Отчёт по одиннадцатой лабораторной работе

По дисциплине Операционные Системы

Плугатар Илья Михайлович

Цели работы:

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

Выполнение задания:

Используя команды getopts и grep, пишем командный файл, который анализирует командную строку с ключами: -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. а затем ищет в указанном файле нужные строки, определяемые ключом -p. В качестве основы используем конструкции while-do-done и case-in-esac. С помощью суммы ключей по степеням двойки делаем основную конструкцию case, где прописываем исполнение команд grep с разными опциями в зависимости от ключей командного файла.

```
#!/bin/bash
optkey=0
iflag=0
oflag=0
pflag=0
Cflag=0
nflag=0
while getopts i:o:p:Cn optlet
do case $optlet in
        i)   let "iflag = 1";    ival=$OPTARG;;
        o)   let "oflag = 1";    oval=$OPTARG;;
        p)   let "pflag = 1";    pval=$OPTARG;;
        C)   let "Cflag = 1";;
        n)   let "nflag = 1";;
        *)    echo nonexistent option $optlet
   esac

done
if [ $oflag == 1 ]
then
    let "optkey = optkey + 1"
fi
echo $optkey
if [ $Cflag == 1 ]
then
    let "optkey = optkey + 2"
fi
echo $optkey
if [ $nflag == 1 ]
then
    let "optkey = optkey + 4"
fi
echo $optkey
case $optkey in
       0) grep -i $pval $ival;;
       1) grep -i $pval $ival > $oval; grep -i $pval $ival;;
       2) grep $pval $ival;;
       3) grep $pval $ival > $oval; grep $pval $ival;;
       4) grep -i -n $pval $ival;;
       5) grep -i -n $pval $ival > $oval; grep -i -n $pval $ival;;
       6) grep -n $pval $ival;;
       7) grep -n $pval $ival > $oval; grep -n $pval $ival;;
esac
```

Скрипт для первого задания

Поиск текста без ключа -C и с ним

2. Пишем 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.

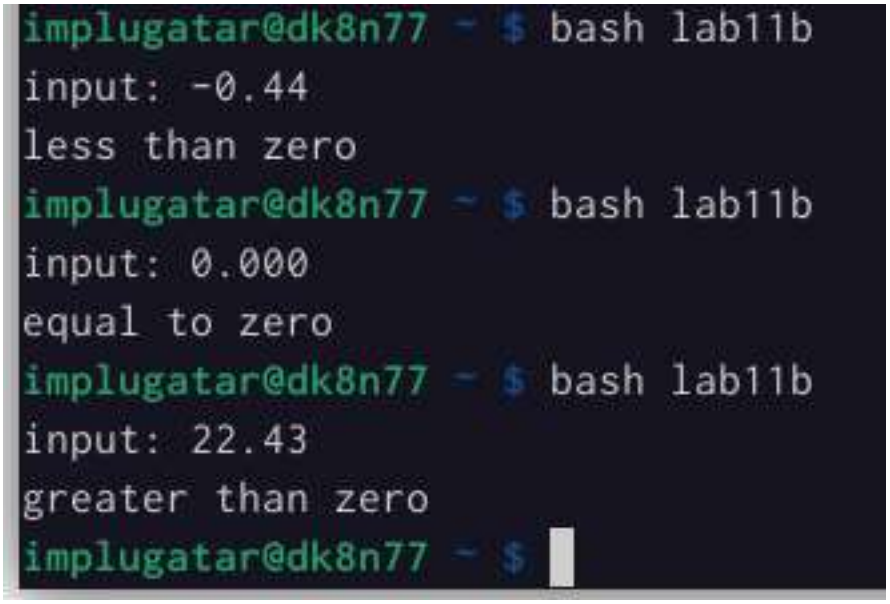
```
#!/bin/bash

gcc lab11b_prog.c -o lab11b_prog
./lab11b_prog
case $? in
0) echo "equal to zero";;
1) echo "less than zero";;
2) echo "greater than zero";;
esac
```

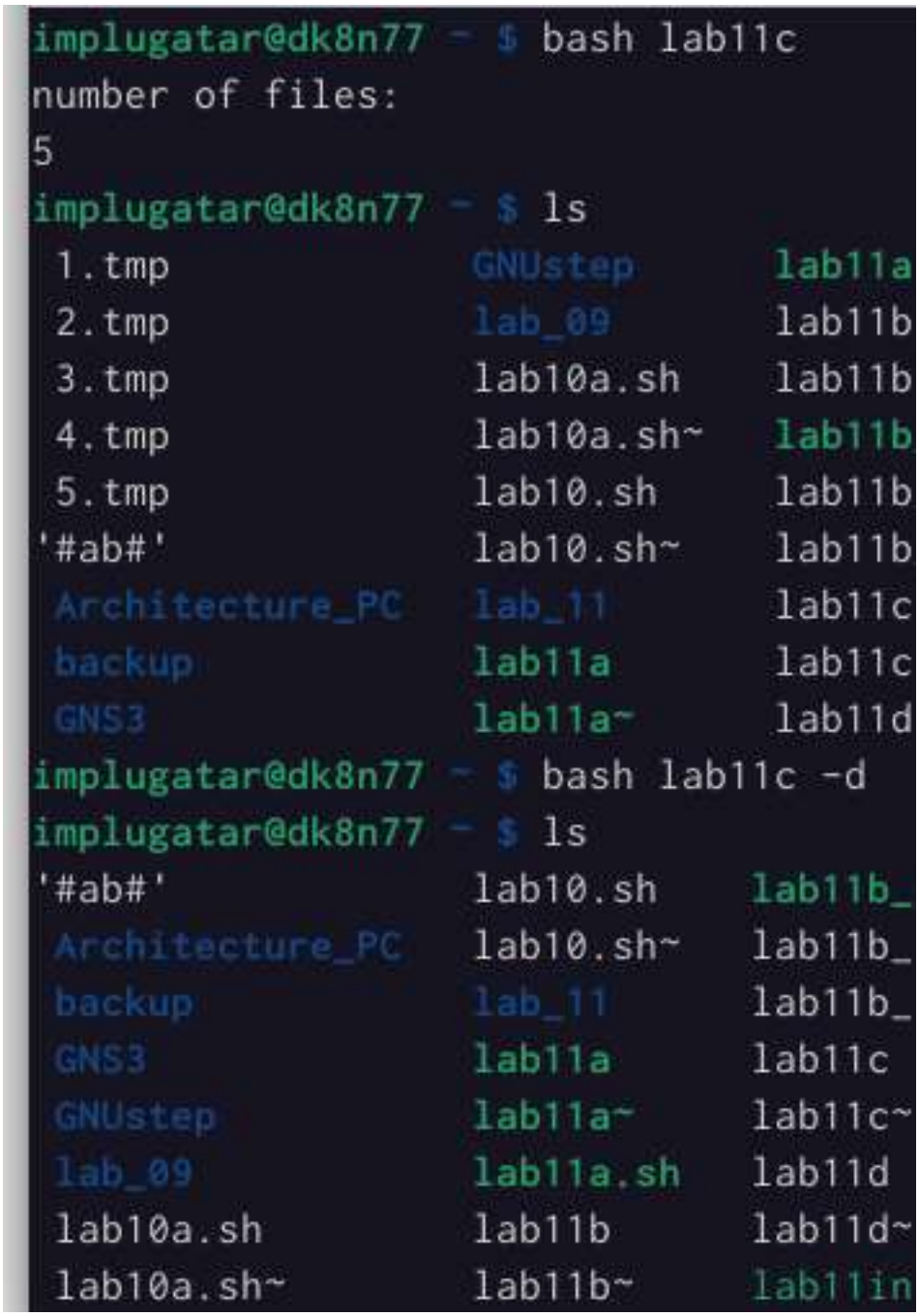
```c
#include <stdio.h>
#include <stdlib.h>

float num;

int main ()
{
  printf("input: ");
  scanf ("%f", &num);
  if (num == 0)
    {
      exit(0);
      printf("");
    }
  else
    {
    if (num <= 0)
    {
      exit(1);
      printf("");
    }
  else
    {
      exit(2);
      printf("");
    }
    }
  return (0);
}
```



3. Пишем 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 может удалять все созданные им файлы (если они существуют). Для создания и удаления файлов используем команду touch и команду rm соответственно.


implugatar@dk8n77 ~ $ bash lab11c
number of files:
5
implugatar@dk8n77 ~ $ ls
1.tmp GNUstep lab11a
2.tmp lab_09 lab11b
3.tmp lab10a.sh lab11b
4.tmp lab10a.sh~ lab11b
5.tmp lab10.sh lab11b
'#ab#' lab10.sh~ lab11b
Architecture_PC lab_11 lab11c
backup lab11a lab11c
GNS3 lab11a~ lab11d
implugatar@dk8n77 ~ $ bash lab11c -d
implugatar@dk8n77 ~ $ ls
'#ab#' lab10.sh lab11b_
Architecture_PC lab10.sh~ lab11b_
backup lab_11 lab11b_
GNS3 lab11a lab11c
GNUstep lab11a~ lab11c~
lab_09 lab11a.sh lab11d
lab10a.sh lab11b lab11d~
lab10a.sh~ lab11b~ lab11in

```
#!/bin/bash
dflag=0
while getopts d optlet
do
    case $optlet in
        d)let "dflag = 1";;
        *)echo "inavlid option"
    esac
done
if [ $dflag == 0 ]
then
echo "number of files: "
read filenum
let "filenum = filenum + 1"
name=1

while ((filenum-=1))
do
    touch $filenum.tmp
    let "name = name + 1"
done
else
    for file in ~/*.tmp
    do
        rm $file
    done
fi
```

4. Пишем 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). Итоговый скрипт оформляем как команду find с опциями -mtime, которая создаёт временное ограничение пространства поиска, и -exec, которая вызывает архиватор tar.

```
#!/bin/bash
echo "directory: "
read place
find ~/$place -type f -mtime -7 -exec tar -rf archd.tar {} \;
```

Скрипт для четвёртого задания

Заключение

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