

Шаблон отчёта по лабораторной работе № 12

Операционные Системы

Яссин мохамад аламин

Содержание

Цель работы	1
Задание	1
Выполнение лабораторной работы	2
Выводы	4
Контрольные вопросы	4

Цель работы

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

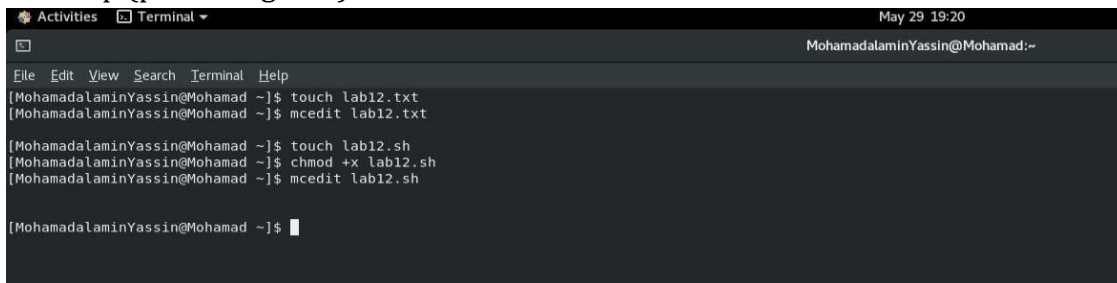
Задание

1. Используя команды `getopts` `grep`, написать 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 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 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).

Выполнение лабораторной работы

1. Используя команды getoptс 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 -r.(рис. -@fig:001)

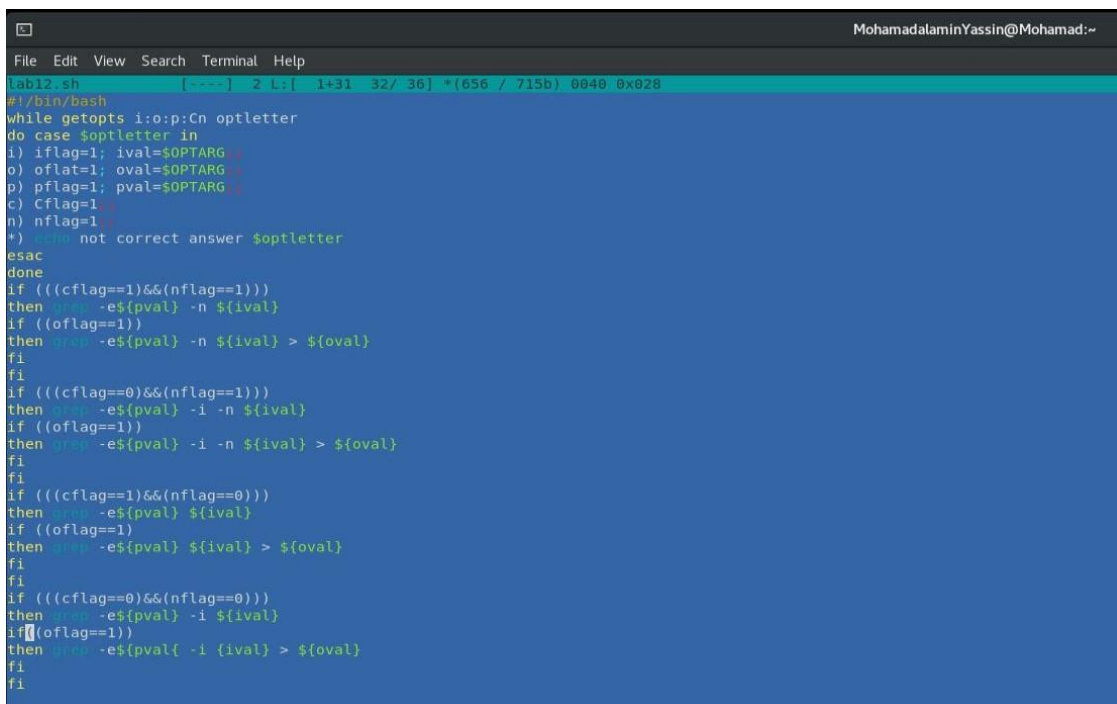


```
Activities Terminal May 29 19:20
MohamadAlaminYassin@Mohamad:~
File Edit View Search Terminal Help
[MohamadAlaminYassin@Mohamad ~]$ touch lab12.txt
[MohamadAlaminYassin@Mohamad ~]$ mcedit lab12.txt

[MohamadAlaminYassin@Mohamad ~]$ touch lab12.sh
[MohamadAlaminYassin@Mohamad ~]$ chmod +x lab12.sh
[MohamadAlaminYassin@Mohamad ~]$ mcedit lab12.sh

[MohamadAlaminYassin@Mohamad ~]$
```

mcedit



```
File Edit View Search Terminal Help
lab12.sh [-----] 2 L: 1+31 32/ 36] *(656 / 715b) 0040 0x028
#i/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 not correct answer $optletter
esac
done
if (((cflag==1)&&(nflag==1)))
then grep -e${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
```

mcedit

```
File Edit View Search Terminal Help
MohamadAlaminYassin@Mohamad:~
[MohamadAlaminYassin@Mohamad ~]$ touch lab12_2.sh
[MohamadAlaminYassin@Mohamad ~]$ chmod lab12_2.sh
chmod: missing operand after 'lab12_2.sh'
Try 'chmod --help' for more information.
[MohamadAlaminYassin@Mohamad ~]$ touch lab12.c
[MohamadAlaminYassin@Mohamad ~]$ mcedit lab12.c

[MohamadAlaminYassin@Mohamad ~]$ mcedit lab12_2.sh

[MohamadAlaminYassin@Mohamad ~]$ ./lab12_2.sh
bash: ./lab12_2.sh: Permission denied
[MohamadAlaminYassin@Mohamad ~]$ chmod +x lab12_2.sh
[MohamadAlaminYassin@Mohamad ~]$ ./lab12_2.sh
lab12.c:1:10: fatal error: iostream: No such file or directory
#include <iostream>
         ^~~~~~
compilation terminated.
./lab12_2.sh: line 3: ./cprog: No such file or directory
[MohamadAlaminYassin@Mohamad ~]$ mcedit lab12.c

[MohamadAlaminYassin@Mohamad ~]$ ./lab12_2.sh
5
number is bigger than zero
[MohamadAlaminYassin@Mohamad ~]$ ./lab12_2.sh
-2
number is lower than zero
[MohamadAlaminYassin@Mohamad ~]$ ./lab12_2.sh
0
number is equal to 0
[MohamadAlaminYassin@Mohamad ~]$
```

вывод

2. Написал 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. (рис. -@fig:002)

```
File Edit View Search Terminal Help
MohamadAlaminYassin@Mohamad:~
lab12.c [---] 15 L: [ 1+ 0 1/ 11] *(15 / 134b) 0101 0x065
#include <iostream>

int main () {
int x;
while (cin >> x;
if (x==0) exit(0);
else if (x<0) exit(1);
else if (x>0) exit(2);
return (3);
}
```

mcedit

```
MohamadAlaminYassin [Running] - Oracle VM VirtualBox
May 29 19:33
MohamadAlaminYassin@Mohamad:~
File Edit View Search Terminal Help
lab12_2.sh [---] 0 L: [ 1+ 8 9/ 9] *(167 / 167b) <E0F>
#!/bin/bash
g++ -o cprog lab12.c
./cprog
case $? in
0) echo "number is equal to 0" ;;
1) echo "number is lower than zero" ;;
2) echo "number is bigger than zero" ;;
esac
```

mcedit

```
File Edit View Search Terminal Help
MohamadAlaminYassin@Mohamad ~$ touch lab12_3.sh
MohamadAlaminYassin@Mohamad ~$ chmod +x lab12_3.sh
MohamadAlaminYassin@Mohamad ~$ mcedit lab12_3.sh

MohamadAlaminYassin@Mohamad ~$ ./lab12_3.sh -a10
./lab12_3.sh: line 15: syntax error near unexpected token `)'
./lab12_3.sh: line 15: `if (df!ag=1))'
MohamadAlaminYassin@Mohamad ~$ mcedit lab12_3.sh

MohamadAlaminYassin@Mohamad ~$ ./lab12_3.sh -a10
18.tmp 3.tmp 6.tmp 9.tmp conf.txt Desktop equipment Folder lab07.sh lab12.c labor11_2.sh labor11_3.sh- labor11.sh Music Templates Videos
1.tmp 4.tmp 7.tmp abc1 cdrom Documents file.txt lab lab12_2.sh lab12.sh labor11_2.sh- labor11_4.sh labor11.sh- Pictures Text work
2.tmp 5.tmp 8.tmp backup default.target Downloads folder *lab07.sh- lab12_3.sh lab12.txt labor11_3.sh- labor11_4.sh- may Public text.txt
MohamadAlaminYassin@Mohamad ~$
```

вывод

- 3. Написал 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).(рис. -@fig:003)

```
File Edit View Search Terminal Help
MohamadAlaminYassin@Mohamad ~$ touch lab12_3.sh
MohamadAlaminYassin@Mohamad ~$ chmod +x lab12_3.sh
MohamadAlaminYassin@Mohamad ~$ mcedit lab12_3.sh

MohamadAlaminYassin@Mohamad ~$ ./lab12_3.sh -a10
./lab12_3.sh: line 15: syntax error near unexpected token `)'
./lab12_3.sh: line 15: `if (df!ag=1))'
MohamadAlaminYassin@Mohamad ~$ mcedit lab12_3.sh

MohamadAlaminYassin@Mohamad ~$ ./lab12_3.sh -a10
18.tmp 3.tmp 6.tmp 9.tmp conf.txt Desktop equipment Folder lab07.sh lab12.c labor11_2.sh labor11_3.sh- labor11.sh Music Templates Videos
1.tmp 4.tmp 7.tmp abc1 cdrom Documents file.txt lab lab12_2.sh lab12.sh labor11_2.sh- labor11_4.sh labor11.sh- Pictures Text work
2.tmp 5.tmp 8.tmp backup default.target Downloads folder *lab07.sh- lab12_3.sh lab12.txt labor11_3.sh- labor11_4.sh- may Public text.txt
MohamadAlaminYassin@Mohamad ~$ ./lab12_3.sh -a10 -d
abc1 cdrom Documents file.txt lab lab12_2.sh lab12.sh labor11_2.sh- labor11_4.sh labor11.sh- Pictures Text work
Backup default.target Downloads folder *lab07.sh- lab12_3.sh lab12.txt labor11_3.sh- labor11_4.sh- may Public Text.txt
conf.txt Desktop equipment Folder lab07.sh lab12.c labor11_2.sh labor11_3.sh- labor11.sh Music Templates Videos
MohamadAlaminYassin@Mohamad ~$
```

вывод

- 4. Написал 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).(рис. -@fig:004)

```
MohamadAlaminYassin@Mohamad ~$ ./lab12_4.sh lab
abc1 cdrom Documents file.txt lab lab12_2.sh lab12_4.tar lab12.txt labor11_3.sh labor11_4.sh- may Public text.txt
Backup default.target Downloads folder *lab07.sh- lab12_3.sh lab12.c labor11_2.sh labor11_3.sh- labor11_4.sh- may Public Text.txt
conf.txt Desktop equipment Folder lab07.sh lab12_4.sh lab12.sh labor11_2.sh- labor11_4.sh labor11.sh- Pictures Text work
MohamadAlaminYassin@Mohamad ~$
```

вывод

Выводы

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

Контрольные вопросы

- 1. Весьма необходимой при программировании является команда getopt, которая осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. Синтаксис команды следующий: getopt 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 $optletter in
    o) iflag=1; ival=$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 `mans/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`