

Лабораторная работа №12

Простой способ выполнения лабораторной работы №12

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	22

Список иллюстраций

4.1	Использую команду touch	8
4.2	Использую команду chmod	8
4.3	Использую команду touch	9
4.4	Использую команду touch	9
4.5	Использую команду nano	9
4.6	Использую команду nano	9
4.7	Использую редактор nano	11
4.8	Использую команду ./	12
4.9	Использую команду touch	12
4.10	Использую команду chmod	13
4.11	Использую команду nano	13
4.12	Использую редактор nano	13
4.13	Использую команду ./	14
4.14	Открываю файл через ./	15
4.15	Использую команду ./	16
4.16	Использую команду chmod	17
4.17	Использую команду nano	17
4.18	Использую редактор nano	18
4.19	Использую команду ./	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. Создаю файл `command.sh`. (рис. [4.1])



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

Рис. 4.1: Использую команду `touch`

4.1

2. Делаю файл исполняемым. (рис. [4.2])



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

Рис. 4.2: Использую команду `chmod`

4.2

3. Создаю файл `semaphore.lock`. (рис. [4.3])


```
[aatanesov@fedora ~]$ touch semaphore.lock
```

Рис. 4.3: Использу команду touch

4.3

4. Создаю файл output.txt для автоматической записи изменений . (рис. [4.4])

```
[aatanesov@fedora ~]$ touch output.txt
```

Рис. 4.4: Использу команду touch

4.4

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

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

Рис. 4.5: Использу команду nano

4.5

6. Открываю файл command.sh через nano. (рис. [4.6])

```
[aatanesov@fedora lab12]$ nano command.sh
```

Рис. 4.6: Использу команду nano

4.6

7. Пишу необходимый код для выполнения условий задачи 1 . (рис. [4.7])

```

#!/bin/bash

# Проверяем наличие аргументов
if [ $# -lt 3 ] || [ $# -gt 4 ]; then
    echo "Usage: $0 <semaphore_name> <t1> <t2> [<output_device>]"
    exit 1
fi

# Задаем переменные
semaphore_name="$1"
t1="$2"
t2="$3"
semaphore_file="/tmp/$semaphore.lock"

# Создаем семафор
if [ ! -e "${semaphore_file}" ]; then
    touch "${semaphore_file}"
    echo "Semaphore '${semaphore_name}' created!"
fi

# Захватываем семафор
while [ -e "${semaphore_file}" ]; do
    sleep "${t1}"
    echo "Waiting for semaphore '${semaphore_name}' to be released..."
done
touch "${semaphore_file}"
echo "Semaphore '${semaphore_name}' captured by process $$!"

# Используем ресурс
sleep "${t2}"
echo "Resource used for ${t2} seconds by process $$!"

# Освобождаем семафор
rm "${semaphore_file}"
echo "Semaphore '${semaphore_name}' released by process $$!"

# Проверяем наличие аргумента для перенаправления вывода
if [ $# -eq 4 ] && [ -c "/dev/${4}" ]; then
    exec > "/dev/${4}" 2>&1
fi

# Запускаем процесс в фоновом режиме
while true; do
    if ! "./$0" "${semaphore_name}" "${t1}" "${t2}" "${4}" &>/dev/null; then
        echo "Failed to start background process!"
        exit 1
    fi
done

```

Рис. 4.7: Используя редактор nano

4.7

8. Запускаю файл `command.sh` с `semaphore` . (рис. [4.8])

```
[aatanesov@fedora lab12]$ sudo bash ./command.sh semaphore 1 5
[sudo] пароль для aatanesov:
Попробуйте ещё раз.
[sudo] пароль для aatanesov:
Semaphore 'semaphore' created!
Waiting for semaphore 'semaphore' to be released...
Waiting for semaphore 'semaphore' to be released...
Waiting for semaphore 'semaphore' to be released...
Waiting for semaphore 'semaphore' to be released...
Waiting for semaphore 'semaphore' to be released...
Waiting for semaphore 'semaphore' to be released...
Waiting for semaphore 'semaphore' to be released...
Waiting for semaphore 'semaphore' to be released...
Waiting for semaphore 'semaphore' to be released...
Waiting for semaphore 'semaphore' to be released...
Waiting for semaphore 'semaphore' to be released...
Waiting for semaphore 'semaphore' to be released...
Waiting for semaphore 'semaphore' to be released...
```

Рис. 4.8: Используя команду `./`

4.8

9. Создаю файл `man.sh` . (рис. [4.9])

```
[aatanesov@fedora lab12]$ touch man.sh
```

Рис. 4.9: Используя команду `touch`

4.9

10. Делаю файл исполняемым. (рис. [4.10])

```
[aatanesov@fedora lab12]$ chmod +x man.sh
```

Рис. 4.10: Используя команду chmod

4.10

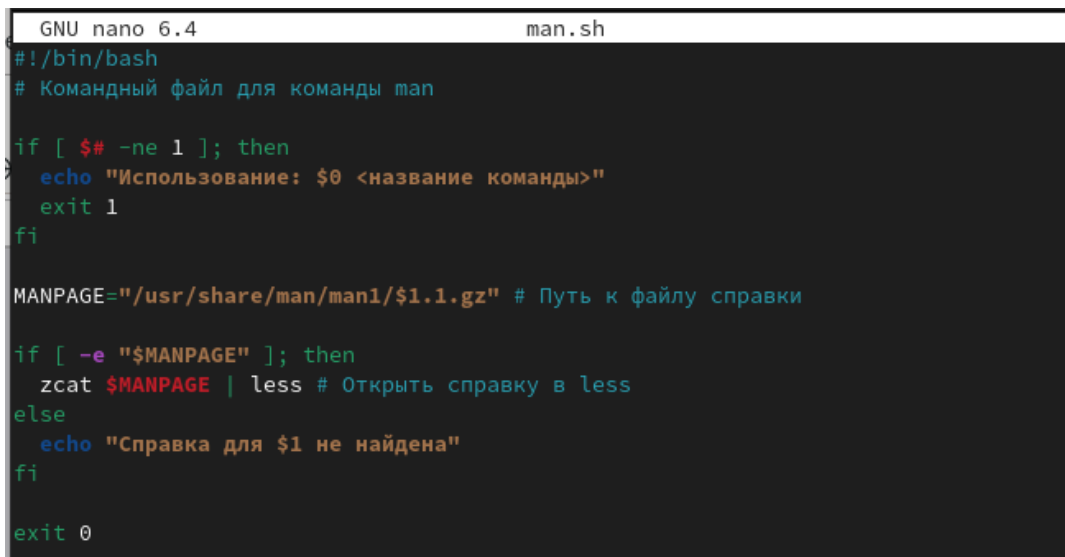
11. Открываю файл man.sh через nano. (рис. [4.11])

```
[aatanesov@fedora lab12]$ nano man.sh
```

Рис. 4.11: Используя команду nano

4.11

12. Пишу необходимый код. (рис. [4.12])



```
GNU nano 6.4 man.sh
#!/bin/bash
# Командный файл для команды man

if [ $# -ne 1 ]; then
    echo "Использование: $0 <название команды>"
    exit 1
fi

MANPAGE="/usr/share/man/man1/$1.1.gz" # Путь к файлу справки

if [ -e "$MANPAGE" ]; then
    zcat $MANPAGE | less # Открыть справку в less
else
    echo "Справка для $1 не найдена"
fi

exit 0
```

Рис. 4.12: Используя редактор nano

4.12

13. Запускаю файл `man.sh` с выводом справки для команды `ls`. (рис. [4.13])

```
[aatanesov@fedora lab12]$ ./man.sh ls
```

Рис. 4.13: Используя команду `./`

4.13

14. Вижу вывод на команду `ls`. (рис. [??])

```
\" DO NOT MODIFY THIS FILE! It was generated by help2man 1.48.5.
TH LS "1" "January 2023" "GNU coreutils 9.1" "User Commands"
SH NAME
ls \- list directory contents
SH SYNOPSIS
B ls
[\fI\,OPTION\|\fR]... [\fI\,FILE\|\fR]...
SH DESCRIPTION
\" Add any additional description here
PP
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of \fB\--cftuvSUX\fR nor \fB\--sort\fR is sp
ecified.
PP
Mandatory arguments to long options are mandatory for short options too.
TP
\fB\--a\fR, \fB\--all\fR
do not ignore entries starting with .
TP
\fB\--A\fR, \fB\--almost-all\fR
do not list implied . and ..
TP
\fB\--author\fR
with \fB\--l\fR, print the author of each file
TP
\fB\--b\fR, \fB\--escape\fR
print C-style escapes for nongraphic characters
TP
\fB\--block-size\fR=SIZE
with \fB\--l\fR, scale sizes by SIZE when printing them;
e.g., '\fB\--block-size=M'; see SIZE format below
TP
```

(image/13.p

width=90%}

4.14

15. Запускаю файл `man.sh` с выводом справки для команды `rm`. (рис. [4.14])



```
[aatanesov@fedora lab12]$ ./man.sh rm
```

Рис. 4.14: Открываю файл через ./

4.15

16. Вижу вывод на команду `rm`. (рис. [??])

```

.\" DO NOT MODIFY THIS FILE! It was generated by help2man 1.48.5.
.TH RM "1" "January 2023" "GNU coreutils 9.1" "User Commands"
.SH NAME
rm \- remove files or directories
.SH SYNOPSIS
.B rm
[\fI\,OPTION\/\fR]... [\fI\,FILE\/\fR]...
.SH DESCRIPTION
This manual page
documents the GNU version of
.BR rm .
.B rm
removes each specified file. By default, it does not remove
directories.
.P
If the \fI\-I\fR or \fI\--interactive=once\fR option is given,
and there are more than three files or the \fI\-r\fR, \fI\-R\fR,
or \fI\--recursive\fR are given, then
.B rm
prompts the user for whether to proceed with the entire operation. If
the response is not affirmative, the entire command is aborted.
.P
Otherwise, if a file is unwritable, standard input is a terminal, and
the \fI\-f\fR or \fI\--force\fR option is not given, or the
\fI\-i\fR or \fI\--interactive=always\fR option is given,
.B rm
prompts the user for whether to remove the file. If the response is
not affirmative, the file is skipped.
.SH OPTIONS
.PP
Remove (unlink) the FILE(s).
.TP
:
```

(image/18.png){#fig:

width=90%}

4.16

17. Создаю файл `latyn.sh` для выполнения третьего этапа. (рис. [4.15])

```
[aatanesov@fedora lab12]$ touch latyn.sh
```

Рис. 4.15: Используя команду `./`

4.17

18. Делаю файл исполняемым . (рис. [4.16])



```
[aatanesov@fedora lab12]$ chmod +x latyn.sh
```

Рис. 4.16: Использую команду chmod

4.18

19. Открываю файл latyn.sh через nano. (рис. [4.17])



```
[aatanesov@fedora lab12]$ nano latyn.sh
```

Рис. 4.17: Использую команду nano

4.19

20. Ввожу необходимый код. (рис. [4.18])

```
#!/bin/bash

for i in {1..10} # генерируем 10 букв
do
    rand_num=$((RANDOM % 26)) # генерируем случайное число от 0 до 25
    letter=$(echo {A..Z} | cut -d ' ' -f $((rand_num + 1))) # находим букву по н>
    echo -n $letter # выводим букву без перевода строки
done
echo # перевод строки после последней буквы
```

[Прочитано 9 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. 4.18: Используя редактор nano

4.20

21. Запускаю файл latyn.sh . (рис. [4.19])

```
[aatanesov@fedora lab12]$ ./latyn.sh
URKHUTHRYZ
[aatanesov@fedora lab12]$ ./latyn.sh
NZRTDZUZDV
[aatanesov@fedora lab12]$ ./latyn.sh
JZQRUDXEEI
```

Рис. 4.19: Используя команду ./

5 Выводы

- Познал основы программирования в UNIX. # Ответы на контрольные вопросы
- 1. Отсутствует пробел между скобками и оператором условия. Правильно: `while [$1 != "exit"]`.
- 2. Можно использовать оператор конкатенации - символ "+" или переменную, содержащую объединяемые строки.
- 3. Утилита `seq` используется для генерации числовых последовательностей. Её функционал можно реализовать с помощью циклов с оператором перебора и арифметических операций.
- 4. Результатом вычисления будет число 3.3333333, но в `bash` результат целочисленного деления (оператор `//`) будет без округления, т.е. равным 3.
- 5. `zsh` имеет более широкие возможности для настройки и расширения, например, более продвинутое автодополнение команд и параметров. Она также поддерживает более мощный синтаксис и некоторые удобные функции, например, поддержку массивов. Однако `bash` является более распространенным и стабильным в использовании.
- 6. Синтаксис верен, но значение переменной `LIMIT` необходимо задать заранее.

- 7. Bash относится к скриптовым языкам программирования. Он отличается от императивных языков (например, C++, Java) тем, что команды выполняются последовательно, без явного объявления переменных и типов данных. Преимуществами bash является его простота в использовании, поддержка большинства UNIX-систем и гибкость в написании скриптов. Однако он может иметь низкую производительность при обработке больших объемов данных.

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.