

Шаблон отчёта по лабораторной работе № 12

1022204143

Надиа Эззакат

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	15
5	Контрольные вопросы	16

List of Tables

List of Figures

3.1	mcredit	8
3.2	mcredit	9
3.3	ВЫВОД	9
3.4	mcredit	10
3.5	mcredit	10
3.6	ВЫВОД	11
3.7	mcredit	12
3.8	ВЫВОД	13
3.9	mcredit	14
3.10	ВЫВОД	14

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 N (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`, написать командный файл, который анализирует командную строку с ключами: `-iinputfile` — прочитать данные из указанного файла; `-ooutputfile` — вывести данные в указанный файл; `-rшаблон` — указать шаблон для поиска; `-C` — различать большие и малые буквы; `-n` — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом `-p`.

```

lab12.sh          [-M--] 14 L:[ 1+25 26/ 35] *(5
#!/bin/bash
while getopts i:o:p:Cn optletter
do case $optletter in
i) iflag=1; ival=$OPTARG;;
o) oflag=1; oval=$OPTARG;;
p) pflag=1; pval=$OPTARG;;
C) Cflag=1;;
n) nflag=1;;
*) echo incorrect input $optletter
esac
done
if (((Cflag==1)&&(nflag==1)))
then grep -e${pval} -n ${ival}
if((oflag==1))
then grep -e${pval} -n ${ival} > ${oval}
fi
fi
if (((Cflag==0)&&(nflag==1)))
then grep -e${pval} -i -n ${ival}
if((oflag==1))
then grep -e${pval} -i -n ${ival} > ${oval}
fi
fi
if (((Cflag==1)&&(nflag==0)))
then grep -e${pval} ${ival}
if((oflag==1))
then grep -e${pval} ${ival} > ${oval}
fi
fi
if (((Cflag==0)&&(nflag==0)))
then grep -e${pval} -i ${ival}
if((oflag==1))
then grep -e${pval} -i ${ival} > ${oval}
fi
fi

```

Figure 3.1: mcedit


```

lab12.txt      [-M--] 18 L:[ 1+ 9 10/ 10] *(285 / 285b) <EOF>
in this life there is no such thing as good or bad
people always judge you if you didn't act the way they want
so in order to be good
you have to please them.
and in order to be bad.
you just have to be against
their believes and thought.
so being good.
or being bad
is just a myth lol

```

Figure 3.2: mcedit

```

[nadiaezza@nadiaezza ~]$ mcedit lab12.sh

[nadiaezza@nadiaezza ~]$ mcedit lab12.txt

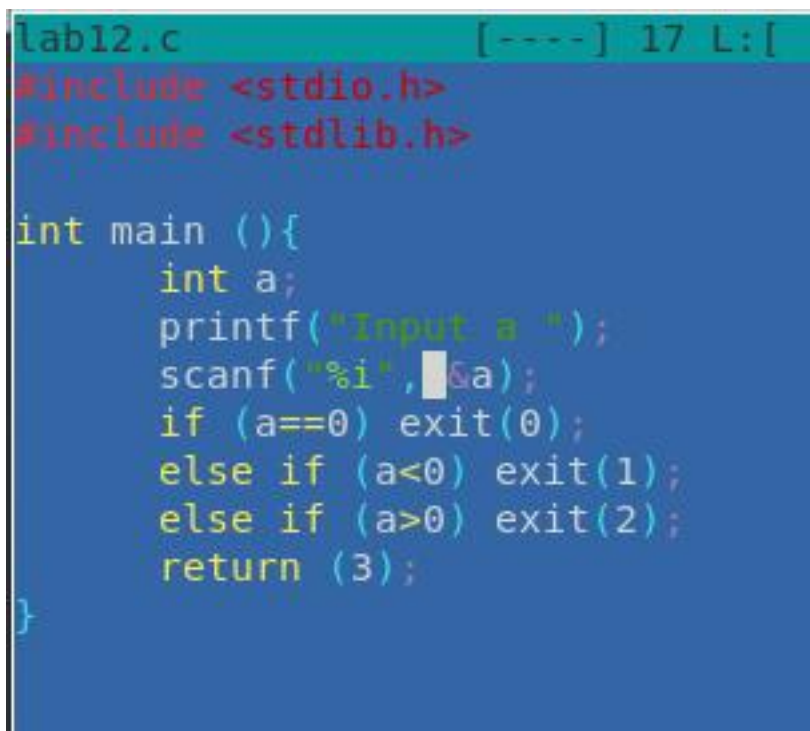
[nadiaezza@nadiaezza ~]$ ./lab12.sh -ilab12.txt -olab12_1.txt -pood
in this life there is no such thing as good or bad
so in order to be good
you just have to be againsttheir good minds
so being good
[nadiaezza@nadiaezza ~]$ ./lab12.sh -ilab12.txt -olab12_1.txt -pood -n
1:in this life there is no such thing as good or bad
3:so in order to be good
6:you just have to be againsttheir good minds
8:so being good
[nadiaezza@nadiaezza ~]$ ./lab12.sh -ilab12.txt -olab12_1.txt -pood -C -n
1:in this life there is no such thing as good or bad
3:so in order to be good
6:you just have to be againsttheir good minds
8:so being good
[nadiaezza@nadiaezza ~]$ ./lab12.sh -ilab12.txt -olab12_1.txt -pood -C
in this life there is no such thing as good or bad
so in order to be good
you just have to be againsttheir good minds
so being good
[nadiaezza@nadiaezza ~]$ █

```

Figure 3.3: вывод

2. Написала 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

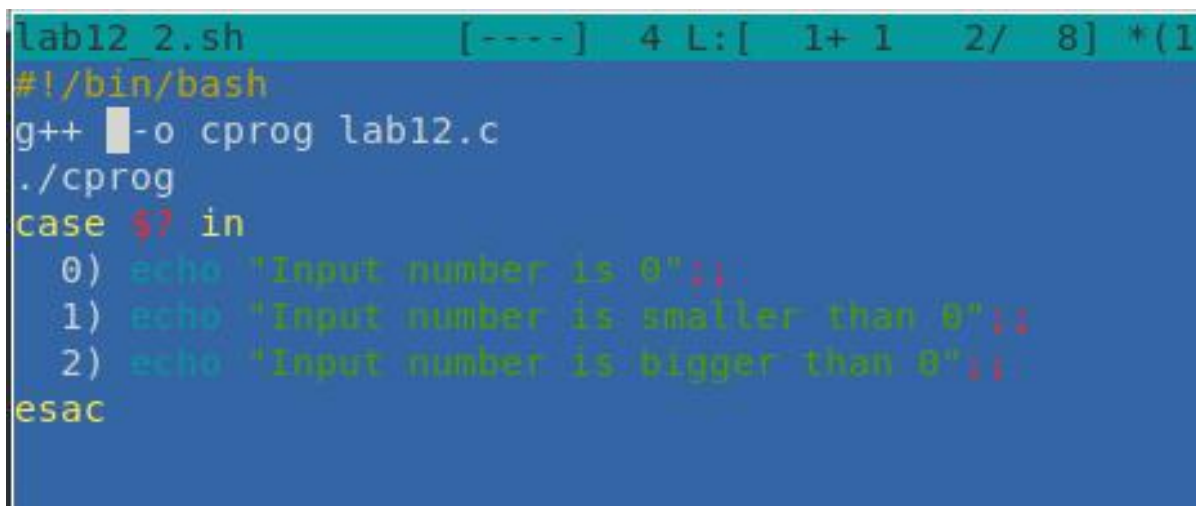
оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды \$?, выдать сообщение о том, какое число было введено.



```
lab12.c [-----] 17 L:[
#include <stdio.h>
#include <stdlib.h>

int main (){
    int a;
    printf("Input a ");
    scanf("%i", &a);
    if (a==0) exit(0);
    else if (a<0) exit(1);
    else if (a>0) exit(2);
    return (3);
}
```

Figure 3.4: mcedit



```
lab12_2.sh [-----] 4 L:[ 1+ 1 2/ 8] *(1
#!/bin/bash
g++ -o cprog lab12.c
./cprog
case $? in
    0) echo "Input number is 0";;
    1) echo "Input number is smaller than 0";;
    2) echo "Input number is bigger than 0";;
esac
```

Figure 3.5: mcedit

```
[nadiaezza@nadiaezza ~]$ mcedit lab12.c

[nadiaezza@nadiaezza ~]$ mcedit lab12_2.sh

[nadiaezza@nadiaezza ~]$ ./lab12_2.sh
Input a 0
Input number is 0
[nadiaezza@nadiaezza ~]$ ./lab12_2.sh
Input a 10
Input number is bigger than 0
[nadiaezza@nadiaezza ~]$ ./lab12_2.sh
Input a -9
Input number is smaller than 0
[nadiaezza@nadiaezza ~]$ █
```

Figure 3.6: вывод

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

```
lab12_3.sh      [-M--]  2 L:[  1+18  19/ 1
#!/bin/bash
let dflag=0;
while getopts a:d optletter
do case $optletter in
a) aflag=1;aval=$OPTARG;;
d) dflag=1;;
*) echo Incorrect input $optletter
esac
done
if ((dflag==0))
then for ((i=1;i<=aval;i++))
do touch ${i}.tmp
done
fi
if ((dflag==1))
then for ((i=1;i<=aval;i++))
do rm ${i}.tmp
done
fi
```

Figure 3.7: mcedit

```

[nadiaezza@nadiaezza ~]$ ./lab12_3.sh -a10
[nadiaezza@nadiaezza ~]$ ls
10.tmp          Desktop         lab12.txt
1.tmp           dmesg          legalcode.txt
2.tmp           dmesg less     letters
3.tmp           doc1           #lol#
4.tmp           Documents      may
5.tmp           Downloads      memos
6.tmp           file.txt       misk
7.tmp           #lab07.sh#    monthly
8.tmp           lab07.sh       morefun
9.tmp           lab10.sh       Music
AA             lab11_2.sh     os-intro
abc1           lab11.sh       Pictures
academic-laboratory-report-template lab12_1.txt    Public
academic-presentation-markdown-template lab12_2.sh     ski.plases
backup         lab12_3.sh     Templates
conf.txt       lab12.c        Videos
cprog         lab12.sh       work
[nadiaezza@nadiaezza ~]$ ./lab12_3.sh -a10 -d
[nadiaezza@nadiaezza ~]$ ls
AA             Documents      lab12.c        Music
abc1           Downloads     lab12.sh       os-intro
academic-laboratory-report-template file.txt       lab12.txt      Pictures
academic-presentation-markdown-template #lab07.sh#    legalcode.txt  Public
backup         lab07.sh      letters        ski.plases
conf.txt       lab10.sh     #lol#         Templates
cprog         lab11_2.sh   may           Videos
Desktop       lab11.sh     memos         work
dmesg         lab12_1.txt  misk
dmesg less    lab12_2.sh   monthly
doc1          lab12_3.sh   morefun
[nadiaezza@nadiaezza ~]$ █

```

Figure 3.8: вывод

4. Написала командный файл, который с помощью команды tar 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).


```
lab12 4.sh [-M--] 55 L:[ 1+ 2 3/ 3] *(90 / 90b) <E
#!/bin/bash
tar -cf lab12_4.tar $0
find $0 -mtime -7 -exec tar -rf lab12_4modif.tar {} \;
```

Figure 3.9: mcedit

```
[nadiaezza@nadiaezza ~]$ mcedit lab12_4.sh

[nadiaezza@nadiaezza ~]$ ./lab12_4.sh os-intro
bash: ./lab12_4.sh: Permission denied
[nadiaezza@nadiaezza ~]$ chmod +x lab12_4.sh
[nadiaezza@nadiaezza ~]$ ./lab12_4.sh os-intro

[nadiaezza@nadiaezza ~]$
[nadiaezza@nadiaezza ~]$ ls
AA                                lab07.sh                        #lol#
abc1                             lab10.sh                       may
academic-laboratory-report-template lab11_2.sh                     memos
academic-presentation-markdown-template lab11.sh                       misk
backup                          lab12_1.txt                    monthly
conf.txt                        lab12_2.sh                     morefun
cprog                           lab12_3.sh                     Music
Desktop                         lab12_4modif.tar               os-intro
dmesg                           lab12_4.sh                     Pictures
dmesg less                      lab12_4.tar                    Public
doc1                             lab12.c                        ski.plases
Documents                       lab12.sh                       Templates
Downloads                       lab12.txt                      Videos
file.txt                        legalcode.txt                  work
#lab07.sh#
[nadiaezza@nadiaezza ~]$
```

Figure 3.10: вывод

4 Выводы

В результате работы, научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

5 Контрольные вопросы

1. Весьма необходимой при программировании является команда `getopts`, которая осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. Синтаксис команды следующий: `getopts option-string variable [arg...]`. Флаги – это опции командной строки, обычно помеченные знаком минус; Например, `-F` является флагом для команды `ls -F`. Иногда эти флаги имеют аргументы, связанные с ними. Программы интерпретируют эти флаги, соответствующим образом изменяя свое поведение. Строка опций `option-string` — это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за этой буквой должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введенные данные с помощью оператора `case`. Предположим, необходимо распознать командную строку следующего формата: `testprog -infile_in.txt -outfile_out.doc -L -t -r` Вот как выглядит использование оператора `getopts` в этом случае:

```
while getopts o:i:Ltr optletter do case $optletter in o) oflag = 1; oval =OPTARG;; i) iflag=1; ival=$OPTARG;; L) Lflag=1;; t) tflag=1;; r) rflag=1;; *) echo Illegal option $optletter esac done
```

 Функция `getopts` включает две специальные переменные среды – `OPTARG` и `OPTIND`. Если ожидается дополнительное значение, то `OPTARG` устанавливается в значение этого аргумента (будет равна `file_in.txt` для опции `i` и `file_out.doc` для опции `o`).

OPTIND является числовым индексом на упомянутый аргумент. Функция `getopts` также понимает переменные типа массив, следовательно, можно использовать ее в функции не только для синтаксического анализа аргументов функций, но и для анализа введенных пользователем данных.

2. При перечислении имен файлов текущего каталога можно использовать следующие символы:

- — соответствует произвольной, в том числе и пустой строке;

? — соответствует любому одному символу;

[c1-c1] — соответствует любому символу, лексикографически находящемуся между символами c1 и c2.

`echo *` — выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`;

`ls *.c` — выведет все файлы с последними двумя символами, равными `.c`.

`echo prog.?` — выдаст все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются `prog.`

[a-z]* — соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.

3. Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости от результатов проверки некоторого условия. Для решения подобных задач язык программирования `bash` предоставляет Вам возможность использовать такие управляющие конструкции, как `for`, `case`, `if` и `while`. С точки зрения командного процессора эти управляющие конструкции являются обычными командами и могут использоваться как при создании командных файлов, так и при работе в интерактивном режиме. Команды, реализующие подобные конструкции, по сути дела являются операторами языка программирования `bash`. Поэтому при описании языка программирования `bash` термин оператор будет использоваться наравне с термином команда.

4. Два несложных способа позволяют вам прерывать циклы в оболочке `bash`. Команда `break` завершает выполнение цикла, а команда `continue` завершает данную итерацию блока операторов. Команда `break` полезна для завершения цикла `while` в ситуациях, когда условие перестает быть правильным. Пример бесконечного цикла `while`, с прерыванием в момент, когда файл перестает существовать:

```
while true
do
if [! -f $file]
then
break
fi
sleep 10
done
```

5. Команды ОС UNIX возвращают код завершения, значение которого может быть использовано для принятия решения о дальнейших действиях. Команда `test`, например, создана специально для использования в командных файлах. Единственная функция этой
6. Введенная строка означает условие существования файла `man s/i.$s`
7. Если речь идет о 2-х параллельных действиях, то это `while`. когда мы показываем, что сначала делается 1-е действие. потом оно заканчивается при наступлении 2-го действия, применяем `until`