Лабораторная работа №12

Российский университет дружбы народов

Тимур Андреевич Дарижапов

Содержание

1 Цель работы **5**

2 Выполнение лабораторной работы **6**

3 Выводы **17**

4 Ответы на контрольные вопросы **18**

Список таблиц

Список иллюстраций

2.1 Командный файл 7
2.2 Командный файл 8
2.3 Проверка 9
2.4 Командный файл 10
2.5 Командный файл 11
2.6 Проверка 12
2.7 Командный файл 13
2.8 Проверка 14
2.9 Проверка 14
2.10 Командный файл 15
2.11 Проверка 16

1 Цель работы

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

2 Выполнение лабораторной работы

1.Используя команды getopts grep, напишем командный файл, который анализирует командную строку с ключами: -i inputfile — прочитать данные из указанного файла; -o outputfile — вывести данные в указанный файл; -p шаблон — указать шаблон для поиска; -C — различать большие и малые буквы; -n — выдавать номера строк(Рисунок 2.1, 2.2).

```
#!/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
```

Рис. 2.1: Командный файл

```
                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
```

Рис. 2.2: Командный файл

Далее проверяем работу написанного скрипта, используя различные опции, предварительно добавив право на исполнение файла (команда «chmod +x prog1.sh») и создав 2 файла, которые необходимы для выполнения программы: a1.txt и a.txt. Скрипт работает корректно(Рисунок 2.3).

```
tadarizhapov@tadarizhapov-VirtualBox:~$ touch a.txt
tadarizhapov@tadarizhapov-VirtualBox:~$ chmod +x prog1.sh
tadarizhapov@tadarizhapov-VirtualBox:~$ cat a1.txt
I'm Timur
I'm a student of Peoples' Friendship University of Russia
tadarizhapov@tadarizhapov-VirtualBox:~$ ./prog1.sh -i a1.txt -o a.txt -p of -C -n
tadarizhapov@tadarizhapov-VirtualBox:~$ cat a.txt
2:I'm a student of Peoples' Friendship University of Russia
tadarizhapov@tadarizhapov-VirtualBox:~$ ./prog1.sh -i a1.txt -o a.txt -p of -n
tadarizhapov@tadarizhapov-VirtualBox:~$ cat a1.txt
I'm Timur
I'm a student of Peoples' Friendship University of Russia
tadarizhapov@tadarizhapov-VirtualBox:~$ cat a.txt
2:I'm a student of Peoples' Friendship University of Russia
tadarizhapov@tadarizhapov-VirtualBox:~$ ./prog1.sh -i a1.txt -C -n
Шаблон не найден
tadarizhapov@tadarizhapov-VirtualBox:~$ ./prog1.sh -o a.txt -p of -C -n
Файл не найден
tadarizhapov@tadarizhapov-VirtualBox:~$
```

Рис. 2.3: Проверка

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(Рисунок 2.4, 2.5).

```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;
}
```

Рис. 2.4: Командный файл

```
#!/bin/bash
gcc lab12-2.c -o lab12-2
./lab12-2
code=$?
case $code in
    0) echo "Меньше нуля";;
    1) echo "Больше нуля";;
    2) echo "Равно нулю";;
esac
```

Рис. 2.5: Командный файл

Далее проверяем работу написанных скриптов(команда «./lab12-2.sh»), предварительно добавив право на исполнение файла (команда «chmod +x lab12-2.sh»). Скрипты работают корректно(Рисунок 2.6).

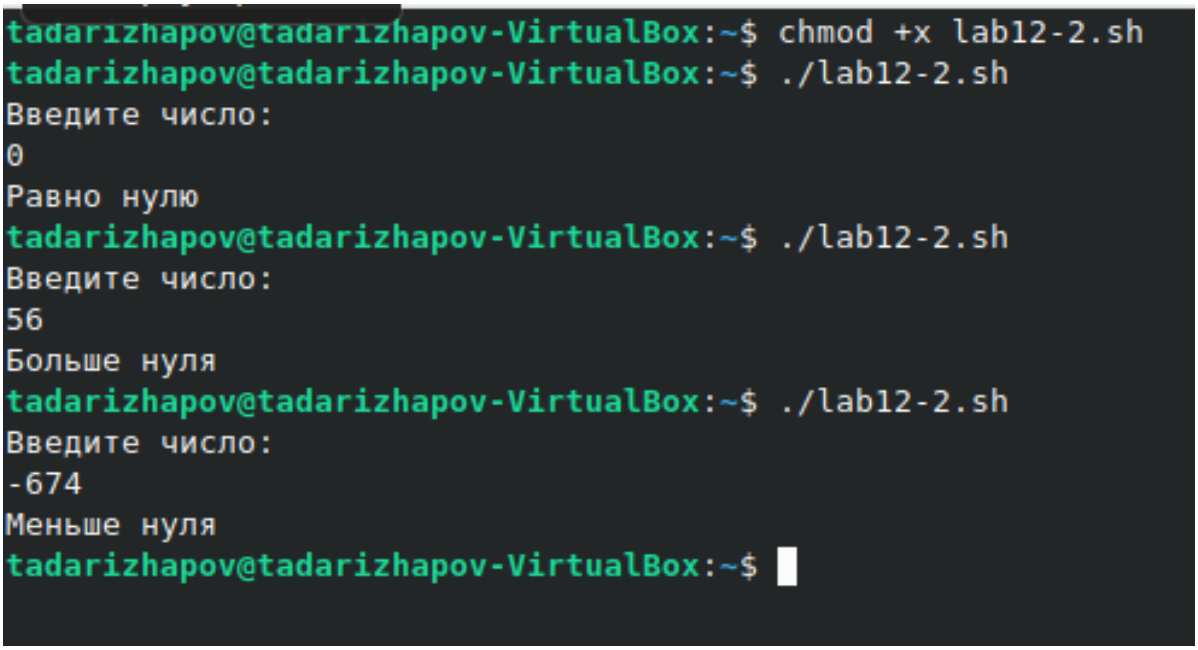

Рис. 2.6: Проверка

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)(Рисунок 2.7).

```
#!/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
```

Рис. 2.7: Командный файл

Далее проверяем работу написанного скрипта (команда «./lab12-3.sh»), предварительно добавив право на исполнение файла (команда «chmod +x lab12-3.sh»). Сначала я создала три файла (команда «./lab12-3.sh –c qwerty#.txt 3»), удовлетворяющие условию задачи, а потом удалила их(команда «./lab12-3.sh –r qwerty#.txt 4»)(Рисунок 2.8, 2.9).

```
tadarizhapov@tadarizhapov-VirtualBox:~$ chmod +x lab12-3.sh
tadarizhapov@tadarizhapov-VirtualBox:~$ ls
 1            lab11-4.sh             os.txt                          text.txt
 1.pdf        lab12-2                pandoc-crossref                 work
 2.doc        lab12-2.c              pandoc-crossref-Linux.tar.xz    Видео
 a1.txt       lab12-2.sh             prog1.sh                        Документы
 a.txt        lab12-3.sh             PTSerif-BoldItalic.ttf          Загрузки
 backup       laboratory             PTSerif-Bold.ttf                Изображения
 backup.sh    labs                   PTSerif-Italic.ttf              Музыка
 dist         LM_Sans_10             PTSerif-Regular.ttf             Общедоступные
 lab11-2.sh   lmsans10-regular.otf    q.log                          'Рабочий стол'
 lab11-3.sh   OFL.txt                snap                            Шаблоны
tadarizhapov@tadarizhapov-VirtualBox:~$ ./lab12-3.sh -c qwerty#.txt 4
tadarizhapov@tadarizhapov-VirtualBox:~$ ls
 1            lab12-2                pandoc-crossref-Linux.tar.xz    snap
 1.pdf        lab12-2.c              prog1.sh                        text.txt
 2.doc        lab12-2.sh             PTSerif-BoldItalic.ttf          work
 a1.txt       lab12-3.sh             PTSerif-Bold.ttf                Видео
 a.txt        laboratory             PTSerif-Italic.ttf              Документы
 backup       labs                   PTSerif-Regular.ttf             Загрузки
 backup.sh    LM_Sans_10             q.log                           Изображения
 dist         lmsans10-regular.otf   qwerty1.txt                     Музыка
 lab11-2.sh   OFL.txt                qwerty2.txt                     Общедоступные
 lab11-3.sh   os.txt                 qwerty3.txt                    'Рабочий стол'
 lab11-4.sh   pandoc-crossref        qwerty4.txt                     Шаблоны
```

Рис. 2.8: Проверка

```
tadarizhapov@tadarizhapov-VirtualBox:~$ ./lab12-3.sh -r qwerty#.txt 4
tadarizhapov@tadarizhapov-VirtualBox:~$ ls
 1            lab11-4.sh             os.txt                          text.txt
 1.pdf        lab12-2                pandoc-crossref                 work
 2.doc        lab12-2.c              pandoc-crossref-Linux.tar.xz    Видео
 a1.txt       lab12-2.sh             prog1.sh                        Документы
 a.txt        lab12-3.sh             PTSerif-BoldItalic.ttf          Загрузки
 backup       laboratory             PTSerif-Bold.ttf                Изображения
 backup.sh    labs                   PTSerif-Italic.ttf              Музыка
 dist         LM_Sans_10             PTSerif-Regular.ttf             Общедоступные
 lab11-2.sh   lmsans10-regular.otf    q.log                          'Рабочий стол'
 lab11-3.sh   OFL.txt                snap                            Шаблоны
tadarizhapov@tadarizhapov-VirtualBox:~$
```

Рис. 2.9: Проверка

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)(Рисунок 2.10).

```
#!/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
```

Рис. 2.10: Командный файл

Далее проверяем работу написанного скрипта (команды «sudo ~/lab12-4.sh» и «tar-tf 123.tar»), предварительно добавив право на исполнение файла (команда «chmod +x lab12-4.sh») и создав отдельный каталог 123 с несколькими файлами. Файлы, измененные более недели назад, заархивированы не были. Скрипт работает корректно(Рисунок 2.11).

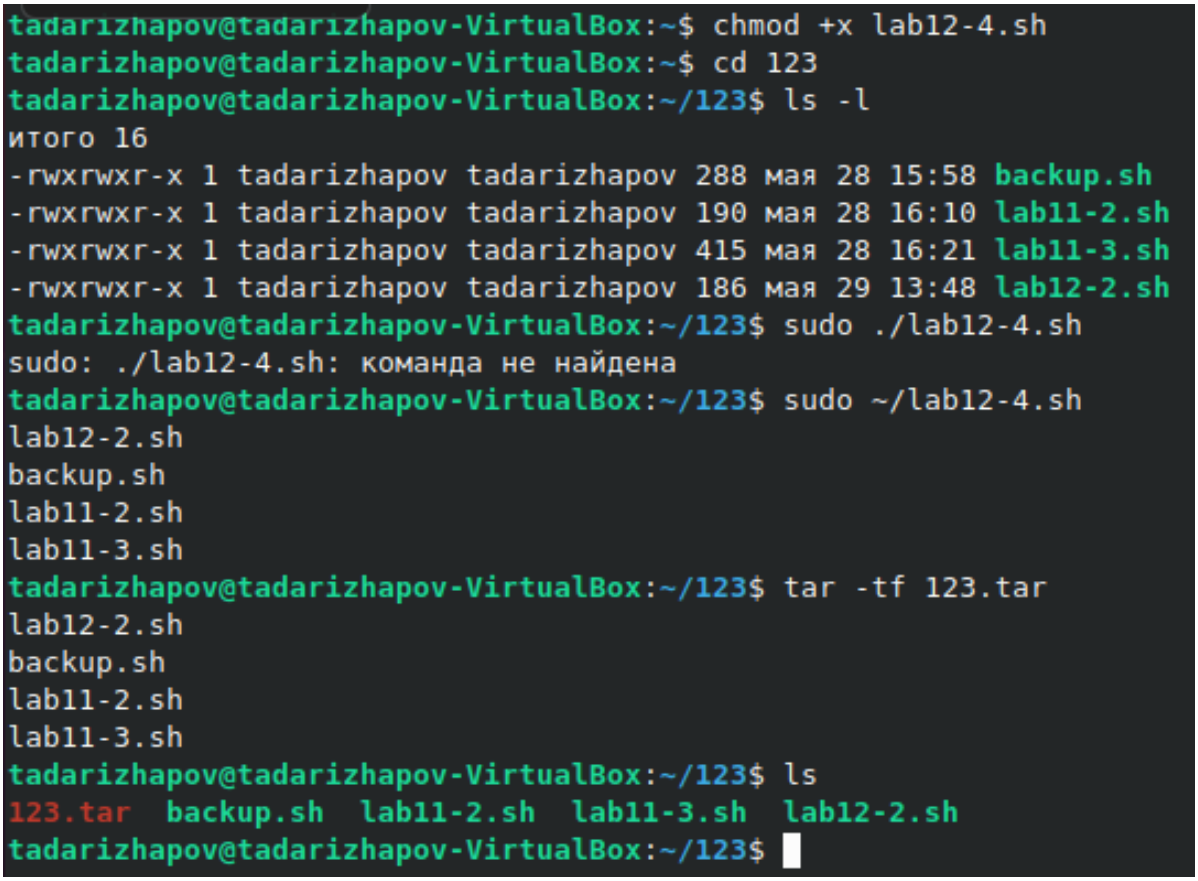

Рис. 2.11: Проверка

3 Выводы

Я изучаю основы программирования в оболочке ОС UNIX. Я научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

4 Ответы на 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.с. echoprog.?–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, посути, 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).

6)Строка if test -f man$s$/i.s, 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 последовательност ькоманд(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.