

Лабораторная работа №11. Программирование в командном процессоре ОС UNIX. Ветвления и циклы.

Парфенова Елизавета Евгеньевна

RUDN University, Moscow, Russian Federation

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:

- `-iinputfile` — прочитать данные из указанного файла;
- `-ooutputfile` — вывести данные в указанный файл;
- `-р`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 `-р`.

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)`.

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 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 работы

Первый командный файл

В первом задании было необходимо написать командный файл, который анализирует командную строку с ключами и ищет в указанном файле строки с ключом `r`. Осуществляем это с помощью циклов и команды `getopts`. Поиск выполняем командой `grep`. (рис. 1)

```
#!/bin/bash

while getopts "i:o:p:cn" opt
do
    case $opt in
        i) inputfile="$OPTARG";;
        o) outputfile="$OPTARG";;
        p) sample="$OPTARG";;
        c) reg="";;
        n) line="";;
    esac
done

grep -n "$sample" "$inputfile" > "$outputfile"
```

Figure 1: Код превого командого файла

Первый командный файл

После предоставляем право на выполнение командой ***chmod +x script1*** и запускаем, написав в консоль название скрипта и указав все данные: ***./script1 -i conf.txt -o result.txt -p n etconf -c -n***. В результате в папке создается файл с заданным названием и в него перезаписываются все файлы с указанными характеристиками. (рис. 2)

```
[eeeparfenova@fedora lab1]$ ./script1 -i conf.txt -o result.txt -p n etconf -c -n
[eeeparfenova@fedora lab1]$ ls
c++.cpp  c++.cpp-  conf.txt  result.txt  script1  script1-
[eeeparfenova@fedora lab1]$ cat result.txt
1:anthy-unicode.conf
2:appstream.conf
3:asound.conf
4:brltty.conf
5:chrony.conf
6:dleyna-renderer-service.conf
7:dleyna-server-service.conf
8:dnsmasq.conf
9:dracut.conf
10:dracut.conf.d
11:extlinux.conf
12:fpriintd.conf
13:fuse.conf
14:host.conf
15:tdmapi.conf
16:jwhois.conf
17:kdump.conf
```

Figure 2: Работа превого командого файла

Приступаем к написанию второго командного файла. Для него пишем программу на с++, которая определяет, больше нуля, меньше нуля или равно нулю заданное число. (рис. 3)

```
#include <iostream>

using namespace std;

int main(int var_1, char *var_2[])
{
    if (atoi(var_2[1])>0) exit(1);
    else if (atoi(var_2[1])==0) exit(2);
    else exit(3);

    return 0;
}
```

Figure 3: Код программы на C++

Второй командный файл

После создаем файл script2 и осуществляю программу с помощью циклов if. Также используем компилятор g++. (рис. 4)

```
#!/bin/bash

RES=result
LIN=c++.cpp

if [ "$LIN" -nt "$RES" ]
then
    echo "Creating $RES"
    g++ -o $RES $LIN
fi

./$RES $1

ec=$?

if [ "$ec" == "1" ]
then
    echo "Input number > 0"
fi

if [ "$ec" == "2" ]
then
    echo "Input number = 0"
fi

if [ "$ec" == "3" ]
then
    echo "Input number < 0"
fi
```

Figure 4: Код второго командного файла

После делаем файл исполняемым и запускаем его командой `./script2` записывая в качестве входных данных любое число. Программа работает успешно и для положительный чисел, и для отрицательных, и для чисел равных нулю. (рис. 5)

```
[eeparfenova@fedora lab11]$ ./script2 0
Creating result
Input number = 0
[eeparfenova@fedora lab11]$ emacs
[eeparfenova@fedora lab11]$ ./script2 3
Input number > 0
[eeparfenova@fedora lab11]$ ./script2 -1
Input number < 0
```

Figure 5: Работа второго командного файла

Начинаем писать третий скрипт. Создаем файл “script3”. Пишем код, используя команду getoptс и два цикла for. Выбираем формат файлов, у меня это tmp. (рис. 6)

```
#!/bin/bash
while getoptс c:r opt
do
    case $opt in
        c)n="$OPTARG"; for i in $(seq 1 $n); do touch "$i.tmp";done;;
        r) for i in $(find -name "*.tmp");do rm $i; done;;
        esac
    done
```


Figure 6: Код третьего командного файла

После делаем файл исполняемым и проверяем работу скрипта, используя `./script3` и одну из опций. Опцией `-s` создаем 5 файлов нужного формата (число вводим после опции) и проверяем их создание, а поцией `-r` удаляем эти файлы и снова проверяем успешность команды. Скрипт работает корректно. (рис. 7)

```
root@ferociousfedora lab111:~# ./script3 -s 5
root@ferociousfedora lab111:~# ls
1.tmp  2.tmp  3.tmp  4.tmp  5.tmp  conf.txt  result.txt  script1-  script2-  'script3-'  script3-
root@ferociousfedora lab111:~# ./script3 -r
root@ferociousfedora lab111:~# ls
1++.cpp  2++.cpp  3++.cpp  4++.cpp  5++.cpp  conf.txt  result.txt  script1-  script2-  'script3-'  script3-
root@ferociousfedora lab111:~#
```

Figure 7: Работа третьего командного файла

Приступаем к последнему заданию. В нем требовалось заархивировать файлы из указанного каталога, измененные не более, чем неделю назад. Архивируем с помощью tar. Снова используем getopt. Вот так выглядит код программы: (рис. 8)



```
#!/bin/bash

while getopt :d: opt
do
    case $opt in
        d)dir="$OPTARG";;
        esac
    done

    find $dir -mtime -7 -mtime +0 -type f > archiv.txt
    tar -cf result.tar -T archiv.txt
```

Figure 8: Код четвертого командного файла

Четвертый командный файл

Далее делаем файл исполняемым и запускаем скрипт командой `./script4`. В итоге создается архив с заданным названием, в котором находятся все файлы указанного каталога, которые были изменены в последнюю неделю. (рис. 9) (рис. 10)

```
sempar@semparFemur:~/1081118$ ./script4 -d /home/sempar/sempar
cp: /home/sempar/sempar: / не имеет дисков
cp: /home/sempar/sempar: / не имеет дисков
sempar@semparFemur:~/1081118$ ls
script4.txt  result.txt  script1  script2  script3  script4
c++ .cpp  result1  script1  script2  script3  script4
c++ .cpp  result1.txt  script1  script2  script3  script4
```

Figure 9: Работа четвертого командного файла



Figure 10: Содержимое созданного архива

Мы изучили основы программирования в оболочке ОС UNIX и научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.