

Российский Университет Дружбы Народов

Факультет Физико-Математических и Естественных Наук

## ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №12

дисциплина: *Операционные системы*

Студент: ГАБРИЭЛЬ ТЬЕРРИ

Группа: НКНбд 01-20

МОСКВА 2021 г.

### Цель работы:

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

### Ход работы:

- Используя команды `getopts` `grep`, написал 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`

Для данной задачи я создал файл `program12.sh` (Рисунки 1,2) и написал соответствующие скрипты

emacs@tgabriel

File Edit Options Buffers Tools Sh-Script Help



Save



Undo



```
#!/bin/bash
iflag=0; oflag=0; pflag=0; Cflag=0; nflag=0;
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 illegal option $optletter
    esac
done
if (($pflag==0))
then echo "template not found"
else
    if (($iflag==0))
    then echo "file not found"
    else
        if (($oflag==0))
        then if (($Cflag==0))
            then if (($nflag==0))
                then grep $pval $ival
                else grep -n $pval $ival
                fi
            else if (($nflag==0))
```

```
else if (($nflag==0))
then grep -i $pval $ival
else grep -i -n $pval $ival
fi
fi
else if (($cflag==0))
then if (($nflag==0))
then grep $pval $ival > $oval
else grep -n $pval $ival > $oval
fi
else if (($nflag==0))
then grep -i $pval $ival > $oval
else grep -i -n $pval $ival > $oval
fi
fi
fi
fi
fi
```

Далее я проверил работу написанного скрипта, используя различные опции (например, команда «./program12.sh -I tmp1.txt -o tmp2.txt -p capital -C -n»), предварительно добавив право на исполнение файла (команда «chmod +x program12.sh») и создав 2 файла, которые необходимы для выполнения программы: tmp1.txt и tmp2.txt (Рисунки 3,4)

```
tgabriel@tgabriel: ~  
tgabriel@tgabriel:~$ touch program12.sh  
tgabriel@tgabriel:~$ emacs  
tgabriel@tgabriel:~$ touch tmp1.txt tmp2.txt  
tgabriel@tgabriel:~$ chmod +x program12.sh  
tgabriel@tgabriel:~$ gedit tmp1.txt  
tgabriel@tgabriel:~$ cat tmp1.txt  
I am living in Moscow  
My friend is linving in london  
I like driving  
My parents are Living in Haiti  
tgabriel@tgabriel:~$ ./program12.sh -i tmp1.txt -o tmp2.txt -p liv  
ing -C -n  
tgabriel@tgabriel:~$ cat tmp2.txt  
1:I am living in Moscow  
4:My parents are Living in Haiti  
tgabriel@tgabriel:~$ ./program12.sh -i tmp1.txt -o tmp2.txt -p liv  
ing -n  
tgabriel@tgabriel:~$ cat tmp2.txt  
1:I am living in Moscow  
tgabriel@tgabriel:~$ ./program12.sh -i tmp1.txt -c -n  
  
tgabriel@tgabriel:~$ ./program12.sh -i tmp1.txt -C -n  
template not found  
tgabriel@tgabriel:~$ ./program12.sh -o tmp2.txt -p living -C -n  
file not found  
tgabriel@tgabriel:~$
```

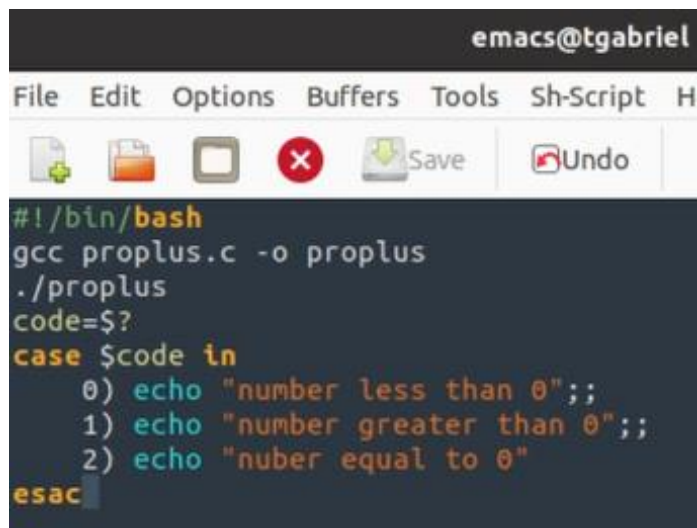
Скрипт работает корректно

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. Для данной задачи я создал 2 файла: `proplus.c` и `proplus.sh` (Рисунок 5) и написал соответствующие скрипты.(Рисунки 6,7)

```
tgabriel@tgabriel:~$ touch proplus.c proplus.sh  
tgabriel@tgabriel:~$ emacs
```



```
emacs@tgabriel
File Edit Options Buffers Tools C Help
[Icons] Save Undo
#include <stdio.h>
#include <stdlib.h>
int main()
{
    printf ("Enter a number\n");
    int a;
    scanf ("%d", &a);
    if (a<0) exit(0);
    if (a>0) exit(1);
    if (a==0) exit(2);
    return 0;
}
```



```
emacs@tgabriel
File Edit Options Buffers Tools Sh-Script H
[Icons] Save Undo
#!/bin/bash
gcc proplus.c -o proplus
./proplus
code=$?
case $code in
    0) echo "number less than 0";;
    1) echo "number greater than 0";;
    2) echo "nuber equal to 0"
esac
```

Далее я проверил работу написанных скриптов (команда «./proplus.sh»), предварительно добавив право на исполнение файла (команда «chmod +x proplus.sh») (Рисунок 8).

```

tgabriel@tgabriel:~$ chmod +x proplus.sh
tgabriel@tgabriel:~$ ./proplus.sh
Enter a number
5
number greater than 0
tgabriel@tgabriel:~$ ./proplus.sh
Enter a number
-5
number less than 0
tgabriel@tgabriel:~$ ./proplus.sh
Enter a number
0
number equal to 0

```

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). Для данной задачи я создал файл: template.sh (Рисунки 9,10) и написал соответствующий скрипт.

```

tgabriel@tgabriel:~$ touch template.sh
tgabriel@tgabriel:~$ emacs

```

```

#!/bin/bash
opt=$1;
format=$2;
number=$3;
function template()
{
    for (( i=1; i<=$number; i++ )) do
        file=$(echo $format | tr '#' "$i")
        if [ $opt == "-r" ]
        then
            rm -f $file
        elif [ $opt == "-c" ]
        then
            touch $file
        fi
    done
}
template

```



Далее я проверил работу написанного скрипта (команда «./template.sh»), предварительно добавив право на исполнение файла (команда «chmod +x template.sh»). Сначала я создал три файла (команда «./files.sh -c job#.txt 3»), удовлетворяющие условию задачи, а потом удалил их (команда «./template.sh -r job#.txt 3») (Рисунок 11,12)

```

tgabriel@tgabriel:~$ chmod +x template.sh
tgabriel@tgabriel:~$ ./template.sh -c job#.txt 3
tgabriel@tgabriel:~$ ls
abc1      extensions.sh  job3.txt      proplus.sh
australia extensions.sh~ lab07.sh      proplus.sh~
backup    extentions.sh lab07.sh~     Public
backup.sh feathers     may           reports
backup.sh~ file1.pdf     monthly      ski.plases
conf.txt  file2.doc     Music        snap
Desktop  file2.sh      my_os        template.sh
Documents file2.sh~     Pictures     template.sh~
Downloads file3.jpg     play        tmp1.txt
ex1.txt  filels.sh     program12.sh tmp2.txt
ex2.txt  filels.sh~    program12.sh~ Videos
ex3.txt  file.txt      proplus      work
ex4.txt  job1.txt      proplus.c
exit     job2.txt      proplus.c~

```

```

tgabriel@tgabriel:~$ ./template.sh -r job#.txt 3
tgabriel@tgabriel:~$ ls
abc1      exit          lab07.sh      proplus.sh
australia extensions.sh  lab07.sh~     proplus.sh~
backup    extensions.sh~ may           Public
backup.sh extentions.sh monthly      reports
backup.sh~ feathers     Music        ski.plases
conf.txt  file1.pdf     my_os        snap
Desktop  file2.doc     Pictures     template.sh
Documents file2.sh      play        template.sh~
Downloads file2.sh~     program12.sh tmp1.txt
ex1.txt  file3.jpg     program12.sh~ tmp2.txt
ex2.txt  filels.sh     proplus      Videos
ex3.txt  filels.sh~    proplus.c    work
ex4.txt  file.txt      proplus.c~

```

- Написал командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировал его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find). Для данной задачи я создал файл: pro.sh и написал соответствующий скрипт.(Рисунок 13)

```
#!/bin/bash
template=$(find ./ -maxdepth 1 -mtime -7)
listing=""
for file in "$template" ; do
    file=$(echo "$file" | cut -c 3-)
    listing="$listing $file"
done
dir=$(basename $(pwd))
tar -cvf $dir.tar $listing
```

Далее я проверил работу написанного скрипта (команды «sudo ~/pro.sh» и «tar -tf tgabriel.tar»), предварительно добавив право на исполнение файла (команда «chmod +x pro.sh») и создав отдельный tgabriel с несколькими файлами. Как видно из Рисунков 14,15,16,17,18, файлы, измененные более недели назад, заархивированы не были.

```
tgabriel@tgabriel:~$ touch pro.sh
tgabriel@tgabriel:~$ emacs
tgabriel@tgabriel:~$ chmod +x pro.sh
```

```
tgabriel@tgabriel:~$ ls -l
total 156
-rw-rw-r-- 1 tgabriel tgabriel    0 мая 15 10:21 abc1
drwxr--r-- 2 tgabriel tgabriel  4096 мая 15 11:28 australia
drwxrwxr-x 2 tgabriel tgabriel  4096 мая 28 05:24 backup
-rwxrwxr-x 1 tgabriel tgabriel    99 мая 28 05:10 backup.sh
-rw-rw-r-- 1 tgabriel tgabriel    0 мая 28 05:07 backup.sh~
-rw-rw-r-- 1 tgabriel tgabriel   436 мая 15 14:03 conf.txt
drwxr-xr-x 2 tgabriel tgabriel  4096 мая 15 06:45 Desktop
drwxr-xr-x 2 tgabriel tgabriel  4096 мая 15 03:06 Documents
drwxr-xr-x 2 tgabriel tgabriel  4096 мая 15 03:06 Downloads
-rw-rw-r-- 1 tgabriel tgabriel    0 мая 22 01:50 ex1.txt
-rw-rw-r-- 1 tgabriel tgabriel    0 мая 22 01:50 ex2.txt
-rw-rw-r-- 1 tgabriel tgabriel    0 мая 22 01:51 ex3.txt
-rw-rw-r-- 1 tgabriel tgabriel    0 мая 22 01:51 ex4.txt
-rw-rw-r-- 1 tgabriel tgabriel   210 мая 21 21:30 exit
-rwxrwxr-x 1 tgabriel tgabriel  1304 мая 28 06:04 extensions.sh
-rwxrwxr-x 1 tgabriel tgabriel    0 мая 28 05:38 extensions.sh~
```

```
tgabriel@tgabriel:~$ sudo ~/pro.sh
```



```
.cache/libgweather/  
program12.sh  
extensions.sh~  
extensions.sh  
proplus.c  
file2.sh  
pro.sh~  
.bash_history  
file1.pdf  
.gnupg/  
.gnupg/trustdb.gpg  
.gnupg/pubring.kbx  
.vboxclient-clipboard.pid  
template.sh~  
.vboxclient-seamless.pid  
proplus.c~  
backup.sh  
backup/  
backup/backup.sh.bz2  
template.sh  
tgabriel@tgabriel:~$ tar -tf tgabriel.tar
```

```
.cache/libgweather/  
program12.sh  
extensions.sh~  
extensions.sh  
proplus.c  
file2.sh  
pro.sh~  
.bash_history  
file1.pdf  
.gnupg/  
.gnupg/trustdb.gpg  
.gnupg/pubring.kbx  
.vboxclient-clipboard.pid  
template.sh~  
.vboxclient-seamless.pid  
proplus.c~  
backup.sh  
backup/  
backup/backup.sh.bz2  
template.sh  
tgabriel@tgabriel:~$
```

## Вывод:

В ходе выполнения данной лабораторной работы я изучил основы программирования в оболочке ОС UNIX и научился писать более сложные

командные файлы с использованием логических управляющих конструкций и циклов.

### Контрольные вопросы:

1. Команда `getopts` 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, для команды `ls` флагом может являться `-F`. Строка опций `option-string` – это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за символом, обозначающим этот флаг, должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, то она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введенные данные с помощью оператора `case`. Функция `getopts` включает две специальные переменные среды – `OPTARG` и `OPTIND`. Если ожидается дополнительное значение, то `OPTARG` устанавливается в значение этого аргумента. Функция `getopts` также понимает переменные типа массив, следовательно, можно использовать её в функции не только для синтаксического анализа аргументов функций, но и для анализа введенных пользователем данных.
2. При перечислении имён файлов текущего каталога можно использовать следующие символы:
  - – – соответствует произвольной, в том числе и пустой строке;
  - ? – соответствует любому одинарному символу;
  - [c1-c2] – соответствует любому символу, лексикографически находящемуся между символами c1 и c2. Например,
  - `echo *` – выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`;
  - `ls *.c` – выведет все файлы с последними двумя символами, совпадающими с `.c`.
  - `echo prog.?` – выведет 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 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, являются операторами языка программирования `bash`.

Поэтому при описании языка программирования `bash` термин оператор будет использоваться наравне с термином команда. 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 команды заключается в выработке кода завершения.

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 перестаёт быть правильным. Команда `continue` используется в ситуациях, когда больше нет необходимости выполнять блок операторов, но вы можете захотеть продолжить проверять данный блок на других условных выражениях.
5. Следующие две команды ОС UNIX используются только совместно с управляющими конструкциями языка программирования `bash`: это команда `true`, которая всегда возвращает код завершения, равный нулю (т.е. истина), и команда `false`, которая всегда возвращает код завершения, не равный нулю (т.е. ложь). Примеры бесконечных циклов: `while true do echo hello andy done` `until false do echo hello mike done`
6. Строка `if test -f mans/i.s` проверяет, существует ли файл `mans/i.s` и является ли этот файл обычным файлом. Если данный файл является каталогом, то команда вернет нулевое значение (ложь).
7. Выполнение оператора цикла `while` сводится к тому, что сначала выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, а затем, если последняя выполненная команда из этой последовательности команд возвращает нулевой код завершения (истина), выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `do`, после чего осуществляется безусловный переход на начало оператора цикла `while`. Выход из цикла будет осуществлён тогда, когда последняя выполненная команда из последовательности команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, возвратит ненулевой код завершения (ложь). При замене в операторе цикла `while` служебного слова `while` на `until` условие, при выполнении которого осуществляется выход из цикла, меняется на противоположное. В остальном оператор цикла `while` и оператор цикла `until` идентичны.