Отчёт по лабораторной работе №11

Дисциплина: Операционные системы

Горпинич Елена Михайловна

Содержание

Цель работы

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

Задачи

1. Изучите материал лабораторной работы.
2. Сделать отчёт по лабораторной работе №12 в формате Markdown.
3. 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. -p шаблон —указать шаблон для поиска;
4. -C—различать большие и малые буквы;
5. -n—выдавать номера строк,а затем ищет в указанном файле нужные строки, определяемые ключом –p. Для данной задачи я создала файл prog1.sh и написала соответствующие скрипты

```
emgorpinich@emgorpinich:~$ touch pr1.sh
emgorpinich@emgorpinich:~$ emacs &
```

```
#!/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 "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==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
```

Проверила работу написанного скрипта, используя различные опции, предварительно добавив право на исполнение файла и создав два файла, которые необходимы для выполнения программы: lab11_1.txt и lab11_2.txt. Скрипт работает исправно.

```
emgorpinich@emgorpinich:~$ chmod +x pr1.sh
emgorpinich@emgorpinich:~$ touch lab11_1.txt lab11_2.txt
emgorpinich@emgorpinich:~$ cat lab11_1.txt
lab11
stage1
abc abc
ok
l
emgorpinich@emgorpinich:~$ ./pr1.sh -i lab11_1.txt -o lab11_2.txt -p abc -n
emgorpinich@emgorpinich:~$ cat lab11_2.txt
3:abc abc
emgorpinich@emgorpinich:~$ ./pr1.sh -i lab11_1.txt -o lab11_2.txt -p abc -C -n
emgorpinich@emgorpinich:~$ cat lab11_2.txt
3:abc abc
emgorpinich@emgorpinich:~$ ./pr1.sh -i lab11_1.txt -C -n
Шаблон не найден
emgorpinich@emgorpinich:~$ ./pr1.sh -i lab11_2.txt -p abc -C -n
1:3:abc abc
emgorpinich@emgorpinich:~$ ./pr1.sh -o lab11_2.txt -p abc -C -n
Файл не найден
```

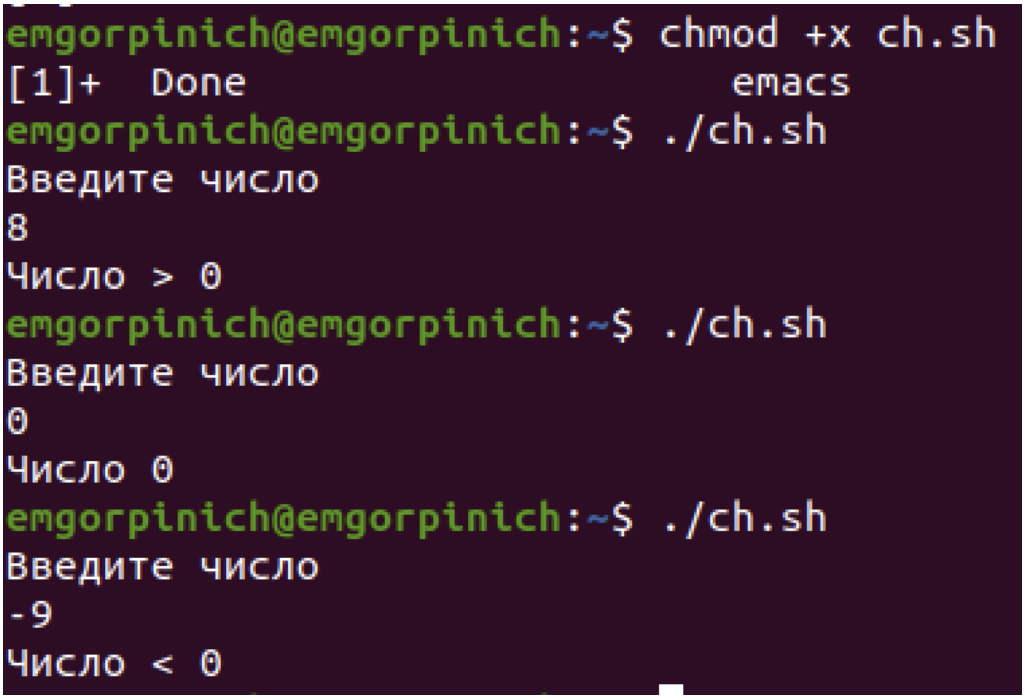
2) Написала на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции exit(n), передавая информацию о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды $?, выдать сообщение о том, какое число было введено. Для данной задачи я создала 2 файла: ch.c и ch.sh и написала соответствующие скрипты.

```
emgorpinich@emgorpinich:~$ touch ch.c ch.sh
emgorpinich@emgorpinich:~$ emacs &
```

```c
#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;
}
```

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

Проверила работу написанных скриптов , предварительно добавив право на исполнение файла. Скрипты работают верно.



3) Написала командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N. Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы. Для данной задачи я создала файл: file.sh и написала соответствующий скрипт.

```
emgorpinich@emgorpinich:~$ touch file.sh
emgorpinich@emgorpinich:~$ emacs &
```

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

Далее я проверила работу написанного скрипта, предварительно добавив право на исполнение файла. Сначала я создала несколько файлов, а потом удалила их

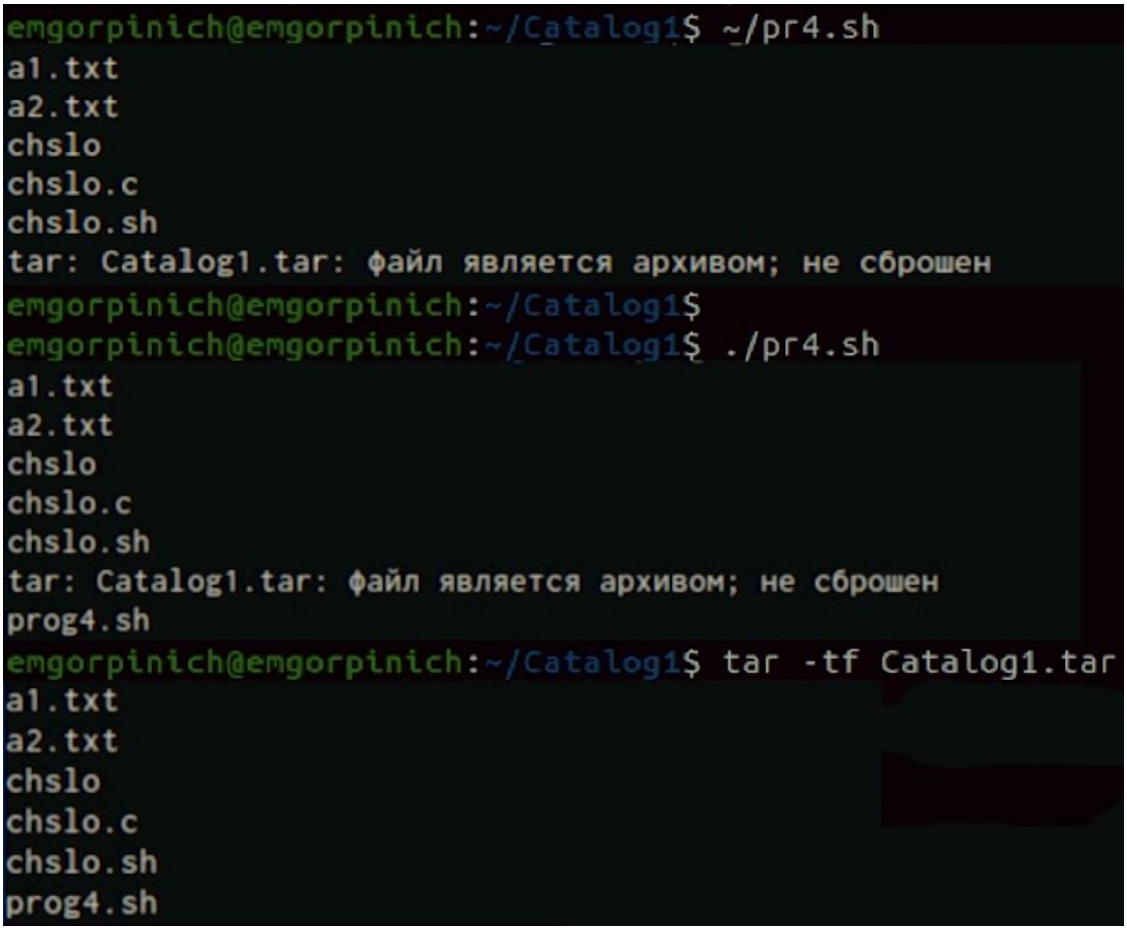
```
emgorpinich@emgorpinich:~$ chmod +x file.sh
[1]+  Done                    emacs
emgorpinich@emgorpinich:~$ ls
 abc          ch.sh~            example2.txt     '#lab07.sh#'   pr4.sh~
 abc1        '#chslo.sh#'      '#example3.txt#'   lab07.sh      prog1.sh~
 backup       chslo.sh~         example3.txt      Lab11         prog2.sh
 backup.sh    Desktop          '#example4.txt#'   lab11_1.txt   prog2.sh~
 backup.sh~   Documents         example4.txt      lab11_2.txt   progls.sh
 ch           Downloads         file.sh           Music         progls.sh~
 ch.c        '#example1.txt#'   file.sh~          Pictures      Public
 ch.c~        example1.txt      format.sh         pr1.sh        Templates
 ch.sh       '#example2.txt#'   format.sh~        pr4.sh        Videos
emgorpinich@emgorpinich:~$ ./file.sh -c l11_#.txt 5
emgorpinich@emgorpinich:~$ ls
 abc          '#chslo.sh#'      example3.txt      l11_4.txt     pr4.sh
 abc1         chslo.sh~        '#example4.txt#'   l11_5.txt     pr4.sh~
 backup       Desktop           example4.txt     '#lab07.sh#'   prog1.sh~
 backup.sh    Documents         file.sh           lab07.sh      prog2.sh
 backup.sh~   Downloads         file.sh~          Lab11         prog2.sh~
 ch          '#example1.txt#'   format.sh         lab11_1.txt   progls.sh
 ch.c         example1.txt      format.sh~        lab11_2.txt   progls.sh~
 ch.c~       '#example2.txt#'   l11_1.txt         Music         Public
 ch.sh        example2.txt      l11_2.txt         Pictures      Templates
 ch.sh~      '#example3.txt#'   l11_3.txt         pr1.sh        Videos
emgorpinich@emgorpinich:~$ ./file.sh -r l11_#.txt 5
emgorpinich@emgorpinich:~$ ls
 abc          ch.sh~            example2.txt     '#lab07.sh#'   pr4.sh~
 abc1        '#chslo.sh#'      '#example3.txt#'   lab07.sh      prog1.sh~
 backup       chslo.sh~         example3.txt      Lab11         prog2.sh
 backup.sh    Desktop          '#example4.txt#'   lab11_1.txt   prog2.sh~
 backup.sh~   Documents         example4.txt      lab11_2.txt   progls.sh
 ch           Downloads         file.sh           Music         progls.sh~
 ch.c        '#example1.txt#'   file.sh~          Pictures      Public
 ch.c~        example1.txt      format.sh         pr1.sh        Templates
 ch.sh       '#example2.txt#'   format.sh~        pr4.sh        Videos
```

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. Для данной задачи я создала файл: pr4.sh и написала соответствующий скрипт

```
emgorpinich@emgorpinich:~$ touch pr4.sh
emgorpinich@emgorpinich:~$ emacs &
```

```
#!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
```

Далее я проверила работу написанного скрипта, предварительно добавив право на исполнение файла и создав отдельный Catalog1 с несколькими файлами. Видно что файлы, измененные более недели назад, заархивированы не были. Скрипт работает корректно.



Вывод

В ходе выполнения данной лабораторной работы я изучила основы программирования в оболочке ОС UNIX/Linuxи научилась писать небольшие командные файлы.

Контрольные вопросы

1) Команда getopts осуществляет синтаксический анализ командной строки, выделяя флаги, ииспользуется 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 – эт осписок возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет со-провождаться некоторым аргументом, то за символом, обозначающим этот флаг, должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Еслик оманда getopts может распознать аргумент, то она возвра- щает 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 доплнительное 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) Приперечислении 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 со- бой простейший аналог команды ls; 1.2. ls.c–выведет все файлы с последними двумя символами, совпадающими с.с. 1.3. echoprog.?–выведет все файлы, состоя- щие 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 иwhile. С точки зрения командного процессора эти управ- ляющие 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 программирова- ния 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. Един- ственная 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 OCUNIX 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-fman*s*/i.*s*, *man*s/*i*.s и является ли этот файл обычным фай- лом.Если данный файл является каталогом,то команда вернет нулевое значение (ложь).

7) Выполнение оператора цикла while сводится к тому,что сначала выполня- ется 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 завер- шения(истина),выполняется последовательность команд(операторов),которую задаёт список-команд в строке,содержащей служебное слово do,после чего осу- ществляется безусловный переход на начало оператора цикла while.Выход из цикла будет осуществлён тогда,когда последняя выполненная команда из после- довательности команд (операторов),которую задаёт список-команд в строке,со- держащей 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 противопо- ложное.В остальном оператор цикла while и оператор цикла until идентичны.