

Лабораторная работа №11

**Программирование в командном процессоре ОС UNIX. Ветвления и
циклы**

Ежова Алиса Михайловна

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
4	Контрольные вопросы	10
5	Выводы	11

Список иллюстраций

3.1	Код	6
3.2	Код 2	7
3.3	Код 3	8
3.4	Код 4	8

1 Цель работы

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

2 Задание

1. Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами: `-iinputfile` — прочитать данные из указанного файла; `-ooutputfile` — вывести данные в указанный файл; `-r` — шаблон — указать шаблон для поиска; `-C` — различать большие и малые буквы; `-n` — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом `-r`.
2. Написать на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции `exit(n)`, передавая информацию в о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено.
3. Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до ∞ (например `1.tmp`, `2.tmp`, `3.tmp`, `4.tmp` и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют).
4. Написать командный файл, который с помощью команды `tag` запаковывает в архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду `find`).

3 Выполнение лабораторной работы

- 1) Используя команды `getopts` `grep`, написала командный файл, который анализирует командную строку с ключами: `-i`inputfile — прочитать данные из указанного файла; `-o`outputfile — вывести данные в указанный файл; `-r`шаблон — указать шаблон для поиска; `-C` — различать большие и малые буквы; `-n` — выдавать номера строк. а затем ищет в указанном файле файле нужные строки, определяемые ключом `-p`:

```
#!/bin/bash cflag=0; nflag=0; while getopts i:o:p:C:n opt do case optini)ival =OPTARG;;  
o) oval=OPTARG;; p)pval =OPTARG;; C) cflag=1;; n) nflag=1;; esac done if [  
$cflag -a $nflag ] then grep -n $pval ival >oval elif test $cflag then grep $pval  
ival >oval elif test $nflag then grep -n -i $pval ival >oval else grep -i $pval  
ival >oval fi
```

```
amezhova@dk2n24 ~ $ touch lab11_1.sh  
amezhova@dk2n24 ~ $ chmod +x lab11_1.sh  
amezhova@dk2n24 ~ $ ./lab11_1.sh  
./lab11_1.sh: строка 16: $oval: неоднозначное перенаправление  
amezhova@dk2n24 ~ $ ./lab11_1.sh -i text.txt -o fout.txt -p файлы -C -n  
amezhova@dk2n24 ~ $ cat fout.txt  
amezhova@dk2n24 ~ $
```

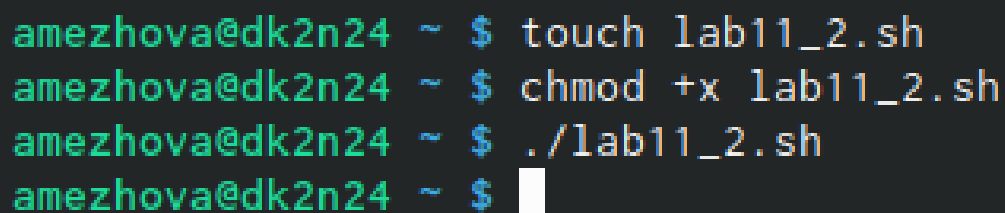
Рис. 3.1: Код

```
#!/bin/bash
cflag=0;
nflag=0;
while getopts i:o:p:C:n opt
do
case $opt in
i) ival=$OPTARG;;
o) oval=$OPTARG;;
p) pval=$OPTARG;;
C) cflag=1;;
n) nflag=1;;
esac
done
if [ $cflag -a $nflag ]
then
grep -n $pval $ival>$oval
elif test $cflag
then
grep $pval $ival>$oval
elif test $nflag
then
grep -n -i $pval $ival>$oval
else
grep -i $pval $ival>$oval
fi
```

Рис. 3.2: Код 2

- 2) Написала сначала на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем завершила программу при помощи функции `exit(n)`, передавая информацию о коде завершения в оболочку. Командный файл вызовет эту программу и, проанализировав с помощью команды `$?`, выдаст сообщение о том, какое число было введено:

```
#!/bin/bash gcc -c script2.c gcc -o script2 script2.c ./script2 case $? in 1) echo отрицательное;; 2) echo равно нулю;; 3) echo положительное;; esac
```



```
amezhova@dk2n24 ~ $ touch lab11_2.sh
amezhova@dk2n24 ~ $ chmod +x lab11_2.sh
amezhova@dk2n24 ~ $ ./lab11_2.sh
amezhova@dk2n24 ~ $
```

Рис. 3.3: Код 3

```
#!/bin/bash
gcc -c script2.c
gcc -o script2 script2.c
./script2
case $? in
1) echo отрицательное;;
2) echo равно нулю;;
3) echo положительное;;
esac
```

Рис. 3.4: Код 4

3) Написала командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N:

```
#!/bin/bash let i=$1+1 while (( i-=1 )) do touch $i.tmp done let j=$2+1; while (( j-=1 )) do rm $j.tmp done
```

4) Написала командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировала его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад.

```
#!/bin/bash (find $1 -mtime -7 -daystart) | xargs tar -cf arhiv.tar
```

4 Контрольные вопросы

1. Каково предназначение команды `getopts`? Ответ: Создание по пользовательским аргументам.
2. Какое отношение метасимволы имеют к генерации имён файлов? Ответ: Используют как файлы так и аргументы.
3. Какие операторы управления действиями вы знаете? Ответ: `if`, `else`, `elif`, `fi`, `while`, `do`, `done`, `until`, `do`, `done`, `for`, `in`, `do`, `done`, `case`, `in`, `esac`
4. Какие операторы используются для прерывания цикла? Ответ:
 - a) `for` – будет выполнять действие до тех пор, пока есть объекты для выполнения.
 - b) `while` – выполняет действие до тех пор, пока условие является истинным.
 - c) `until` – будет выполняться пока условие не станет правдиво.
5. Для чего нужны команды `false` и `true`? Ответ: `until` – будет выполняться до тех пор, пока условие не станет `true`, т.е. пока оно не станет `false`.
6. Что означает строка `if test -f man/i.$s`, встреченная в командном файле? Ответ: Проверяет если существует файл его размерность и тип с двумя разными расширениями, заменяя через переменные.
7. Объясните различия между конструкциями `while` и `until`. Ответ: `while` – выполняет действие до тех пор, пока условие является истинным. `until` – будет выполняться до тех пор, пока условие не станет истинным, т.е. пока оно `false`

5 Выводы

В ходе выполнения Лабораторной работы №11, я изучила основы программирования в оболочке ОС UNIX. Научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.