

Лабораторная работа №12

Дисциплина: Операционные системы

Жибицкая Евгения Дмитриевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Контрольные вопросы	12
5	Выводы	14

Список иллюстраций

3.1	Создание файла	7
3.2	Программа 1	7
3.3	Запуск программы 1	8
3.4	Создание второго файла	8
3.5	Программа 2	8
3.6	Запуск программы 2	9
3.7	Программа 3	9
3.8	Запуск программы 3	10
3.9	Программа 4	11
3.10	Запуск программы 4	11

Список таблиц

1 Цель работы

Изучение основ программирования в оболочке ОС UNIX. Написание небольших программ.

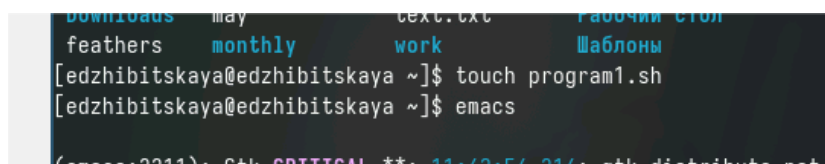
2 Задание

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.

3 Выполнение лабораторной работы

Перед началом непосредственно написания скриптов изучим принципы работы, узнаем синтаксис и особенности такого вида программирования.

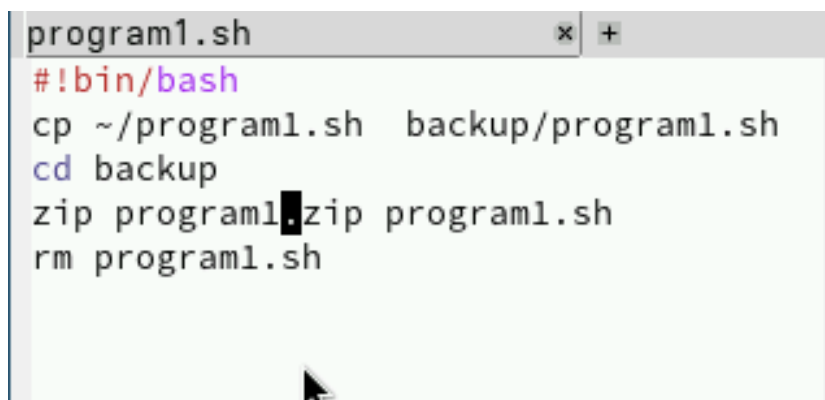
Затем перейдем к написанию программ. Создадим директорию backup для резервного копирования и файл program1.sh. Откроем его для редактирования(рис. 3.1).



```
edzhibitskaya@edzhibitskaya ~]$ touch program1.sh
edzhibitskaya@edzhibitskaya ~]$ emacs
(emacs:3211): Gtk-CRITICAL **: 11:42:54.214: gtk_distribution_pat
```

Рис. 3.1: Создание файла

Напишем первую программу. Копируем файл в нужный нам каталог, переходим в него, архивируем файл и удаляем ненужную версию (рис. 3.2).



```
#!/bin/bash
cp ~/program1.sh backup/program1.sh
cd backup
zip program1.zip program1.sh
rm program1.sh
```

Рис. 3.2: Программа 1

Дадим файлу право на исполнение(chmod +x) и соответственно запустим его

(рис. 3.3).

```
[edzhibitskaya@edzhibitskaya ~]$ ./program1.sh
  adding: program1.sh (deflated 45%)
[edzhibitskaya@edzhibitskaya ~]$ cd backup
[edzhibitskaya@edzhibitskaya backup]$ ls
program1.zip
[edzhibitskaya@edzhibitskaya backup]$
```

Рис. 3.3: Запуск программы 1

Создадим файл для второй программы, запустим его в редакторе emacs (рис. 3.4).

```
[edzhibitskaya@edzhibitskaya ~]$ touch program2.sh
[edzhibitskaya@edzhibitskaya ~]$ emacs
```

Рис. 3.4: Создание второго файла

Напишем программу, получающую на вход различные данные и циклом выводящую их обратно на экран (рис. 3.5).

```
#!/bin/bash
for A in $*
do
    echo $A
done
```

Рис. 3.5: Программа 2

Запустим программу, предварительно дав ей право на исполнение (рис. 3.6).


```
[edzhibitskaya@edzhibitskaya ~]$ ./program2.sh 1 34 7 87 9 0 er  
t 5  
1  
34  
7  
87  
9  
0  
er  
t  
5  
[edzhibitskaya@edzhibitskaya ~]$
```

Рис. 3.6: Запуск программы 2

Перейдем к написанию скрипта 3. Также создадим файл, дадим его необходимые права и откроем в редакторе emacs.

Запустим цикл, который с помощью ключей определяет тип файла(или каталога) и делает проверку на его права(чтение или редактирование) (рис. 3.7).

Запустим файл, посмотрим на результат (рис. 3.8).

```
#!/bin/bash  
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"  
            if test -r $A  
            then  
                echo "readable"  
            else  
                echo neither writeable or readable  
            fi  
        fi  
    fi  
done
```

Рис. 3.7: Программа 3

```
program3.sh is a file and writeable  
readable  
program3.sh~ is a file and writeable  
readable  
reports is a directory  
ski.plases is a directory  
text.txt is a file and writeable  
readable  
work is a directory  
Видео is a directory  
Документы is a directory  
Загрузки is a directory  
Изображения is a directory  
Музыка is a directory  
Общедоступные is a directory  
Рабочий стол is a directory  
Шаблоны is a directory  
[edzhibitskaya@edzhibitskaya ~]$
```

Рис. 3.8: Запуск программы 3

Перейдем к четвертому скрипту. Создадим файл, наделим его правами и откроем. Напишем программу. Здесь нам надо на вход получить название каталога и расширение, а затем пройтись по всем файлам и посчитать все, у которых такое же расширение (рис. 3.9).

Запустим программу (рис. 3.10).

```
#!/bin/bash
echo "Write a format of the file:"
read format
echo "Write a directory:"
read dir
let x=0
do
    if find ${dir} *.${format} - type f | wc -l
    then
        x=x+1
    fi
done
```

Рис. 3.9: Программа 4

```
./program4.sh: строка 10: синтаксическая ошибка: неждана
[edzhibitskaya@edzhibitskaya ~]$ emacs
[edzhibitskaya@edzhibitskaya ~]$ ./program4.sh
Write a format of the file:
pdf
Write a directory:
Downloads
```

Рис. 3.10: Запуск программы 4

4 Контрольные вопросы

1. Командная оболочка - это интерфейс командной строки, который позволяет пользователю работать с операционной системой путем ввода текстовых команд. Примеры командных оболочек: `bash`, `zsh`, `sh`, `ksh`. Они отличаются друг от друга своими возможностями и синтаксисом команд.
2. POSIX (Portable Operating System Interface) - это набор стандартов, разработанных IEEE, определяющих интерфейсы для взаимодействия между операционной системой и прикладными программами.
3. Переменные в `bash` определяются путем присвоения им значения, например: `variable=value`

Массивы в `bash` определяются следующим образом: `array=(value1 value2 value3)`

4. Оператор `let` используется для выполнения арифметических операций, а оператор `read` позволяет считывать данные с клавиатуры.
5. В `bash` можно применять арифметические операции такие как сложение, вычитание, умножение, деление, остаток от деления и т.д.
6. Операция `(())` используется для вычисления арифметических выражений в `bash`.
7. Некоторые стандартные имена переменных в `bash`: `HOME`, `PATH`, `USER`, `SHELL`, `PWD` и т.д.

8. Метасимволы - это символы, которые имеют специальное значение в командной оболочке, например “*”, “?”, “[”, “\$” и т.д.
9. Для экранирования метасимволов в bash используют обратный слэш “\”, например: *.
10. Командные файлы создаются с помощью текстового редактора и сохраняются с расширением “.sh”. Для их запуска необходимо установить права на выполнение с помощью команды `chmod +x`, а затем выполнить файл через команду `./название_файла.sh`.
11. Функции в bash определяются следующим образом: `function_name() { команды }`
12. Для определения, является ли файл каталогом или обычным файлом, можно использовать команду `test`:


```
if [ -d file ]; then echo “Это каталог” else echo “Это обычный файл” fi
```
13. Команды `set`, `typeset` и `unset` используются для работы с переменными в bash. `set` устанавливает значения переменных среды, `typeset` определяет тип переменной и устанавливает ее свойства, `unset` удаляет переменные.
14. Параметры передаются в командные файлы как аргументы командной строки. Например, `$1` - первый параметр, `$2` - второй параметр и так далее.
15. Специальные переменные языка bash:
 - `$0` - название выполняемого файла
 - `$#` - количество аргументов
 - `$@` - все аргументы в виде списка
 - `$$` - идентификатор текущего процесса
 - `$?` - код завершения последней выполненной команды

5 Выводы

В ходе работы мы познакомились с различными командами и особенностями программирования в оболочке UNIX, освоили написание небольших скриптов.