

Отчёт по лабораторной работе № 11

дисциплина: Операционные системы

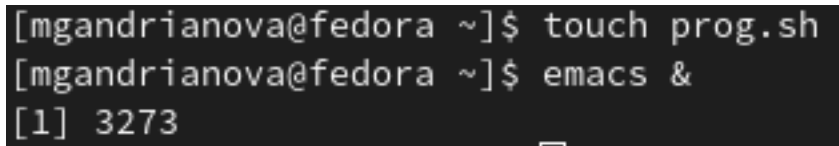
Андрианова Марина Георгиевна

Цель работы

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

Выполнение лабораторной работы

1). Используя команды `getopts` `grep`, написала командный файл, который анализирует командную строку с ключами: 1.- `-iinputfile` — прочитать данные из указанного файла; 2.- `-ooutputfile` — вывести данные в указанный файл; 3.- `-r` — шаблон — указать шаблон для поиска; 4.- `-C` — различать большие и малые буквы; 5.- `-n` — выдавать номера строки, а затем ищет в указанном файле нужные строки, определяемые ключом `-r`. Для данной задачи я создала файл `prog.sh` (рис.1) и написала соответствующий скрипт (рис.2).

A screenshot of a terminal window with a dark background and light-colored text. The prompt is `[mgandrianova@fedora ~]`. The first command entered is `$ touch prog.sh`. The second command is `$ emacs &`. The output of the second command is `[1] 3273`.

```
[mgandrianova@fedora ~]$ touch prog.sh
[mgandrianova@fedora ~]$ emacs &
[1] 3273
```

Рис.1:Создание файла

```
#!/bin/bash
iflag=0; oflag=0; pflag=0; Cflag=0; nflag=0; #Инициализация переменных-флагов, присваиваем им 0
while getopts i:o:p:C:n optletter #Анализируем командную строку на наличие опций
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)) #Проверка,указан ли шаблон для поиска
then echo "Шаблон не найден"
else
    if (($iflag==0))
    then echo "Файл не найден"
    else
        if (($oflag==0))
        then if (($Cflag==0))
            then if (($nflag==0))
                then grep $pval $ival
                else grep -n $pval $ival
                fi
            else if (($nflag==1))
                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==1))
                then grep -i $pval $ival > $oval
                else grep -i -n $pval $ival > $oval
                fi
            fi
        fi
    fi
fi
```

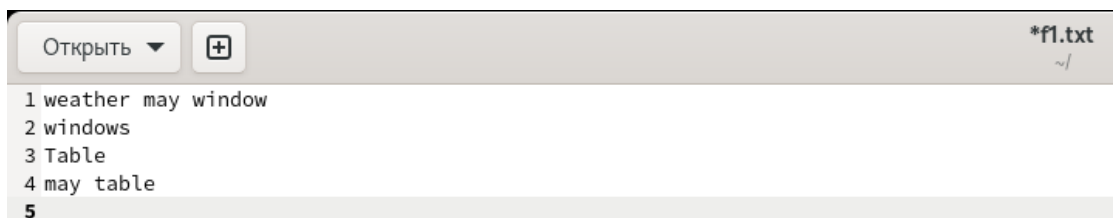
U:--- prog.sh Bot L31 (Shell-script[bash])

Рис.2:Скринш № 1

Для проверки работоспособности скрипта создала 2 файла, которые необходимы для выполнения программы: f1.txt и f2.txt и добавила право на исполнение файла(команда “chmod +x prog.sh”)(рис.3). Зашла в файл f1.txt и ввела там некоторый текст(рис.4). Проверила работу написанного скрипта, используя различные опции(рис.5). Скрипт работает корректно.

```
[mgandrianova@fedora ~]$ touch f1.txt f2.txt
[1]+  Завершён      emacs
[mgandrianova@fedora ~]$ chmod +x prog.sh
```

Рис.3:Создание файлов и предоставление прав доступа



```
1 weather may window
2 windows
3 Table
4 may table
5
```

Рис.4:Файл f1.txt

```
[mgandrianova@fedora ~]$ cat f1.txt
weather may window
windows
Table
may table

[mgandrianova@fedora ~]$ ./prog.sh -i f1.txt -o f2.txt -p may -n
[mgandrianova@fedora ~]$ cat f2.txt
1:weather may window
4:may table
[mgandrianova@fedora ~]$ ./prog.sh -i f1.txt -o f2.txt -p may -C -n
[mgandrianova@fedora ~]$ cat f2.txt
1:weather may window
4:may table
[mgandrianova@fedora ~]$ ./prog.sh -i f1.txt -o f2.txt -p Table -C -n
[mgandrianova@fedora ~]$ cat f2.txt
3:Table
4:may table
[mgandrianova@fedora ~]$ ./prog.sh -i f1.txt -C -n
Шаблон не найден
[mgandrianova@fedora ~]$ ./prog.sh -o f2.txt -p window -C -n
Файл не найден
```

Рис.5:Проверка работы программы

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 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 создала 2 файла: `number.c` и `number.sh`(рис.6) и написала для них скрипты(рис.7,рис.8).

```
[mgandrianova@fedora ~]$ touch number.c
[mgandrianova@fedora ~]$ touch number.sh
[mgandrianova@fedora ~]$ emacs &
[1] 4727
```

Рис.6:Создание файлов

```
#include <stdio.h>
#include <stdlib.h>
int main ()
{
    printf ("Введите число\n");
    int a;
    scanf ("%d", &a);
    if (a<0) exit(0);
    if (a>0) exit(1);
    if (a==0) exit(2);
    return 0;
}
```

```
U:--- number.c      All L13      (C/*l Abbrev)
Wrote /home/mgandrianova/number.c
```

Рис.7:Работа в файле number.c

```
#!/bin/bash
gcc number.c -o number
./number
code=$?
case $code in
    0) echo "Число меньше 0";;
    1) echo "Число больше 0";;
    2) echo "Число равно 0";;
esac
```

U:--- **number.sh** All L10 (Shell-script[bash])

Рис.8:Работа в файле number.sh

Проверила работу написанных скриптов(команда “./number.sh”), предварительно добавив право на исполнение файла(команда “chmod +x number.sh”)(рис.9). Скрипты работают корректно.

```
[mgandrianova@fedora ~]$ chmod +x number.sh
[1]+  Завершён      emacs
[mgandrianova@fedora ~]$ ./number.sh
Введите число
5
Число больше 0
[mgandrianova@fedora ~]$ ./number.sh
Введите число
0
Число равно 0
[mgandrianova@fedora ~]$ ./number.sh
Введите число
-8
Число меньше 0
```

Рис.9:Проверка скрипта № 2

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). Для данной задачи я создала файл-files.sh(рис.10) и написала в нём соответствующий скрипт(рис.11).

```
[mgandrianova@fedora ~]$ touch files.sh
[mgandrianova@fedora ~]$ emacs &
[1] 5328
```

Рис.10:Создание файла

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

```
-:--- files.sh All L19 (Shell-script[bash])
```

Рис.11:Скринш № 3

Затем я проверила работу написанного скрипта(команда “./files.sh”), предварительно добавив право на исполнение файла(команда “chmod +x files.sh”).Я создала 4 файла(команда “./files.sh -c may#.txt 4”),удовлетворяющие условию задачи, а потом удалила их(команда “./files.sh -r may#.txt 4”).Скрипт работает корректно(рис.12).


```
[mgandrianova@fedora ~]$ chmod +x files.sh
[1]+  Завершён      emacs
[mgandrianova@fedora ~]$ ./files.sh -c may#.txt 4
[mgandrianova@fedora ~]$ ls
'##abc1##'  bin          files.sh      go1.18.1.linux-amd64.tar.gz  may3.txt  number.sh  teamfile.sh  work1.txt~  work4.txt~  Изображения.zip
'##abc1#'  f1.txt       files.sh~    lab07.sh      may4.txt  number.sh~  teamfile.sh~  work2.txt~  'Без имени 1'  лаба4.odp
abc1       f2.txt       findformat.sh  lab07.sh~    monthly   prog.sh     tutorial      work2.txt~  Видео         Музыка
backup     feathers     findformat.sh~  may          number     prog.sh~    work         work2.txt~  Документы     Общедоступные
backup.sh  files.sh     go            may1.txt     number.c   reports     '#work1.txt#'  work3.txt~  Загрузки      'Рабочий стол'
backup.sh~  files.sh~    go            may2.txt     number.c~  ski.places  work1.txt     work3.txt~  Изображения  Шаблоны

[mgandrianova@fedora ~]$ ./files.sh -c #.tmp 4
./files.sh: строка 7: ((: i<=: синтаксическая ошибка: ожидается операнд (неверный маркер «<=»)
[mgandrianova@fedora ~]$ ./files.sh -r may#.txt 4
[mgandrianova@fedora ~]$ ls
'##abc1##'  bin          files.sh      go1.18.1.linux-amd64.tar.gz  number.c   reports     '#work1.txt#'  work3.txt~  Загрузки      'Рабочий стол'
'##abc1#'  f1.txt       files.sh~    lab07.sh      number.c~  ski.places  work1.txt~    work4.txt~  Изображения  Шаблоны
abc1       f2.txt       findformat.sh  lab07.sh~    monthly   prog.sh     teamfile.sh   work1.txt~  лаба4.odp    Изображения.zip
backup     feathers     findformat.sh~  may          number.sh  teamfile.sh~  work2.txt~    'Без имени 1'  Видео         Музыка
backup.sh  files.sh     go            may1.txt     number.sh  prog.sh      tutorial      work2.txt~    Документы     Общедоступные
backup.sh~  files.sh~    go            number       prog.sh~    work         work3.txt~    Изображения  Шаблоны
```

Рис.12:Проверка работы скрипта № 3

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 (использовать команду find). Для данной задачи я создала файл prog1.sh(рис.13) и написала в нём скрипт(рис.14).

```
[mgandrianova@fedora ~]$ touch prog1.sh
[mgandrianova@fedora ~]$ emacs &
[1] 6144
```

Рис.13:Создание файла

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

```
U:--- prog1.sh All L11 (Shell-script[sh])
Wrote /home/mgandrianova/prog1.sh
```

Рис.14:Скринш № 4

Затем я проверила работу написанного скрипта(команда “tar -tf folder.tar”), предварительно добавив право на исполнение файла(команда “chmod +x prog1.sh”).Я создала каталог folder,добавила туда 3 файла: lab07.sh(изменено 16.05.2022), files.sh(изменено 27.05.2022), f1.txt(изменено 27.05.2022).Затем перешла в данный каталог. Как видно из рисунков 15 и 16, файлы,изменённые более недели назад,заархивированы не были. Скрипт работает корректно.

```

[mgandrianova@fedora ~]$ chmod +x prog1.sh
[mgandrianova@fedora ~]$ mkdir folder
[mgandrianova@fedora ~]$ cd folder
[mgandrianova@fedora folder]$ ls
f1.txt  files.sh  lab07.sh
[mgandrianova@fedora folder]$ ~/prog1.sh
files.sh
f1.txt
[mgandrianova@fedora folder]$ ~/prog1.sh
files.sh
f1.txt
tar: folder.tar: файл является архивом; не сброшен
[mgandrianova@fedora folder]$ tar -tf folder.tar
files.sh
f1.txt

```

Рис.15:Проверка скрипта № 4



Имя	Размер	Дата изменения
f1.txt	43 байта	12:29
files.sh	247 байт	12:54
folder.tar	10,2 кБ	13:03
lab07.sh	112 байт	16 мая

Рис.16:Результат после выполнения программы

Выводы

Я изучила основы программирования в оболочке ОС UNIX и научилась 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 введённые 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 введённых пользователем данных.

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

1.*–соответствует произвольной, в том числе и пустой строке;

2.?–соответствует любому одинарному символу;

3.[*c1-c2*] – соответствует любому символу, лексикографически находящемуся между символами *c1* и *c2*. Например, 1.1 `echo` – выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`; 1.2. `ls.c`–выведет все файлы с последними двумя символами, совпадающими с *c*. 1.3. `echo prog.?`–выведет все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются *prog.* 1.4.[*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 `OCUNIX` 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, неравный 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 man$s/$i.$s` проверяет, существует ли файл `man$s/$i.$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.