

Архитектура компьютеров и операционные системы | Операционные системы

**Лабораторная работа № 11. Программирование в командном
процессоре ОС UNIX. Ветвления и циклы**

Мугари Абдеррахим - НКАбд-03-22

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы :	6
2.1	Контрольные вопросы:	10
2.2	выводы по результатам выполнения заданий:	12
3	Выводы, согласованные с целью работы:	13

Список иллюстраций

2.1	Написание первой программы	6
2.2	изменение прав доступа и выполнение программы	6
2.3	Написание второй программы	7
2.4	компиляция кода, написанного на С	7
2.5	Написание командного файла	8
2.6	изменение прав доступа и выполнение программы	8
2.7	написание третьего программы	9
2.8	изменение прав доступа и выполнение программы	9
2.9	написание четвертой программы	10
2.10	изменение прав доступа и выполнение программы	10

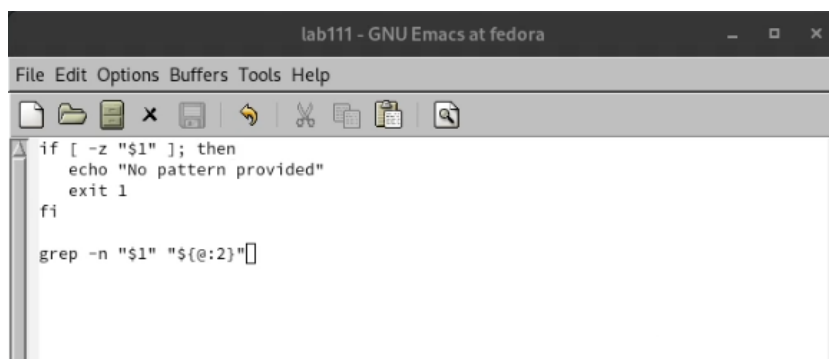
Список таблиц

1 Цель работы

- Изучить основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

2 Выполнение лабораторной работы :

- Прежде всего, я написал программу, которая ищет шаблон в файле и отображает каждую строку, содержащую его, с указанием ее номера (рис. 2.1)

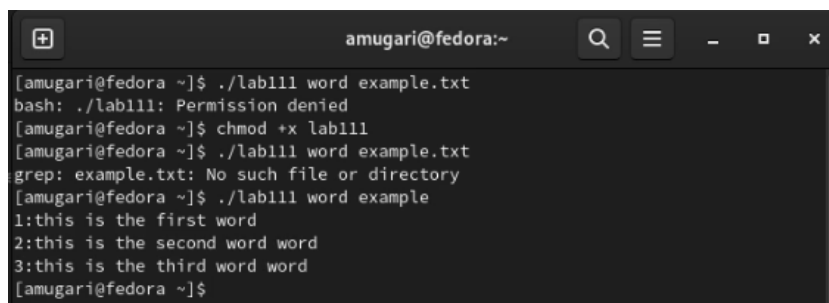


```
if [ -z "$1" ]; then
    echo "No pattern provided"
    exit 1
fi

grep -n "$1" "${@:2}"
```

Рис. 2.1: Написание первой программы

- после этого мне пришлось изменить права доступа к файлу, содержащему код, и добавить к нему выполнение, и после этого я запустил код, и он заработал (рис. 2.2)



```
[amugari@fedora ~]$ ./lab111 word example.txt
bash: ./lab111: Permission denied
[amugari@fedora ~]$ chmod +x lab111
[amugari@fedora ~]$ ./lab111 word example.txt
grep: example.txt: No such file or directory
[amugari@fedora ~]$ ./lab111 word example
1:this is the first word
2:this is the second word word
3:this is the third word word
[amugari@fedora ~]$
```

Рис. 2.2: изменение прав доступа и выполнение программы

- затем здесь мне пришлось написать программу на С, которая вводит число и определяет, больше ли оно нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции `exit(n)`, передавая информацию о коде завершения в

оболочку.(рис. 2.3)

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    int number;
    printf("Enter a number: ");
    scanf("%d", &number);

    if (number > 0) {
        printf("%d is greater than 0\n", number);
        exit(1);
    }
    else if (number < 0) {
        printf("%d is less than 0\n", number);
        exit(-1);
    }
    else {
        printf("%d is equal to 0\n", number);
        exit(0);
    }
}
```

Рис. 2.3: Написание второй программы

- после написания кода колледжа мне пришлось скомпилировать его перед выполнением (рис. 2.4)

```
[amugari@fedora ~]$ gcc lab112.c -o lab112.out
[amugari@fedora ~]$ ls
'2023-04-22 08-46-15.mp4'  '#lab07.sh'  lab3      text
backup                  lab07.sh~    lab4      third.mp4
bin                    lab10        main      Untitled.ipynb
conf.txt               lab111       monthly   Videos
Desktop                lab112       Music     work
Documents              lab112~     Pictures  Архитектура
Downloads              lab112.c    Public    pnp5
example                lab112.out  reports
file.txt               lab2         Templates
[amugari@fedora ~]$ emacs lab112
```

Рис. 2.4: компиляция кода, написанного на С

- затем я написал командный файл, который должен вызывать эту программу и, проанализировав ее с помощью команды `$?`, выдать сообщение о том, какое число было введено. (рис. 2.5)

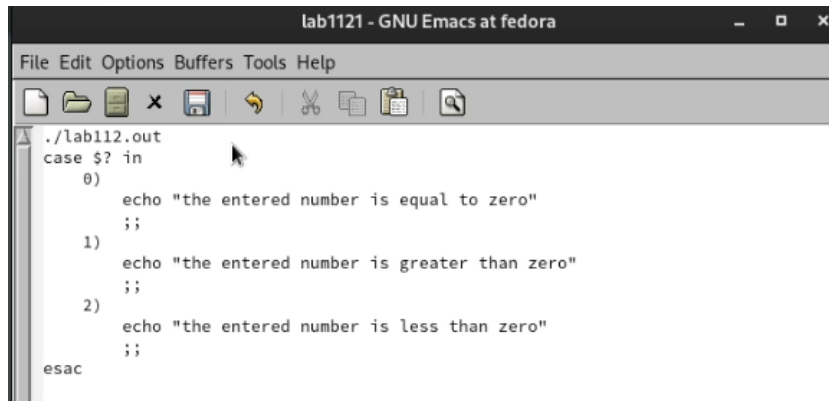


Рис. 2.5: Написание командного файла

- здесь я изменил право доступа к командному файлу, а затем выполнил его для вызова другой программы, написанной на C, где я протестировал три возможные ситуации, и это сработало (рис. 2.6)

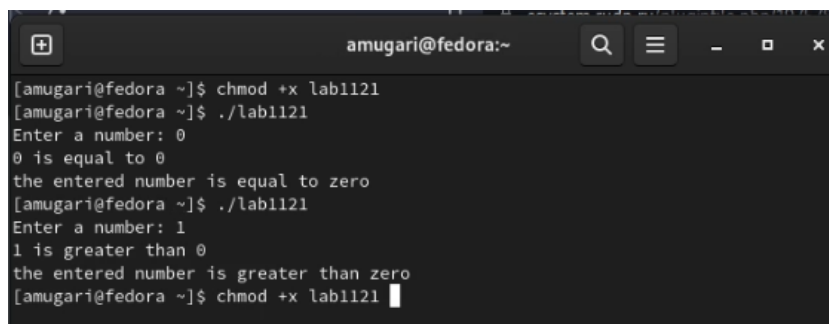
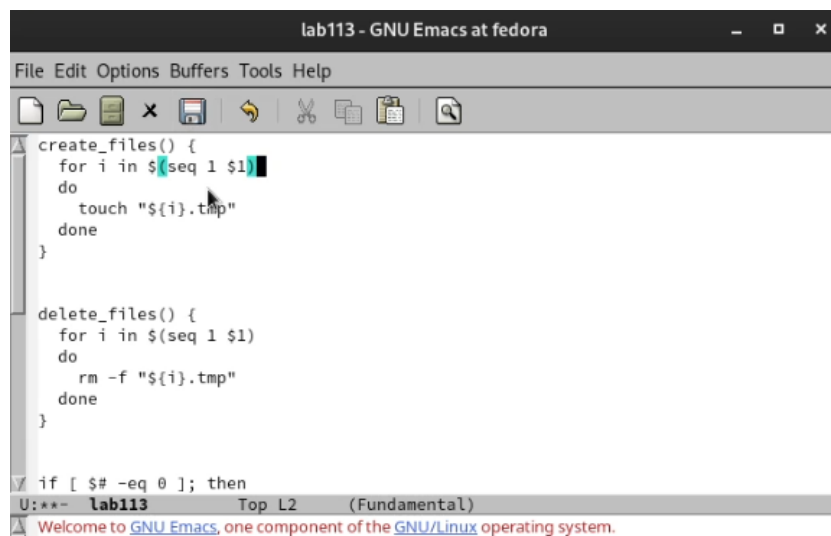


Рис. 2.6: изменение прав доступа и выполнение программы

- затем я написал код, который создает указанное количество файлов, пронумерованных последовательно от 1 до N. Количество файлов, которые должны быть созданы, передается в качестве аргументов командной строки. Один и тот же командный

файл должен иметь возможность удалять все созданные им файлы (если они существуют). (рис. 2.7)



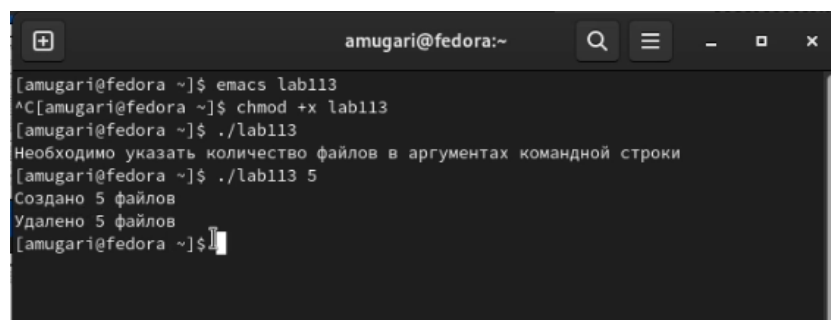
```
lab113 - GNU Emacs at fedora
File Edit Options Buffers Tools Help
create_files() {
  for i in $(seq 1 $1)
  do
    touch "${i}.tmp"
  done
}

delete_files() {
  for i in $(seq 1 $1)
  do
    rm -f "${i}.tmp"
  done
}

if [ $# -eq 0 ]; then
U:***- lab113 Top L2 (Fundamental)
Welcome to GNU Emacs, one component of the GNU/Linux operating system.
```

Рис. 2.7: написание третьего программы

- затем мне пришлось изменить права доступа к коду и выполнить программу, и все работало нормально (рис. 2.8)

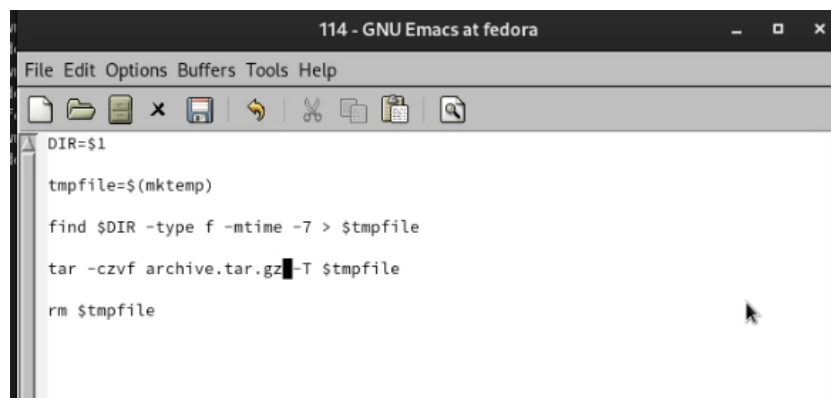


```
amugari@fedora:~
[amugari@fedora ~]$ emacs lab113
^C[amugari@fedora ~]$ chmod +x lab113
[amugari@fedora ~]$ ./lab113
Необходимо указать количество файлов в аргументах командной строки
[amugari@fedora ~]$ ./lab113 5
Создано 5 файлов
Удалено 5 файлов
[amugari@fedora ~]$
```

Рис. 2.8: изменение прав доступа и выполнение программы

- здесь я написал код, который использует команду **tar** для архивирования всех файлов в указанном каталоге. Меняем его таким образом, чтобы упаковывались только те файлы, которые были изменены менее недели назад (с помощью команды

find) (рис. 2.9)



```
114 - GNU Emacs at fedora
File Edit Options Buffers Tools Help

DIR=$1

tmpfile=$(mktemp)

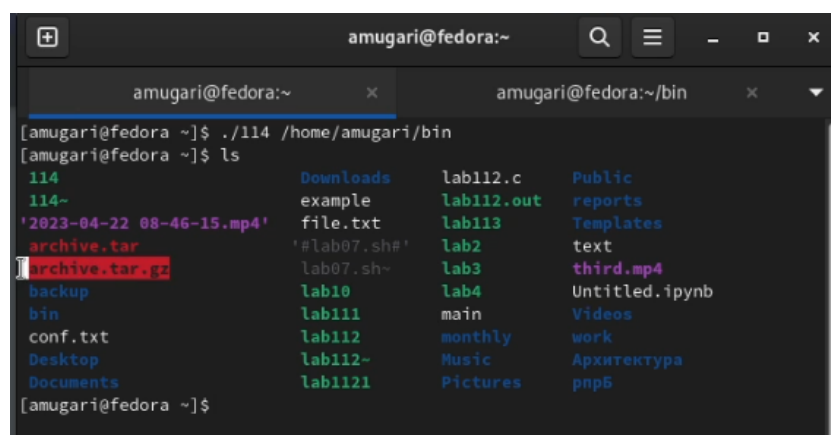
find $DIR -type f -mtime -7 > $tmpfile

tar -czvf archive.tar.gz -T $tmpfile

rm $tmpfile
```

Рис. 2.9: написание четвертой программы

- наконец, я изменил права доступа и выполнил файл, и все заработало, как и должно быть (рис. 2.10)



```
amugari@fedora:~
[amugari@fedora ~]$ ./114 /home/amugari/bin
[amugari@fedora ~]$ ls
114          Downloads    lab112.c     Public
114~         example     lab112.out   reports
'2023-04-22 08-46-15.mp4' file.txt     lab113       Templates
archive.tar  'lab07.sh'  lab2         text
archive.tar.gz lab07.sh~   lab3         third.mp4
backup       lab10       lab4         Untitled.ipynb
bin          lab111     main         Videos
conf.txt     lab112     monthly     work
Desktop      lab112~   Music       Архитектура
Documents    lab1121   Pictures    pnp5
[amugari@fedora ~]$
```

Рис. 2.10: изменение прав доступа и выполнение программы

2.1 Контрольные вопросы:

1. Команда getopts в UNIX-подобных операционных системах используется для анализа аргументов командной строки, переданных в скрипт. Она об-

рабатывает короткие опции, заданные после символа “-” и длинные опции, заданные после символа “-”.

2. Эта команда позволяет программисту легко определять и обрабатывать опции и аргументы, переданные в командной строке. Метасимволы в UNIX используются для генерации имен файлов и для манипуляции с файловой системой. Например, символ звездочки (*) используется для обозначения любого количества любых символов в имени файла, а символ вопросительного знака (?) используется для обозначения любого одного символа в имени файла.
3. Операторы управления действиями в UNIX-подобных системах включают в себя операторы условного выполнения (if, case), операторы циклов (for, while, until) и операторы перенаправления ввода-вывода (>, », <, «).
4. Для прерывания цикла в UNIX используются операторы break и continue. Оператор break прерывает выполнение цикла и переходит к следующей инструкции после цикла, а оператор continue прерывает текущую итерацию цикла и переходит к следующей итерации.
5. Команда false возвращает ненулевое значение и используется для проверки скриптов на ошибки, а команда true возвращает нулевое значение и используется для явной инициализации переменных и для создания бесконечных циклов.
6. Данная строка проверяет наличие файла в директории man\$s/ с именем, состоящим из переменной \$i и переменной \$s, и если файл существует, то скрипт продолжает выполнение.
7. Конструкция while выполняет цикл, пока условие истинно, а конструкция until выполняет цикл, пока условие ложно. То есть, пока условие в while истинно, цикл будет выполняться, а в until, пока условие ложно. Также, в

`while` цикл будет выполняться, если условие истинно с самого начала, а в `until` - если условие ложно.

2.2 выводы по результатам выполнения заданий:

- В ходе этой лабораторной работы у меня была возможность научиться программировать в операционной системе UNIX и приобрести практические навыки написания сложного кода с использованием логических управляющих структур и циклов.

3 Выводы, согласованные с целью работы:

- Изучить основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.