

Лабораторная работа №12

**Программирование в командном процессоре ОС UNIX. Командные
файлы**

Мухин Тимофей Владимирович

Содержание

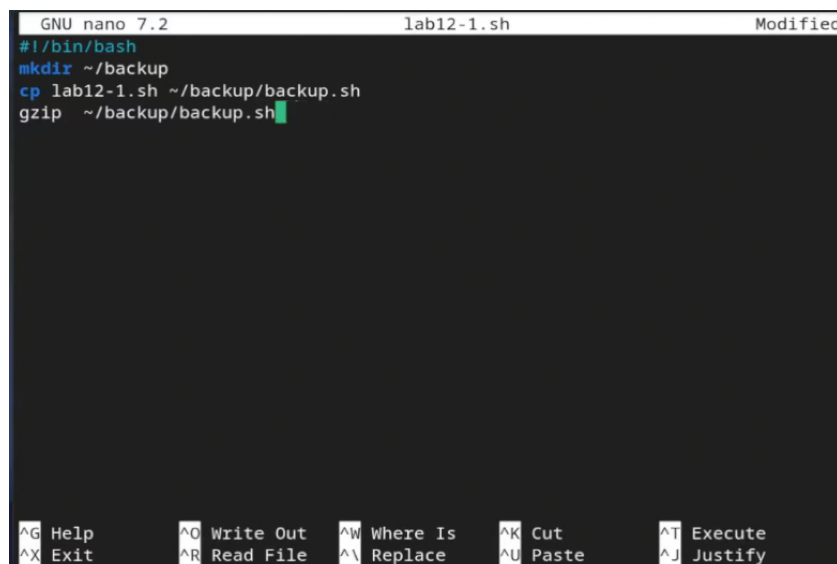
1	Цель работы	3
2	Выполнение лабораторной работы	4
3	Контрольные вопросы	9
4	Выводы	11

1 Цель работы

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

2 Выполнение лабораторной работы

1. Напишем 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



```
GNU nano 7.2 lab12-1.sh Modified
#!/bin/bash
mkdir ~/backup
cp lab12-1.sh ~/backup/backup.sh
gzip ~/backup/backup.sh
```

Рис. 2.1: Скрипт lab12-1

2. Запустим

```
tvmukhin@fedora:~$ chmod +x lab12-1.sh
tvmukhin@fedora:~$ ./lab12-1.sh
mkdir: cannot create directory '/home/sey_chik/backup': File exists
tvmukhin@fedora:~$ ./lab12-1.sh
tvmukhin@fedora:~$ ls
backup  Documents  lab12-1.sh  Pictures  Templates  work
Desktop Downloads Music      Public    Videos
tvmukhin@fedora:~$ cd backup/
tvmukhin@fedora:~/backup$ ls
backup.sh.gz
tvmukhin@fedora:~/backup$
```

Рис. 2.2: Запуск скрипта

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

```
GNU nano 7.2 lab12-2.sh
#!/bin/bash
for arg in "$@"
do
    echo "$arg"
done

[ Read 6 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute
^X Exit      ^R Read File  ^_ Replace    ^U Paste      ^J Justify
```

Рис. 2.3: Скрипт lab12-2

4. Запустим

```

tvmukhin@fedora:~$ chmod +x lab12-2.sh
tvmukhin@fedora:~$ ./lab12-2.sh
tvmukhin@fedora:~$ ./lab12-2.sh 1 2 3 4 5 6 7 8
1
2
3
4
5
6
7
8
tvmukhin@fedora:~$

```

Рис. 2.4: Запуск скрипта

5. Напишем командный файл — аналог команды `ls` (без использования самой этой ко- манды и команды `dir`). Требуется, чтобы он выдавал информацию о нужном каталоге и выводил информацию о возможностях доступа к фай- лам этого каталога.

```

GNU nano 7.2 lab12-3.sh
#!/bin/bash
echo "Read permission"
find $1 -maxdepth 1 -perm /u=r
echo "Write permission"
find $1 -maxdepth 1 -perm /u=w
echo "Execute permission"
find $1 -maxdepth 1 -perm /u=x

[ Wrote 9 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^I Execute
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify

```

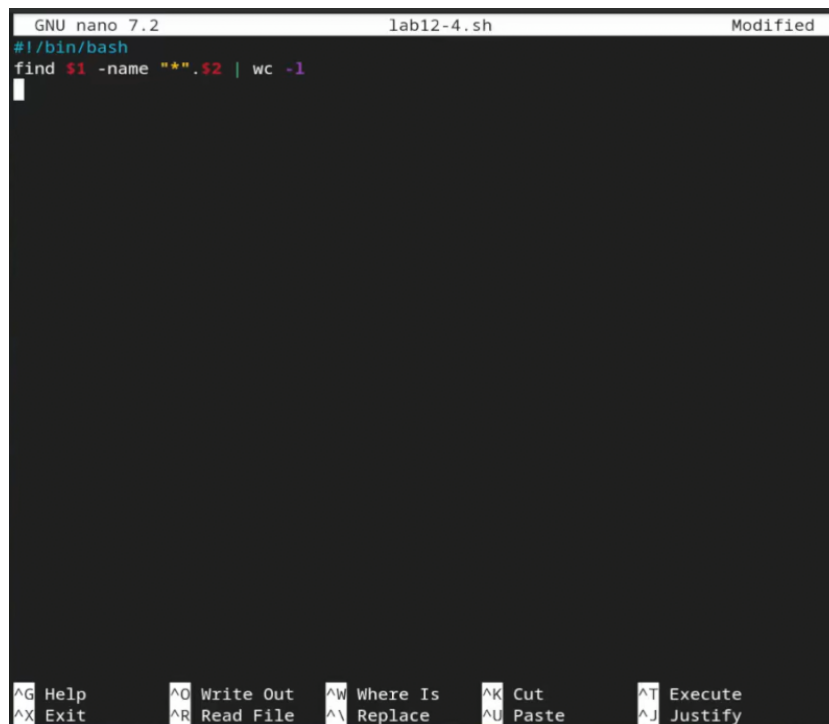
Рис. 2.5: Скрипт lab12-3)

6. Запустим

```
./steam
./steampath
./steampid
./emacs.d
./lab12-1.sh
./backup
./lab12-2.sh
./lab12-3.sh
Execute permission
.
./mozilla
./cache
./config
./local
./Desktop
./Downloads
./Templates
./Public
./Documents
./Music
./Pictures
./Videos
./var
./pki
./gnupg
./ssh
```

Рис. 2.6: Запуск скрипта

7. Напишем 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

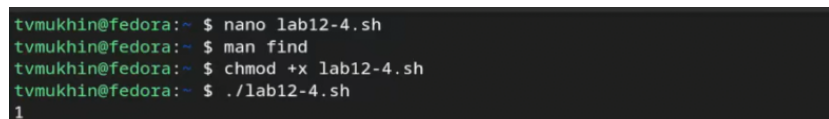


```
GNU nano 7.2 lab12-4.sh Modified
#!/bin/bash
find $1 -name "*.s2" | wc -l

```

Рис. 2.7: Скрипт lab12-4

8. Запустим



```
tvmukhin@fedora:~$ nano lab12-4.sh
tvmukhin@fedora:~$ man find
tvmukhin@fedora:~$ chmod +x lab12-4.sh
tvmukhin@fedora:~$ ./lab12-4.sh
1

```

Рис. 2.8: Запуск скрипта

3 Контрольные вопросы

1. **Командная оболочка** - это программа в операционной системе, которая предоставляет пользователю интерфейс для взаимодействия с операционной системой через команды. Примеры командных оболочек: `bash`, `sh` (Bourne Shell), `csh` (C Shell), `zsh`, и другие. Они отличаются синтаксисом, возможностями и набором встроенных функций.
2. **POSIX** (Portable Operating System Interface) - это стандарт, определяющий интерфейс между операционной системой и прикладными программами. POSIX обеспечивает совместимость между различными операционными системами.
3. **Переменные и массивы в `bash`** определяются следующим образом:
 - Переменные: `variable=value`
 - Массивы: `array_name=([index1]=value1 [index2]=value2 ...)`
4. **Операторы `let` и `read`:**
 - `let` используется для выполнения арифметических операций в `bash`.
 - `read` используется для считывания ввода пользователя в переменные.
5. **Арифметические операции** в `bash` включают операции сложения, вычитания, умножения, деления, остатка от деления и другие арифметические операции.
6. **Оператор `(())`** используется для выполнения арифметических вычислений в `bash`.

7. **Стандартные имена переменных** включают HOME, PATH, USER, SHELL, PWD и другие.
8. **Метасимволы** - это символы, которые имеют специальное значение в командной оболочке, например, *, ?, [], |.
9. **Экранирование метасимволов** выполняется путем добавления обратного слэша \ перед метасимволом.
10. **Создание и запуск командных файлов** - создается текстовый файл с командами, предоставляются права на выполнение (chmod +x filename) и запускаются через ./filename или полным путем к файлу.
11. **Определение функций в bash** осуществляется с использованием ключевого слова function или просто указанием имени функции и блока кода.
12. **Проверка файла** на то, является ли он каталогом или обычным файлом, осуществляется с помощью команды test -d filename для каталога и test -f filename для обычного файла.
13. **set, typeset и unset** предназначены для управления переменными: установка значений, определение типа переменной и удаление переменной соответственно.
14. **Передача параметров в командные файлы** осуществляется через \$1, \$2, \$3, и т.д. для первого, второго, третьего параметра и так далее.
15. **Специальные переменные bash** включают \$0 (имя скрипта), \$# (количество переданных аргументов), \$* (все параметры в виде одной строки), @\$ (все параметры в виде списка), \$\$ (PID текущего процесса) и другие.

4 Выводы

В ходе выполнения работы я изучил основы программирования в ос Linux и научился писать простые командные файлы.