Отчёт по лабораторной работе №11

Операционные системы

Сячинова Ксения Ивановна

Содержание

Список иллюстраций

Список таблиц

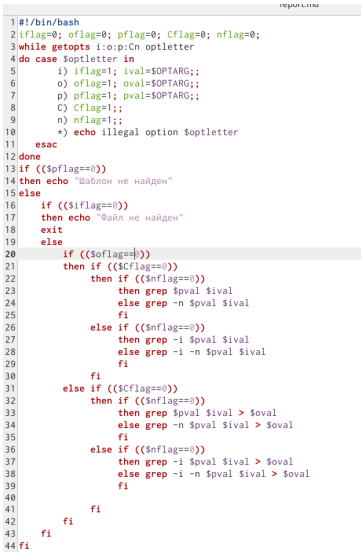
1 Цель работы

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

2 Выполнение лабораторной работы

1. Используя команды getopts grep,напишем командный файл, который анализирует командную строку с ключами:(рис. 2.1)

- -iinputfile — прочитать данные из указанного файла;
- -ooutputfile — вывести данные в указанный файл;
- -pшаблон—указать шаблон для поиска;
- -C — различать большие и малые буквы;
- -n—выдавать номера строк.



```
#!/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 (($iflag==0))
    then echo "Файл не найден"
    exit
    else
        if (($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
```

Рис. 2.1: Скрипт 1

После этого, проверяем работу написанного скрипта, используя различные опции. Но перед этим добавим право на исполнение файла “chmod +x os11.sh”.

Перед этим создадим два файла, в один из которых запишем текст. С ним и будем проверять выполнение программы. Используем разные команды “./os11.sh -i 1.txt -o 2.txt -p capital -C -n”,“./os11.sh -i 1.txt -o 2.txt -p capital -n” и т.д. (рис. 2.2)

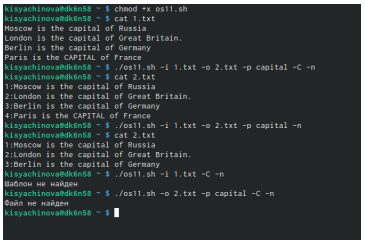


Рис. 2.2: Проверка работы

2. Напишем на языке Си программу, которая вводит число и определяет, являетсяли оно больше нуля, меньше нуля или равно нулю. Эта программа будет завершается с помощью функции exit(n), передавая информацию в о коде завершения в оболочку. Командный файл будет вызывать эту программу и, проанализировав с помощью команды “$?”, выдать сообщение о том, какое число было введено. Создадим два файла: chislo.c и chislo.sh.(рис. 2.3).(рис. 2.4)

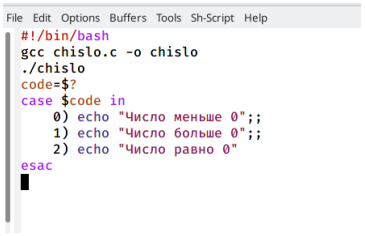


Рис. 2.3: Скрипт 2

```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: Скрипт 2.1

Проверим выполнение работы, даём разрешение. Всё работает верно. (рис. 2.5)

```
kisyachinova@dk6n58 ~ $ chmod +x chislo.sh
kisyachinova@dk6n58 ~ $ ./chislo.sh
Введите число
0
Число равно 0
kisyachinova@dk6n58 ~ $ ./chislo.sh
Введите число
7
Число больше 0
kisyachinova@dk6n58 ~ $ ./chislo.sh
Введите число
-7
Число меньше 0
kisyachinova@dk6n58 ~ $
```

Рис. 2.5: Проверка работы

3. Напишем командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до⊠ (например1.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). Создаём новый файл: fiels.sh.(рис. 2.6)

```
*report.m
#!/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. 2.6: Скрипт 3

Проверяем работу, Добавляем право на выполнение. Затем создаём три файла “./files.sh -c abc#.txt 3”, затем удаляем их с помощью команды “./files.sh -r abc#.txt 3”.(рис. 2.7)

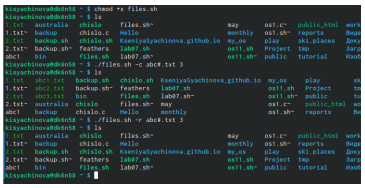


Рис. 2.7: Проверка работы

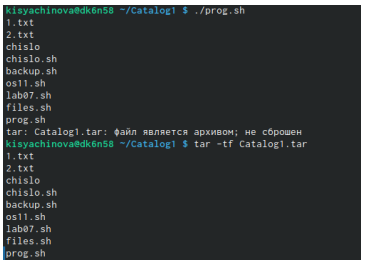
4. Напишем командный файл, который с помощью команды “tar” запаковывает в архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find). Создаю файл prog.sh и пишу скрипт.(рис. 2.8)

*report.md

```
#!/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. 2.8: Скрипт 4

Далее для проверки разрешаем право на исполнение и с помощью команд “./prog.sh”, “tar -tf Catalog1.tar” проверяем работу скрипта. Файлы, которые были измененны более недели назад, не были заархивированны. (рис. 2.9)



```
kisyachinova@dk6n58 ~/Catalog1 $ ./prog.sh
1.txt
2.txt
chislo
chislo.sh
backup.sh
os11.sh
lab07.sh
files.sh
prog.sh
tar: Catalog1.tar: файл является архивом; не сброшен
kisyachinova@dk6n58 ~/Catalog1 $ tar -tf Catalog1.tar
1.txt
2.txt
chislo
chislo.sh
backup.sh
os11.sh
lab07.sh
files.sh
prog.sh
```

Рис. 2.9: Проверка работы

3 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 .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.

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 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). 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, 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 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.

4 Выводы

В ходе выполнения данной лабораторной работы я изучила основы программирования в оболочке ОС UNIX и научилась писать более сложные командные фалйы с использованием логических конструкций и циклов.