Титульный лист

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Лабораторная работа 12

По дисциплине "Операционные системы"

Выполнил:

Студент группы НПВбм-01-19

Студенческий билет №: 1032193844

Саидов Ахият Магомадович

Руководитель: Валиева Татьяна Рефатовна

Москва 2023

Цель работы

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

Начало работы

1. Используя команды *getopts grep*, напишем командный файл, который анализирует командную строку с ключами:
   - *-iinputfile* — прочитать данные из указанного файла;
   - *-ooutputfile* — вывести данные в указанный файл;
   - *-p 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 *-p*.
   - Для этого создадим файл script1.sh, который будет в дальнейшем скрипт.

     ```
     [AMSaidov@amsaidov ~]$ mkdir lab12
     [AMSaidov@amsaidov ~]$ cd lab12
     [AMSaidov@amsaidov lab12]$ touch script1.sh
     ```

     Рисунок 1
   - Перейдем в наш файл *script1.sh* при помощи редактора `vi`.

     ```
     [AMSaidov@amsaidov lab12]$ vi script.sh
     ```

     Рисунок 2
   - Запишем код скрипта.

AMSaidov@amsaidov:~/lab12

Файл Правка Вид Поиск Терминал Справка

```
#!/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 illagel option $optletter
        esac
done
if (($pflag==0))
then echo "template not foung"
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))
                                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

~
                                                                7,1-8         Весь
```

Рисунок 3

- Разрешим управление для владельца.

```
[AMSaidov@amsaidov lab12]$ chmod +x script1.sh
```

Рисунок 4

- Запустим скрипт и проверим работу, но предварительно создадим 2 файла, куда запишем любой текст.

```
[AMSaidov@amsaidov lab12]$ touch text1.txt text2.txt
[AMSaidov@amsaidov lab12]$ vi text1.txt
[AMSaidov@amsaidov lab12]$ cat text1.txt
My name is Akhiyat
And from Moscow too like my friends.
I like to watch the animation "Смешарики".
[AMSaidov@amsaidov lab12]$ ./script1.sh -i text1.txt -o text2.txt -p like -C -n
[AMSaidov@amsaidov lab12]$ cat text2.txt
2:And from Moscow too like my friends.
3:I like to watch the animation "Смешарики".
[AMSaidov@amsaidov lab12]$ ./script1.sh -i text1.txt -C -n
template not foung
[AMSaidov@amsaidov lab12]$ ./script1.sh -i text1.txt -c -n
./script1.sh: недопустимая опция -- c
illagel option ?
template not foung
[AMSaidov@amsaidov lab12]$ ./script1.sh -o text2.txt -p like -C -n
file not found
[AMSaidov@amsaidov lab12]$
```

Рисунок 5

2. Напишем 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 этого создадим файлы script2.c и script2.sh, которые будут в дальнейшем скриптами.

     ```
     [AMSaidov@amsaidov lab12]$ touch script2.c script2.sh
     ```

     Рисунок 6

   - Перейдем в наш файл *script2.c* при помощи редактора `vi`, запишем код.

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

     Рисунок 7

   - Перейдем в наш файл _script2.sh при помощи редактора `vi`, запишем код скрипта.

     ```
     #!/bin/bash
     gcc script2.c -o script2
     ./script2
     code=$?
     case $code in
             0) echo "Номер меньше нуля";;
             1) echo "Номер больше нуля";;
             2) echo "Номер равен нулю";;
     esac
     ```

     Рисунок 8

   - Разрешим управление для владельца, запустим скрипт и проверим работу.

```
[AMSaidov@amsaidov lab12]$ chmod +x script2.sh
[AMSaidov@amsaidov lab12]$ ./script2.sh
Введите номер
12
Номер больше нуля
[AMSaidov@amsaidov lab12]$ ./script2.sh
Введите номер
-1
Номер меньше нуля
[AMSaidov@amsaidov lab12]$ ./script2.sh
Введите номер
0
Номер равен нулю
[AMSaidov@amsaidov lab12]$
```

Рисунок 9

3. Напишем 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 этого создадим файл script3.sh, который будет в дальнейшем скриптом.

     ```
     [AMSaidov@amsaidov lab12]$ touch script3.sh
     ```

     Рисунок 10

   - Перейдем в наш файл *script3.sh* при помощи редактора `vi`, запишем код.

     ```
     #!/bin/bash
     opt=$1;
     format=$2;
     number=$3;
     function script3()
     {
             for (( i=1; i<=$number; i++ )) do
                     file=$(echo $format | tr '#' "$i")
                     if [ $opt == "-r" ]
                     then
                             rm -f $file
                     elif [ $opt = "-c" ]
                     then
                             touch $file
                     fi
             done
     }
     script3
     ```

     Рисунок 11

   - Разрешим управление для владельца запустим скрипт и проверим работу.

     ```
     [AMSaidov@amsaidov lab12]$ chmod +x script3.sh
     [AMSaidov@amsaidov lab12]$ ./script3.sh -c script3#.txt 3
     [AMSaidov@amsaidov lab12]$ ls
     script1.sh  script2.c   script31.txt  script33.txt  text1.txt
     script2     script2.sh  script32.txt  script3.sh    text2.txt
     [AMSaidov@amsaidov lab12]$ ./script3.sh -r script3#.txt 3
     [AMSaidov@amsaidov lab12]$ ls
     script1.sh  script2  script2.c  script2.sh  script3.sh  text1.txt  text2.txt
     ```

     Рисунок 12

4. Напишем командный файл, который с помощью команды `tar` запаковывает в архив все файлы в указанной директории. Модифицируем его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду *find*).
   - Для этого создадим файл script4.sh, который будет в дальнейшем скриптом.

     ```
     [AMSaidov@amsaidov lab12]$ touch script4.sh
     ```

     Рисунок 13
   - Перейдем в наш файл *script4.sh* при помощи редактора `vi`, запишем код.

     ```
     #!/bin/bash
     script3=$(find ./ -maxdepth 1 -mtime -7)
     listing=""
     for file in "$script3" ; do
             file=$(echo "$file" | cut -c 3-)
             listing="$listing $file"
     done
     dir=$(basename $(pwd))
     tar -cvf $dir.tar $listing
     ```

     Рисунок 14
   - Разрешим управление для владельца запустим скрипт и проверим работу.

     ```
     [AMSaidov@amsaidov lab12]$ chmod +x script4.sh
     [AMSaidov@amsaidov lab12]$ ls -l
     итого 60
     -rw-r--r--. 1 root     root     20480 июн 18 18:22 lab12.tar
     -rwxrwxr-x. 1 AMSaidov AMSaidov   869 июн 18 17:21 script1.sh
     -rwxrwxr-x. 1 AMSaidov AMSaidov  8464 июн 18 17:58 script2
     -rw-rw-r--. 1 AMSaidov AMSaidov   191 июн 18 17:52 script2.c
     -rwxrwxr-x. 1 AMSaidov AMSaidov   210 июн 18 17:56 script2.sh
     -rwxrwxr-x. 1 AMSaidov AMSaidov   245 июн 18 18:13 script3.sh
     -rwxrwxr-x. 1 AMSaidov AMSaidov   207 июн 18 18:26 script4.sh
     -rw-rw-r--. 1 AMSaidov AMSaidov   108 июн 18 17:42 text1.txt
     -rw-rw-r--. 1 AMSaidov AMSaidov    93 июн 18 17:43 text2.txt
     [AMSaidov@amsaidov lab12]$ sudo ~/lab12/script4.sh
     text2.txt
     .#script1.sh
     script1.sh
     text1.txt
     script2.c
     script2.sh
     script2
     script3.sh
     tar: lab12.tar: файл является архивом; не сброшен
     script4.sh
     ```

     Рисунок 15

Вывод

Мы изучили основы программирования в оболочке ОС UNIX. Научились 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:

- ○ − 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.

1. 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.
2. 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.
3. 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

4. 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).

5. 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.