

Лабораторная работа №10

Простейший вариант выполнения лабораторной работы

Атанесов Александр Николаевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	20
	Список литературы	23

Список иллюстраций

4.1	Использую команду <code>man</code>	8
4.2	Использую команду <code>terminal</code>	9
4.3	Использую команду <code>touch</code>	9
4.4	Использую команду <code>chmod</code>	10
4.5	Использую команду <code>nano</code>	10
4.6	Использую редактор <code>nano</code>	11
4.7	Использую команду <code>touch</code>	11
4.8	Использую команду <code>chmod</code>	12
4.9	Использую команду <code>nano</code>	12
4.10	Использую редактор <code>nano</code>	13
4.11	Использую команду <code>./</code>	13
4.12	Использую терминал	14
4.13	Использую команду <code>touch</code>	14
4.14	Открываю файл через <code>nano</code>	14
4.15	Использую команду <code>./</code>	15
4.16	Использую терминал	16
4.17	Использую команду <code>touch</code>	16
4.18	Использую команду <code>chmod</code>	17
4.19	Использую команду <code>nano</code>	17
4.20	Использую редактор <code>nano</code>	18
4.21	Использую команду <code>./</code>	18
4.22	Использую терминал	19

Список таблиц

3.1	Описание некоторых каталогов файловой системы GNU Linux . .	7
-----	---	---

1 Цель работы

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

2 Задание

1. Взаимодействовать с ОС через терминал посредством команд;

3 Теоретическое введение

Здесь описываются теоретические аспекты, связанные с выполнением работы. Например, в табл. [3.1] приведено краткое описание стандартных каталогов Unix.

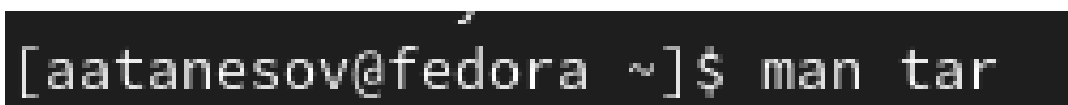
Таблица 3.1: Описание некоторых каталогов файловой системы GNU Linux

Имя каталога	Описание каталога
/	Корневая директория, содержащая всю файловую
/bin	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
/etc	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
/home	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
/media	Точки монтирования для сменных носителей
/root	Домашняя директория пользователя root
/tmp	Временные файлы
/usr	Вторичная иерархия для данных пользователя

Более подробно об Unix см. в [1–6].

4 Выполнение лабораторной работы

1. Ввожу команду `man tar`. (рис. [4.1])

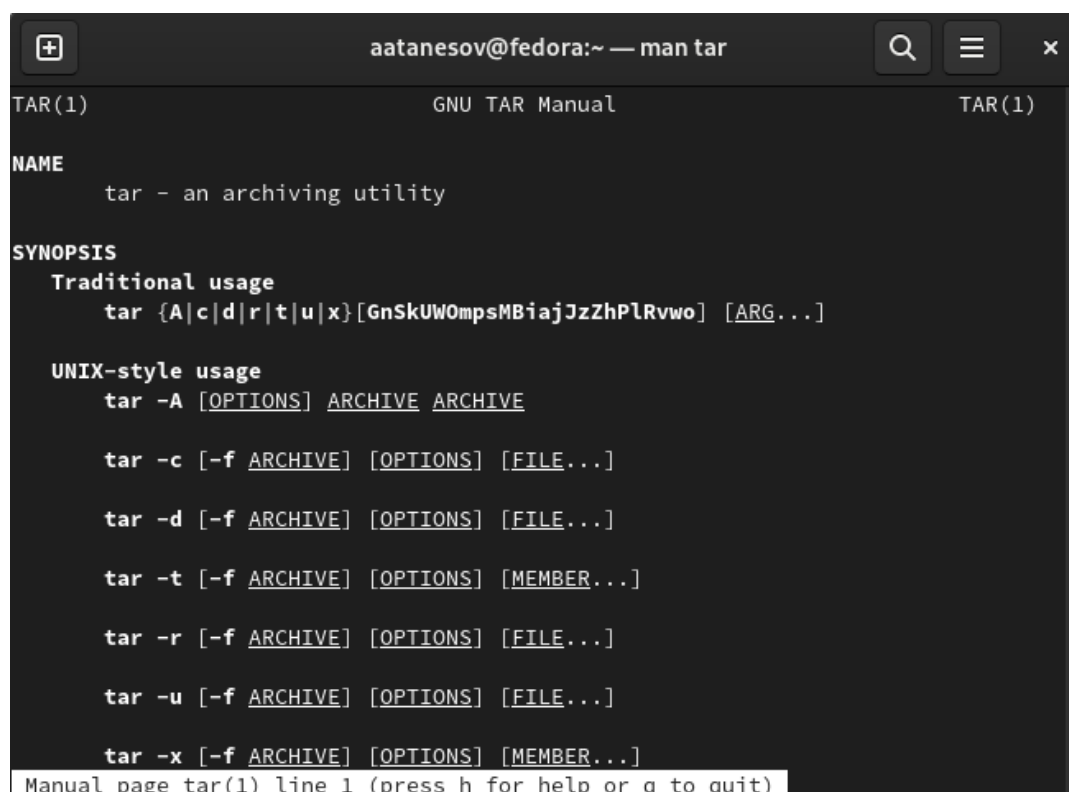
A terminal window with a dark background. The prompt is `[aatanesov@fedora ~]$` and the command `man tar` is entered.

```
[aatanesov@fedora ~]$ man tar
```

Рис. 4.1: Используя команду `man`

4.1

2. Изучаю информацию о команде `tar`. (рис. [4.2])

A terminal window titled 'aatanesov@fedora:~ — man tar'. The window displays the output of the 'man tar' command. The output includes the title 'TAR(1)', the subtitle 'GNU TAR Manual', and the title 'TAR(1)'. The main content is divided into sections: 'NAME' with the description 'tar - an archiving utility', 'SYNOPSIS', 'Traditional usage' with the command 'tar {A|c|d|r|t|u|x}[GnSkUWompsMBiajJzZhPlRvwo] [ARG...]', and 'UNIX-style usage' with several commands: '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...]', and 'tar -x [-f ARCHIVE] [OPTIONS] [MEMBER...]'. At the bottom, a status bar reads 'Manual page tar(1) line 1 (press h for help or q to quit)'.

```
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. 4.2: Использу команду terminal

4.2

3. Создаю файл script.sh. (рис. [4.3])

A terminal window showing the command 'touch script.sh' being executed. The prompt is '[aatanesov@fedora ~]\$'.

```
[aatanesov@fedora ~]$ touch script.sh
```

Рис. 4.3: Использу команду touch

4.3

4. Делаю файл script.sh исполняемым . (рис. [4.4])

```
[aatanesov@fedora ~]$ chmod +x script.sh
```

Рис. 4.4: Используя команду chmod

4.4

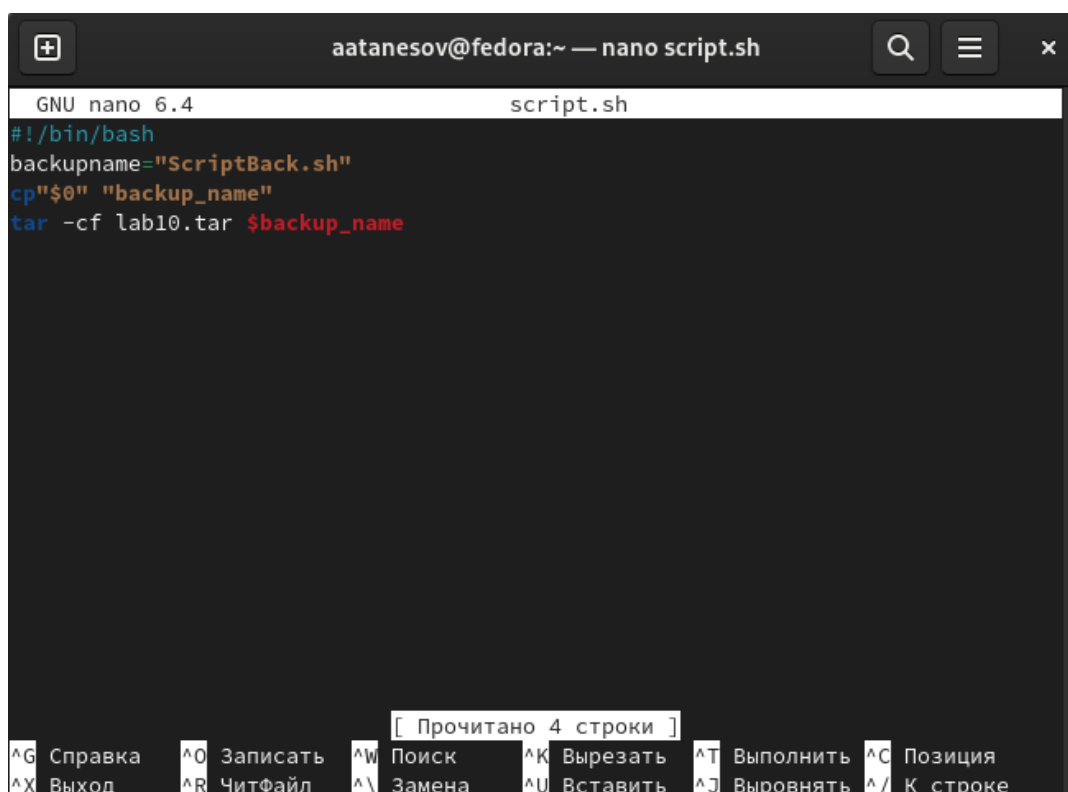
5. Открываю файл через nano. (рис. [4.5])

```
[aatanesov@fedora ~]$ nano script.sh
```

Рис. 4.5: Используя команду nano

4.5

6. Ввожу необходимый код ,для выполнения условия. (рис. [4.6])



```
GNU nano 6.4 script.sh
#!/bin/bash
backupname="ScriptBack.sh"
cp"$0" "backup_name"
tar -cf lab10.tar $backup_name
```

[Прочитано 4 строки]

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выровнять ^_/ К строке

Рис. 4.6: Используя редактор nano

4.6

7. Создаю файл script2.sh . (рис. [4.7])



```
[aatanesov@fedora ~]$ touch script2.sh
```

Рис. 4.7: Используя команду touch

4.7

8. Делаю файл исполняемым . (рис. [4.8])

```
[aatanesov@fedora ~]$ chmod +x script2.sh
```

Рис. 4.8: Используя команду chmod

4.8

9. Открываю файл script2.sh . (рис. [4.9])

```
[aatanesov@fedora ~]$ nano script2.sh
```

Рис. 4.9: Используя команду nano

4.9

10. Ввожу необходимый код. (рис. [4.10])

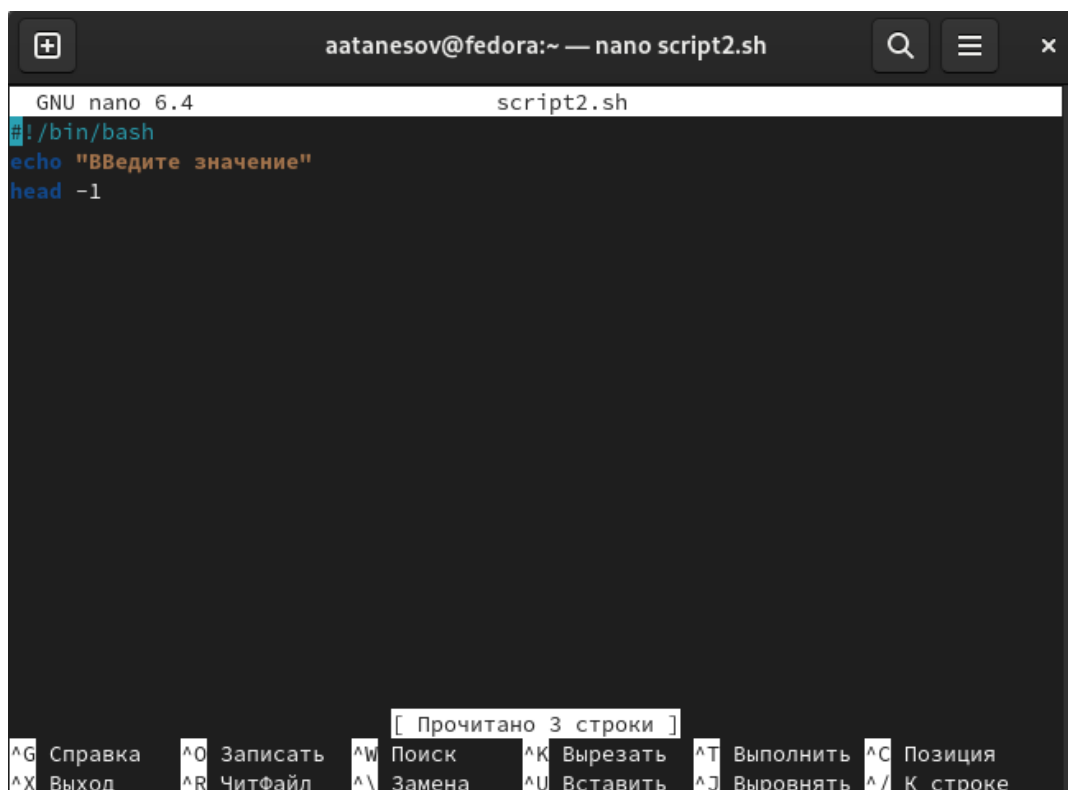


Рис. 4.10: Используя редактор nano

4.10

11. Выполняю файл script2.sh. (рис. [4.11])



Рис. 4.11: Используя команду ./

4.11

12. Смотрю на вывод. (рис. [4.12])

```
[aatanesov@fedora ~]$ ./script2.sh
Введите значение
1 2 3 4 5
1 2 3 4 5
```

Рис. 4.12: Используя терминал

4.12

13. Создаю файл file.sh. (рис. [4.13])

```
[aatanesov@fedora ~]$ touch file.sh
```

Рис. 4.13: Используя команду touch

4.13

14. Делаю файл исполняемым. (рис. [??])

```
[aatanesov@fedora ~]$ chmod +x file.sh
```

(image/13.png){#fig:014

width=90%}

4.14

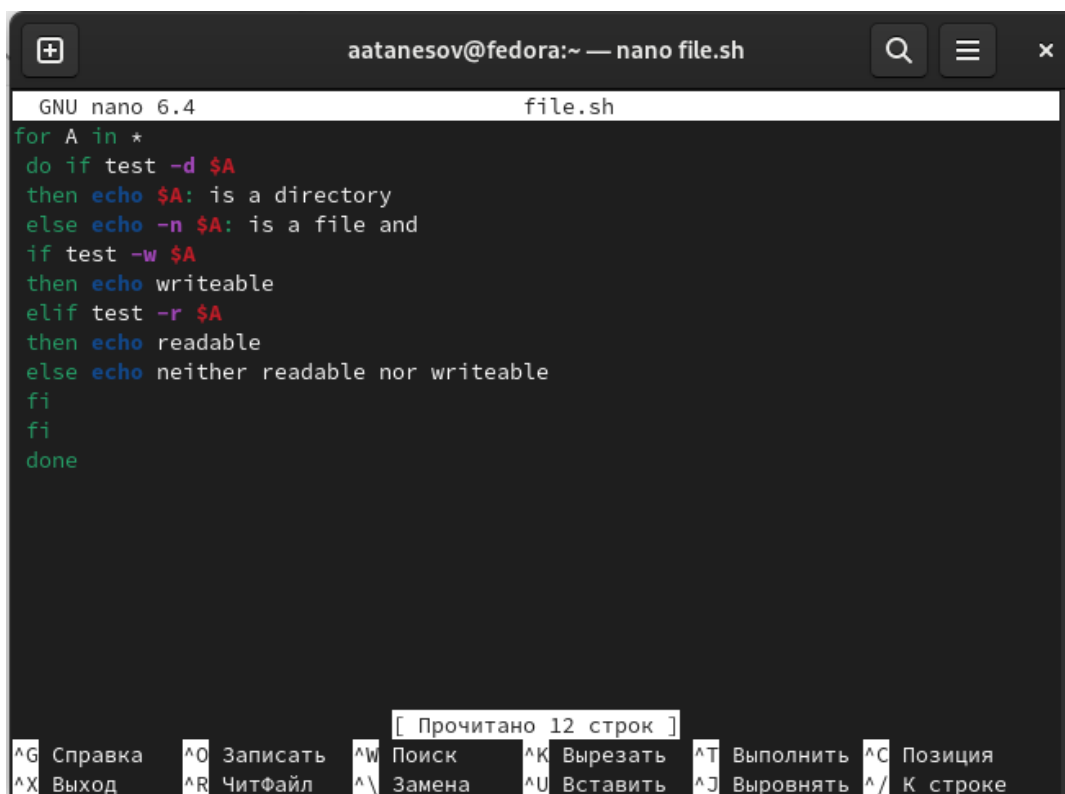
15. Открываю файл через nano. (рис. [4.14])

```
[aatanesov@fedora ~]$ nano file.sh
```

Рис. 4.14: Открываю файл через nano

4.15

16. Ввожу необходимый код. (рис. [??])



```
GNU nano 6.4 file.sh
for A in *
do if test -d $A
then echo $A: is a directory
else echo -n $A: is a file and
if test -w $A
then echo writeable
elif test -r $A
then echo readable
else echo neither readable nor writeable
fi
fi
done
```

[Прочитано 12 строк]


^G Справка	^O Записать	^W Поиск	^K Вырезать	^T Выполнить	^C Позиция
^X Выход	^R ЧитФайл	^_\ Замена	^U Вставить	^J Выводить	^/ К строке

(image/18.png){#fig:

width=90%}

4.16

17. Воспроизвожу файл file.sh. (рис. [4.15])

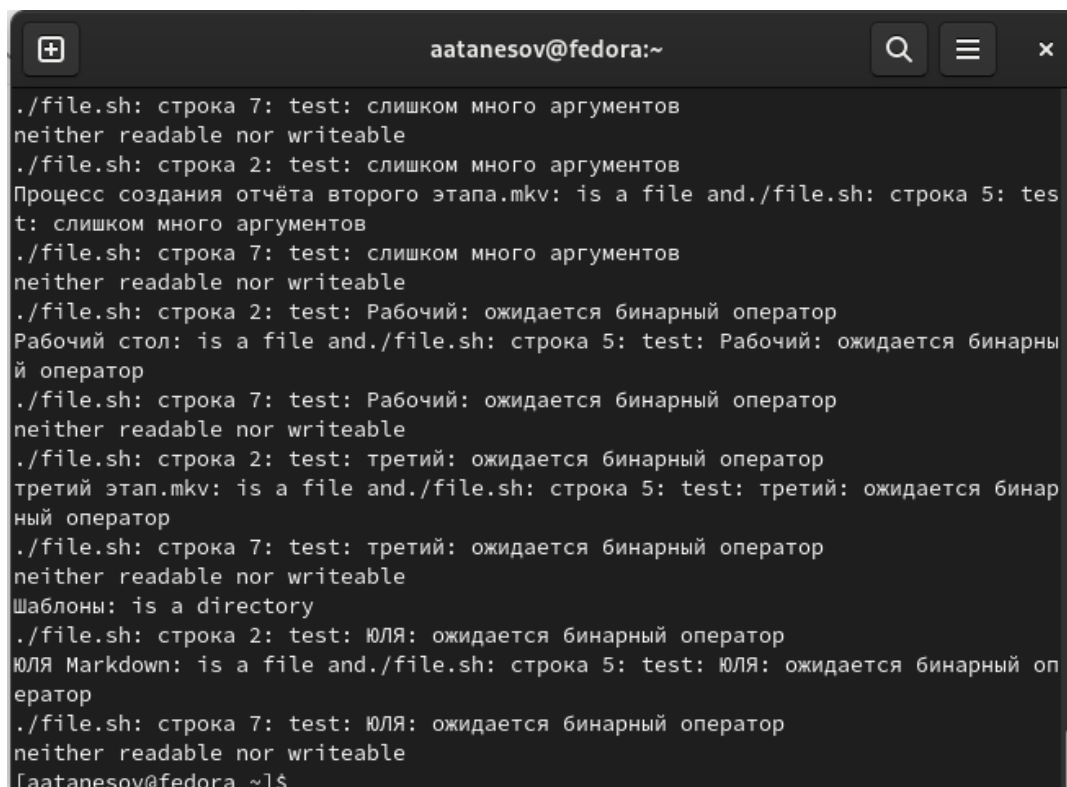


```
[aatanesov@fedora ~]$ ./file.sh
```

Рис. 4.15: Используя команду ./

4.17

18. Смотрю на результат. (рис. [4.16])



```
./file.sh: строка 7: test: слишком много аргументов
neither readable nor writeable
./file.sh: строка 2: test: слишком много аргументов
Процесс создания отчёта второго этапа.mkv: is a file and./file.sh: строка 5: tes
t: слишком много аргументов
./file.sh: строка 7: test: слишком много аргументов
neither readable nor writeable
./file.sh: строка 2: test: Рабочий: ожидается бинарный оператор
Рабочий стол: is a file and./file.sh: строка 5: test: Рабочий: ожидается бинарн
ый оператор
./file.sh: строка 7: test: Рабочий: ожидается бинарный оператор
neither readable nor writeable
./file.sh: строка 2: test: третий: ожидается бинарный оператор
третий этап.mkv: is a file and./file.sh: строка 5: test: третий: ожидается бинар
ный оператор
./file.sh: строка 7: test: третий: ожидается бинарный оператор
neither readable nor writeable
Шаблоны: is a directory
./file.sh: строка 2: test: ЮЛЯ: ожидается бинарный оператор
ЮЛЯ Markdown: is a file and./file.sh: строка 5: test: ЮЛЯ: ожидается бинарный оп
ератор
./file.sh: строка 7: test: ЮЛЯ: ожидается бинарный оператор
neither readable nor writeable
[aatanesov@fedora ~]$
```

Рис. 4.16: Использу терминал

4.18

19. Создаю файл file2.sh. (рис. [4.17])




```
[aatanesov@fedora ~]$ touch file2.sh
```

Рис. 4.17: Использую команду touch

4.19

20. Делаю файл исполняемым. (рис. [4.18])



```
[aatanesov@fedora ~]$ chmod +x file2.sh
```

Рис. 4.18: Использую команду chmod

4.20

21. Запускаю файл через nano. (рис. [4.19])

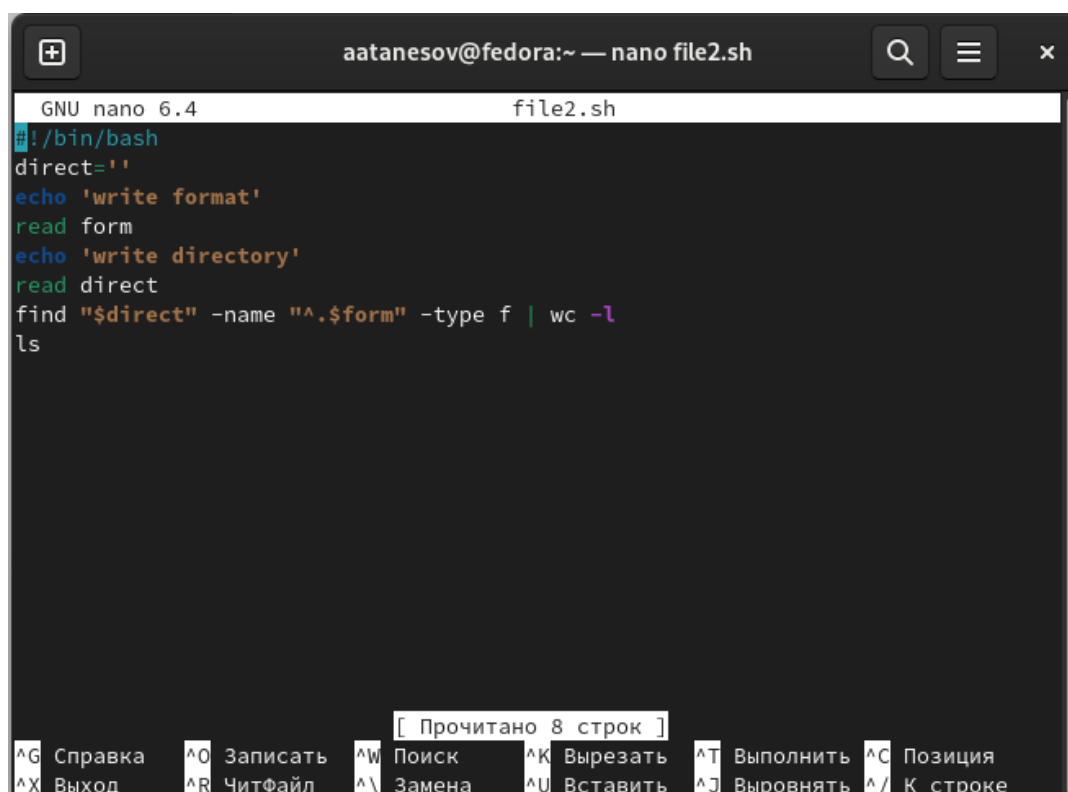


```
[aatanesov@fedora ~]$ nano file2.sh
```

Рис. 4.19: Использую команду nano

4.21

22. Ввожу необходимый код. (рис. [4.20])



```
GNU nano 6.4 file2.sh
#!/bin/bash
direct=''
echo 'write format'
read form
echo 'write directory'
read direct
find "$direct" -name "^.$form" -type f | wc -l
ls
```

[Прочитано 8 строк]

^G Справка	^O Записать	^W Поиск	^K Вырезать	^T Выполнить	^C Позиция
^X Выход	^R ЧитФайл	^_ Замена	^U Вставить	^J Выводить	^_ К строке

Рис. 4.20: Используя редактор nano

4.22

23. Воспроизвожу файл file2.sh . (рис. [4.21])



```
[aatanesov@fedora ~]$ ./file2.sh
```

Рис. 4.21: Используя команду ./

4.23

24. Смотрю на результат . (рис. [4.22])

```
[aatanesov@fedora ~]$ ./file2.sh
write format
txt
write directory
~
find: '~': Нет такого файла или каталога
0
aatanesov
abc1
confederation
conf.txt
Fedora-i3-Live-x86_64-37-1.7.iso.uibak
file2.sh
file.sh
lab01
lab05.mkv
lab06.mkv
lab07.mkv
lab08.mkv
lab10.mkv
may
monthly
newdir
```

Рис. 4.22: Используя терминал

5 Выводы

- Познал основы программирования в UNIX. # Ответы на контрольные вопросы
- 1. Командная оболочка - это программа, которая обеспечивает пользовательский интерфейс для взаимодействия с операционной системой. Она позволяет пользователю вводить команды и запускать программы, а также выполняет подстановку значений переменных и метасимволов. Примеры командных оболочек: bash, zsh, csh, tcsh. Они отличаются синтаксисом и набором доступных функций.
- 2. POSIX (Portable Operating System Interface) - это стандарт, описывающий интерфейс между операционной системой и приложением. Он был разработан для обеспечения переносимости приложений между разными операционными системами и включает в себя спецификации для системных вызовов, библиотек, командных интерпретаторов и других компонентов операционной системы.
- 3. Переменные в языке программирования bash определяются путем присваивания значения имени переменной. Например: `var="hello"`. Массивы определяются с использованием квадратных скобок: `array=(1 2 3)`.
- 4. Оператор `let` используется для выполнения арифметических операций. Пример: `let "var = 2 + 2"`. Оператор `read` используется для чтения данных из пользовательского ввода.

- 5. В языке программирования `bash` можно выполнять арифметические операции сложения, вычитания, умножения, деления, взятия остатка, а также использовать скобки для определения порядка операций.
- 6. Операция `(())` используется для выполнения арифметических операций, а также для сравнения чисел. Например: `((var1 > var2))`. Она возвращает результат в виде 0 (если условие ложно) или 1 (если условие истинно).
- 7. Стандартными именами переменных являются: `HOME`, `USER`, `PATH`, `SHELL`, `TERM` и другие. Они определяют различные настройки и параметры системы.
- 8. Метасимволы - это символы, которые используются для обозначения шаблонов и задания нескольких символов в одной команде. Например: `*`, `?`, `[]`, `{}`.
- 9. Метасимволы можно экранировать, добавив перед ними символ обратного слеша `()`. Например: `*`.
- 10. Командные файлы создаются с помощью текстового редактора и сохраняются с расширением `.sh`. Для запуска командного файла необходимо установить ему права на выполнение с помощью команды `chmod +x filename.sh`, а затем выполнить его с помощью команды `./filename.sh`.
- 11. Функции в языке программирования `bash` определяются с помощью ключевого слова `function` и блока кода, который должен быть выполнен, когда функция вызывается. Например: `function hello { echo "Hello, world!"; }`
- 12. Для определения типа файла используется команда `file`. Например: `file myfile.txt`. В ответ будет указан тип файла (текстовый, исполняемый и т.д.).

- 13. Команда `set` используется для установки настроек оболочки, `typeset` - для определения типа переменных и их свойств, а `unset` - для удаления переменных.
- 14. В командные файлы параметры передаются через аргументы командной строки. Например: `./script.sh arg1 arg2`.
- 15. Специальные переменные языка `bash`: `$0` - имя скрипта, `$1`, `$2` и т.д. - аргументы командной строки, `$#` - количество переданных параметров, `$*` и `$@` - все аргументы командной строки, `$$` - идентификатор текущего процесса, `$?` - статус завершения последней выполненной команды.

Список литературы

1. GNU Bash Manual [Электронный ресурс]. Free Software Foundation, 2016.
URL: <https://www.gnu.org/software/bash/manual/>.
2. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 с.
5. Таненбаум Э. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 874 с.
6. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.