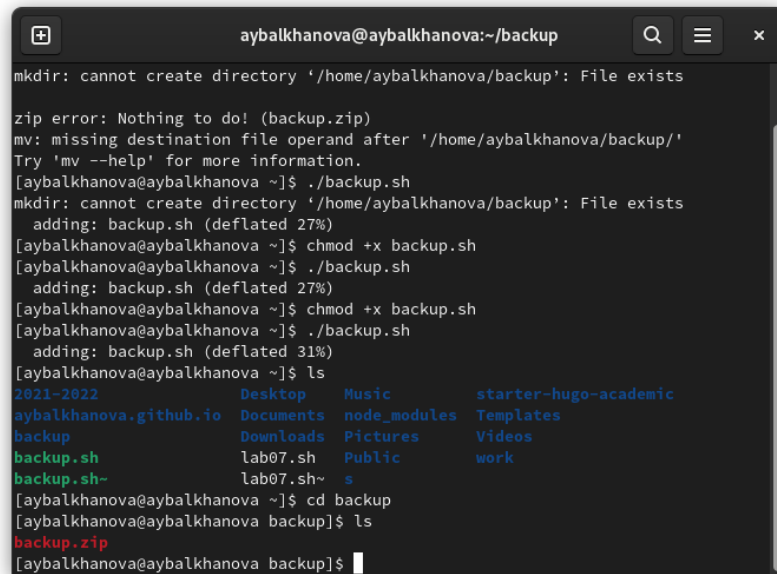


Рис. 0.5: Резервное копирование

4. Дала разрешение на выполнение с помощью команды `chmod +x backup.sh` и проверила работу скрипта, используя команду `./backup.sh` (рисю 0.6).

A terminal window titled 'aybalkhanova@aybalkhanova:~/backup' with search, menu, and close icons. The terminal shows the following commands and output:

```
mkdir: cannot create directory '/home/aybalkhanova/backup': File exists
zip error: Nothing to do! (backup.zip)
mv: missing destination file operand after '/home/aybalkhanova/backup/'
Try 'mv --help' for more information.
[aybalkhanova@aybalkhanova ~]$ ./backup.sh
mkdir: cannot create directory '/home/aybalkhanova/backup': File exists
  adding: backup.sh (deflated 27%)
[aybalkhanova@aybalkhanova ~]$ chmod +x backup.sh
[aybalkhanova@aybalkhanova ~]$ ./backup.sh
  adding: backup.sh (deflated 27%)
[aybalkhanova@aybalkhanova ~]$ chmod +x backup.sh
[aybalkhanova@aybalkhanova ~]$ ./backup.sh
  adding: backup.sh (deflated 31%)
[aybalkhanova@aybalkhanova ~]$ ls
2021-2022      Desktop      Music        starter-hugo-academic
aybalkhanova.github.io  Documents   node_modules Templates
backup        Downloads   Pictures     Videos
backup.sh     lab07.sh   Public       work
backup.sh~    lab07.sh~  s
[aybalkhanova@aybalkhanova ~]$ cd backup
[aybalkhanova@aybalkhanova backup]$ ls
backup.zip
[aybalkhanova@aybalkhanova backup]$
```

Рис. 0.6: Работа скрипта backup.sh

5. Написала командный файл, печатающий более 10 аргументов, передаваемых по командной строке (рис. 0.7).

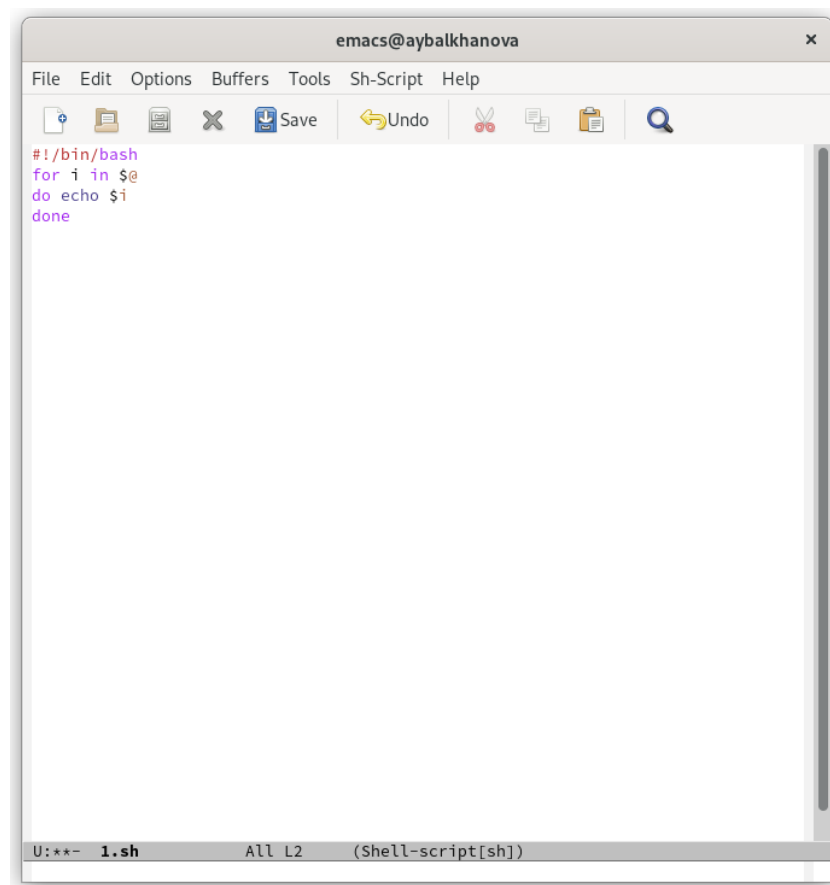
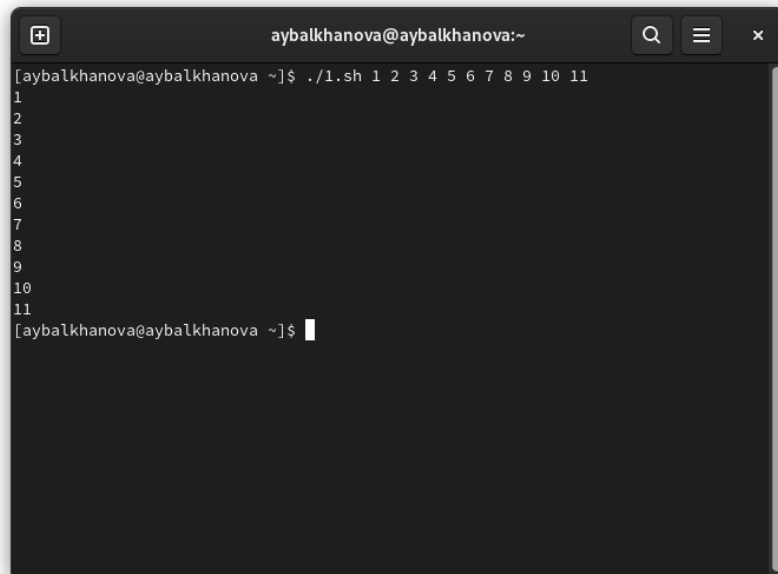


Рис. 0.7: Печать аргументов

6. Проверила его работу (рис. 0.8).

A terminal window with a dark background. The title bar shows 'aybalkhanova@aybalkhanova:~'. The prompt is '[aybalkhanova@aybalkhanova ~]\$'. The command './1.sh 1 2 3 4 5 6 7 8 9 10 11' has been entered. The output consists of the numbers 1 through 11, each on a new line. The prompt is now '[aybalkhanova@aybalkhanova ~]\$' with a cursor.

```
aybalkhanova@aybalkhanova:~  
[aybalkhanova@aybalkhanova ~]$ ./1.sh 1 2 3 4 5 6 7 8 9 10 11  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
[aybalkhanova@aybalkhanova ~]$
```

Рис. 0.8: Работа командного файла

7. Написаал командный файл - аналог команды `ls` (рис. 0.9).

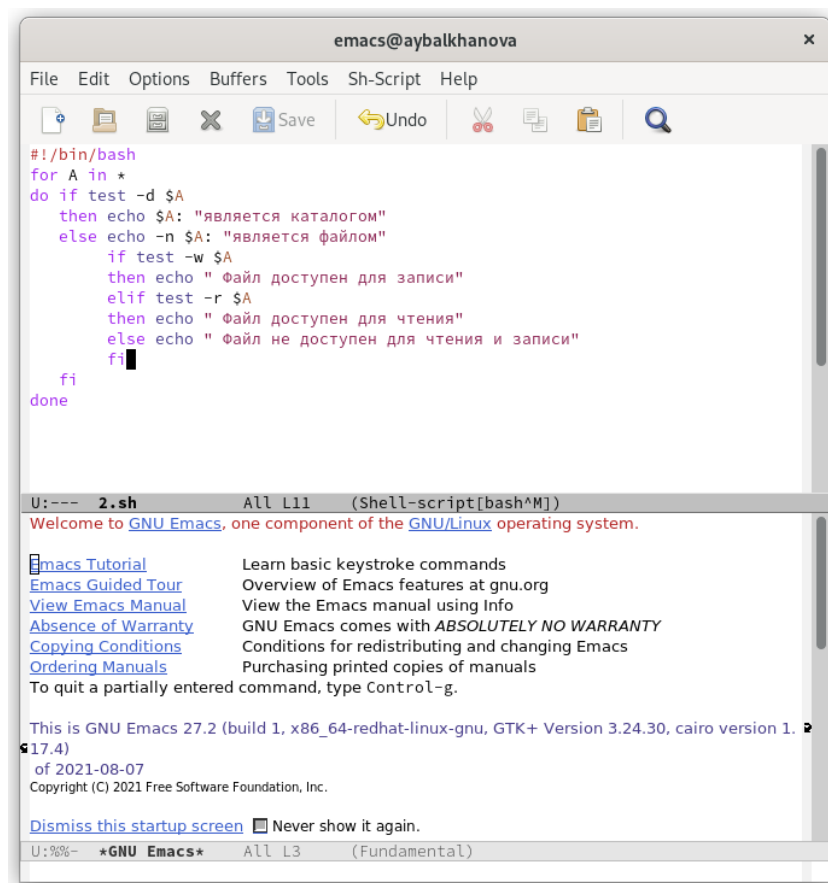
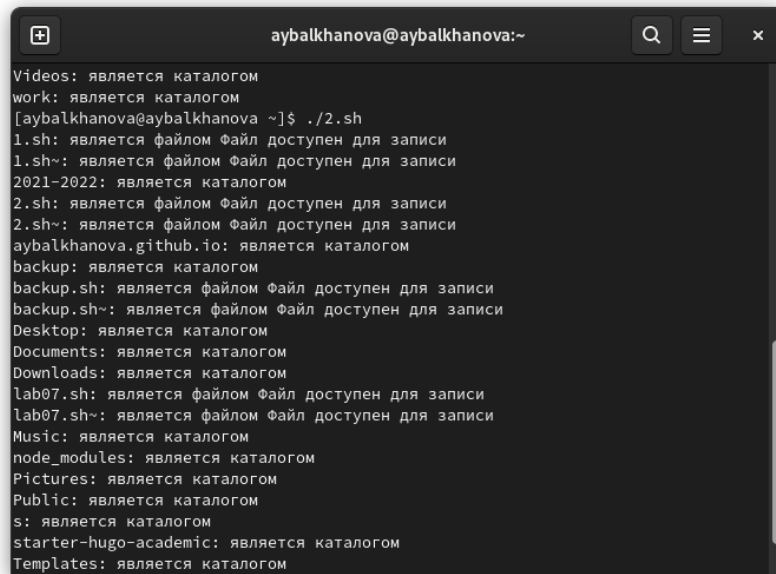


Рис. 0.9: ls

8. Проверила его работу ./2.sh (рис. 0.10).



```
aybalkhanova@aybalkhanova:~  
Videos: является каталогом  
work: является каталогом  
[aybalkhanova@aybalkhanova ~]$ ./2.sh  
1.sh: является файлом Файл доступен для записи  
1.sh~: является файлом Файл доступен для записи  
2021-2022: является каталогом  
2.sh: является файлом Файл доступен для записи  
2.sh~: является файлом Файл доступен для записи  
aybalkhanova.github.io: является каталогом  
backup: является каталогом  
backup.sh: является файлом Файл доступен для записи  
backup.sh~: является файлом Файл доступен для записи  
Desktop: является каталогом  
Documents: является каталогом  
Downloads: является каталогом  
lab07.sh: является файлом Файл доступен для записи  
lab07.sh~: является файлом Файл доступен для записи  
Music: является каталогом  
node_modules: является каталогом  
Pictures: является каталогом  
Public: является каталогом  
s: является каталогом  
starter-hugo-academic: является каталогом  
Templates: является каталогом
```

Рис. 0.10: Работа аналога ls

9. Написала командный файл, который считает количество файлов с заданным расширением в заданном каталоге (рис. 0.11).

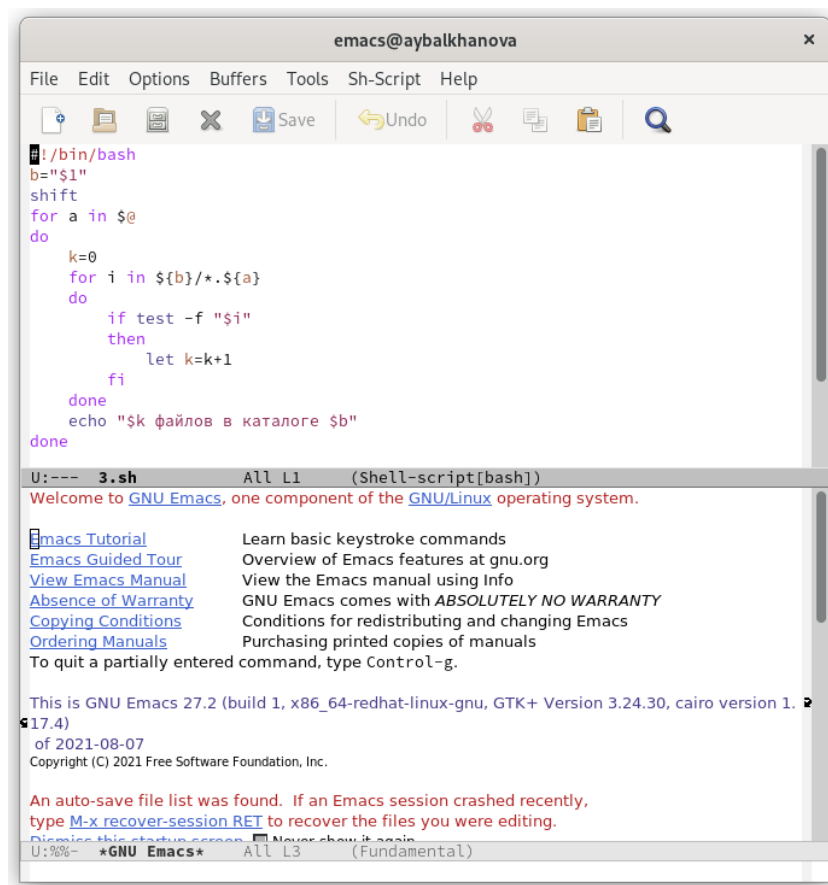
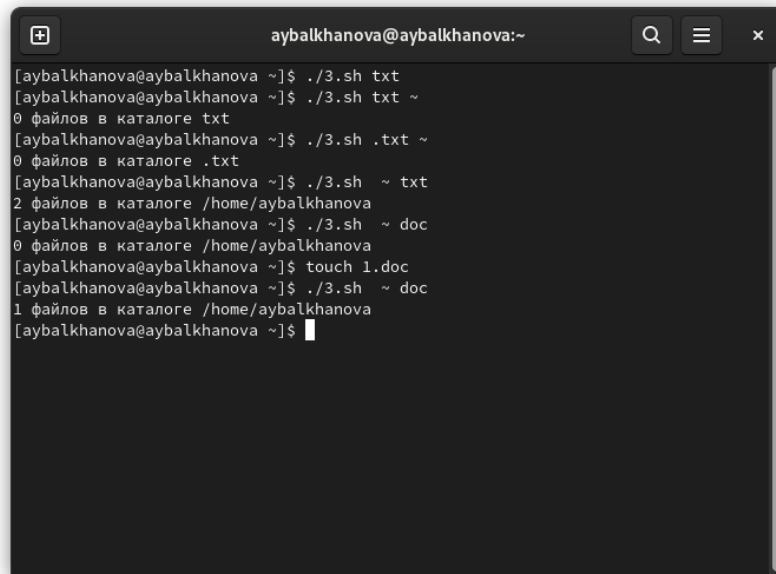


Рис. 0.11: Командный файл для подсчёта нужных файлов

10. Проверила его работу './3.sh ~.txt' и './3.sh ~.doc' (рис. 0.12).

A terminal window titled 'aybalkhanova@aybalkhanova:~' with standard window controls. The terminal displays the output of a script named '.3.sh' being run in different contexts. The output shows file counts in various directories and the successful creation of a file named '1.doc'.

```
[aybalkhanova@aybalkhanova ~]$ ./3.sh txt
[aybalkhanova@aybalkhanova ~]$ ./3.sh txt ~
0 файлов в каталоге txt
[aybalkhanova@aybalkhanova ~]$ ./3.sh .txt ~
0 файлов в каталоге .txt
[aybalkhanova@aybalkhanova ~]$ ./3.sh ~ txt
2 файлов в каталоге /home/aybalkhanova
[aybalkhanova@aybalkhanova ~]$ ./3.sh ~ doc
0 файлов в каталоге /home/aybalkhanova
[aybalkhanova@aybalkhanova ~]$ touch 1.doc
[aybalkhanova@aybalkhanova ~]$ ./3.sh ~ doc
1 файлов в каталоге /home/aybalkhanova
[aybalkhanova@aybalkhanova ~]$
```

Рис. 0.12: Работа



# Контрольные вопросы

1. Командная оболочка Unix — командный интерпретатор, используемый в операционных системах семейства Unix, в котором пользователь может либо давать команды операционной системе по отдельности, либо запускать скрипты, состоящие из списка команд.
2. POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ.
3. Например, команда `mark=/usr/andy/bin` присваивает значение строки символов `/usr/andy/bin` переменной `mark` типа строка символов. Для создания массива используется команда `set` с флагом `-A`.
4. Команда `let` берет два операнда и присваивает их переменной. Положительным моментом команды `let` можно считать то, что для идентификации переменной ей не нужен знак доллара; вы можете писать команды типа `let sum=x+7`, и `let` будет искать переменную `x` и добавлять к ней 7. Команда `let` также расширяет другие выражения `let`, если они заключены в двойные круглые скобки. Команда `read` позволяет читать значения переменных со стандартного ввода.
5. Можно выполнять как простые арифметические выражения, так и сложные выражения, например операторы `!`, `!=` возвращают 0 или 1, `%` возвращает остаток от деления и т.д.

6. Для облегчения программирования можно записывать условия оболочки `bash` в двойные скобки — `(( ))`.
7.
  - `PATH` - список каталогов, в которых командный процессор осуществляет поиск программы или команды, указанной в командной строке.
  - Переменные `PS1` и `PS2` предназначены для отображения промптера командного процессора.
  - `HOME` — имя домашнего каталога пользователя. Если команда `cd` вводится без аргументов, то происходит переход в каталог, указанный в этой переменной. — `IFS` — последовательность символов, являющихся разделителями в командной строке, например, пробел, табуляция и перевод строки (new line). — `MAIL` — командный процессор каждый раз перед выводом на экран промптера проверяет содержимое файла, имя которого указано в этой переменной, и если содержимое этого файла изменилось с момента последнего ввода из него, то перед тем как вывести на терминал промптер, командный процессор выводит на терминал сообщение `You have mail` (у Вас есть почта).
  - `TERM` — тип используемого терминала.
  - `LOGNAME` — содержит регистрационное имя пользователя, которое устанавливается автоматически при входе в систему.
8. Такие символы, как `' < > * ? | " &`, являются метасимволами и имеют для командного процессора специальный смысл. Снятие специального смысла с метасимвола называется экранированием метасимвола.
9. Экранирование может быть осуществлено с помощью предшествующего метасимволу символа `\`, который, в свою очередь, является метасимволом.
10. Нужно создать файл с типа `.sh`, чтобы запустить его нужно в командной строке написать название файла, а перед ним точку со слэшем.
11. Группу команд можно объединить в функцию. Для этого существует ключевое

слово `function`, после которого следует имя функции и список команд, заключённых в фигурные скобки.

12. – `test -f file` — истина, если файл `file` является обычным файлом.
  - `test -d file` — истина, если файл `file` является каталогом.
13.
  - Значение всех переменных можно просмотреть с помощью команды `set`.
  - Команда `typeset` имеет четыре опции для работы с функциями: – `-f` — перечисляет определённые на текущий момент функции; – `-ft` — при последующем вызове функции иницирует её трассировку; – `-fx` — экспортирует все перечисленные функции в любые дочерние программы оболочек; – `-fu` — обозначает указанные функции как автоматически загружаемые. Автоматически загружаемые функции хранятся в командных файлах, а при их вызове оболочка просматривает переменную `FPATH`, отыскивая файл с одноимёнными именами функций, загружает его и вызывает эти функции.
  - Удалить функцию можно с помощью команды `unset` с флагом `-f`.
14. При вызове командного файла на выполнение параметры ему могут быть переданы точно таким же образом, как и выполняемой программе.
15.
  - При вызове команды `clist` будет изменён каталог и выведено его содержимое.
  - Команда `shift` позволяет удалять первый параметр и сдвигает все остальные на места предыдущих. – `$*` — отображается вся командная строка или параметры оболочки; – `$?` — код завершения последней выполненной команды; – `$$` — уникальный идентификатор процесса, в рамках которого выполняется командный процессор; – `#!` — номер процесса, в рамках которого выполняется последняя вызванная на выполнение в командном режиме команда; – `$-` — значение флагов командного процессора; – `${#}` — возвращает целое число — количество

слов, которые были результатом  $\$$ ; —  $\#{\text{name}}$  — возвращает целое значение длины строки в переменной  $\text{name}$ ; —  $\{\text{name}[n]\}$  — обращение к  $n$ -му элементу массива; —  $\{\text{name}[*]\}$  — перечисляет все элементы массива, разделённые пробелом; —  $\{\text{name}[@]\}$  — то же самое, но позволяет учитывать символы пробелы в самих переменных; —  $\{\text{name}:-\text{value}\}$  — если значение переменной  $\text{name}$  не определено, то оно будет заменено на указанное  $\text{value}$ ;

- $\{\text{name}:\text{value}\}$  — проверяется факт существования переменной; —  $\{\text{name}=\text{value}\}$  — если  $\text{name}$  не определено, то ему присваивается значение  $\text{value}$ ; —  $\{\text{name}?\text{value}\}$  — останавливает выполнение, если имя переменной не определено, и выводит  $\text{value}$  как сообщение об ошибке; —  $\{\text{name}+\text{value}\}$  — это выражение работает противоположно  $\{\text{name}-\text{value}\}$ . Если переменная определена, то подставляется  $\text{value}$ ; —  $\{\text{name}\#\text{pattern}\}$  — представляет значение переменной  $\text{name}$  с удалённым самым коротким левым образцом ( $\text{pattern}$ ); —  $\#{\text{name}}[*]$  и  $\#{\text{name}}[@]$  — эти выражения возвращают количество элементов в массиве  $\text{name}$ .

## Выводы

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