

Отчет к лабораторной работе №11

Common information

discipline: Операционные системы

author: Бабина Юлия Олеговна

group: НПМбд-01-21

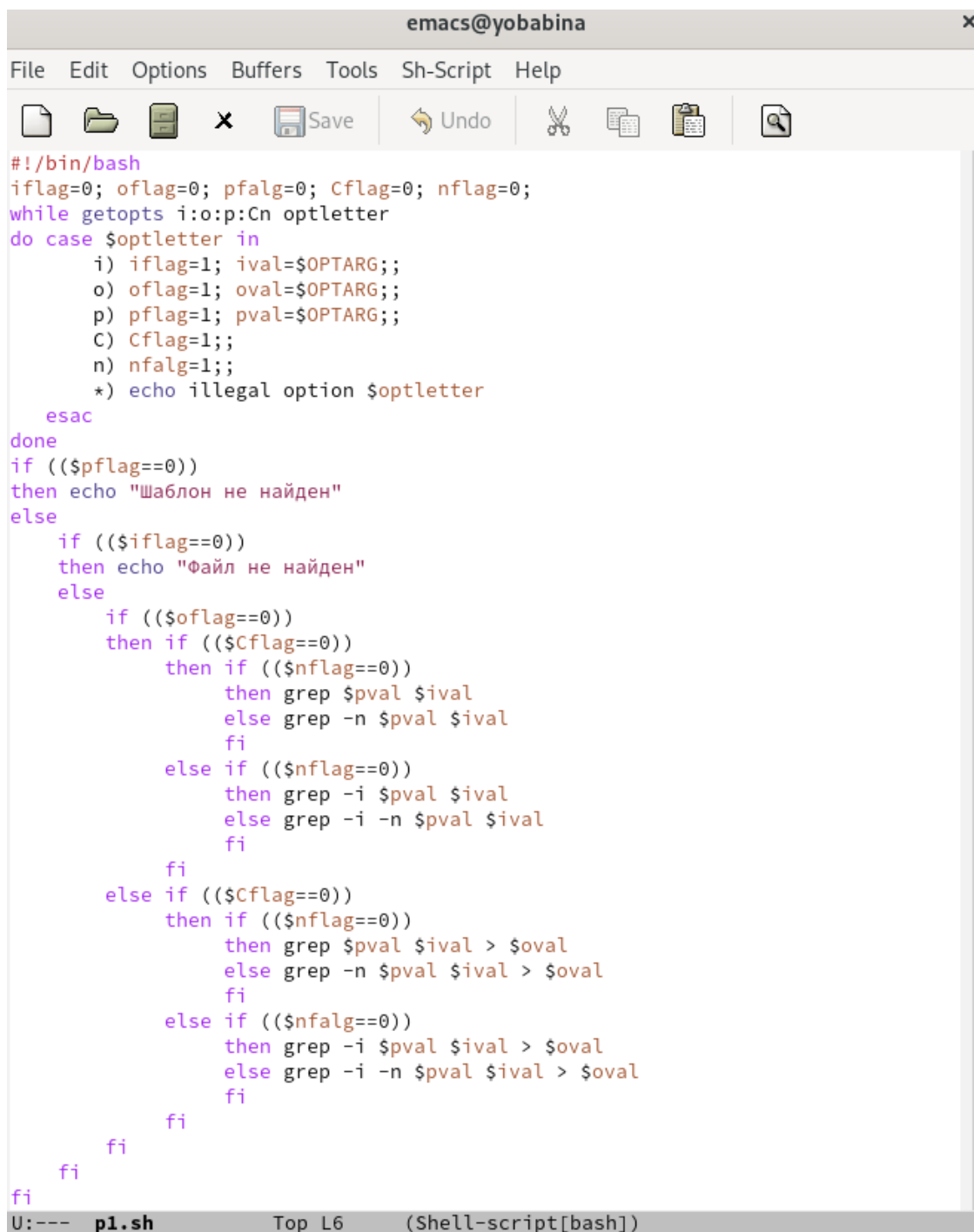
Цель работы

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

Ход работы

Создадим файл `p1.sh` и откроем его в `emacs`. Используя команды `getopts` `grep`, напишем в файле программу, которая анализирует командную строку с ключами: `-i` `inputfile` — прочитать данные из указанного файла; `-o` `outputfile` — вывести данные в указанный файл; `-r` `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 `-r`.



```
#!/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
fi
```

U:--- p1.sh Top L6 (Shell-script[bash])

код первой программы

Проверим корректность работы файла, дав ему права на выполнение, при помощи команды

```
chmod +x p1.sh
```

Далее создадим два файла для проверки работы one.txt и two.txt. Запустим файл.

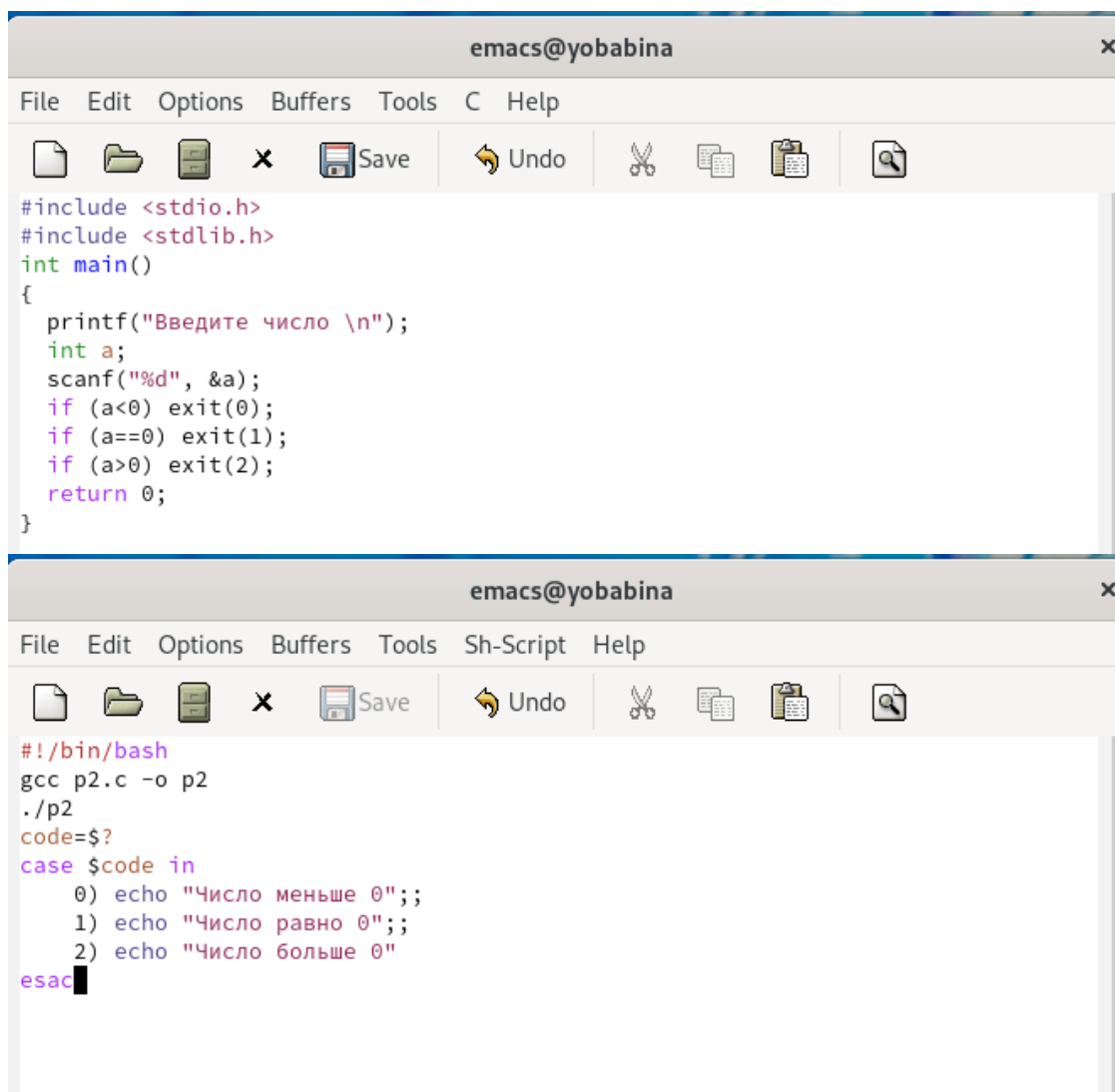
```
babinayuliaolegovna@yobabina:~  
[babinayuliaolegovna@yobabina ~]$ chmod +x p1.sh  
[babinayuliaolegovna@yobabina ~]$ touch one.txt  
[babinayuliaolegovna@yobabina ~]$ touch two.txt
```

права доступа

```
[babinayuliaolegovna@yobabina ~]$ ./p1.sh -i one.txt -o two.txt -p Изучить -C -n  
[babinayuliaolegovna@yobabina ~]$ cat one.txt  
Изучить основы программирования в оболочке ОС UNIX. Научится писать более  
сложные командные файлы с использованием логических управляющих конструкций  
и циклов.  
  
[babinayuliaolegovna@yobabina ~]$ cat two.txt  
1:Изучить основы программирования в оболочке ОС UNIX. Научится писать более  
[babinayuliaolegovna@yobabina ~]$
```

результат выполнения первой программы

Создадим два файла (p2.sh и p2.c) для задания номер два. Откроем в emacs. Далее 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.



Проверим корректность работы файла, дав ему права на выполнение и запустим его.

```
[babinayuliaolegovna@yobabina ~]$ chmod +x p2.sh
[babinayuliaolegovna@yobabina ~]$ ./p2.sh
Введите число
5
Число больше 0
[babinayuliaolegovna@yobabina ~]$
```

результат выполнения второй программы

Далее задание номер 3. Создадим файл `p3.sh` и откроем его в Emacs. 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)

```
emacs@yobabina
File Edit Options Buffers Tools Sh-Script Help
[Icons: File, Folder, Disk, X, Save, Undo, Cut, Copy, Paste, Find]

#!/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
```

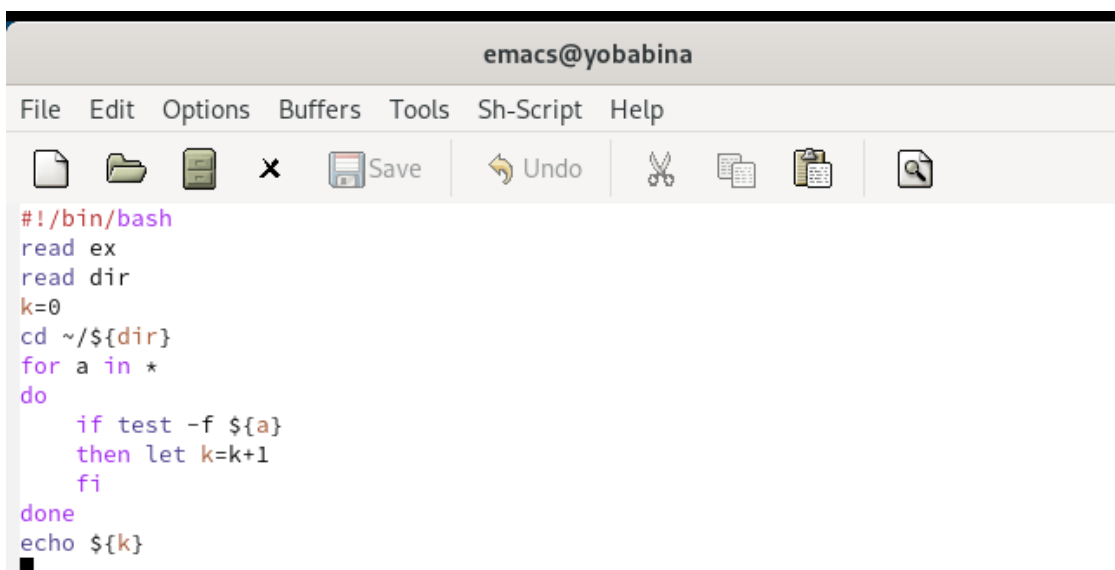
код третьей программы

Проверим корректность работы файла, дав ему права на выполнение и запустим его.

```
[babinayuliaolegovna@yobabina ~]$ chmod +x p3.sh
[babinayuliaolegovna@yobabina ~]$ ./p3.sh -c aaa#.txt 5
[babinayuliaolegovna@yobabina ~]$ ls
aaa1.txt  etc      monthly  p2.sh    s3.sh    Документы
aaa2.txt  feathers my_os    p3.sh    s3.sh~   Загрузки
aaa3.txt  file.txt newdir   p3.sh~   s4.sh    Изображения
aaa4.txt  games   one.txt  reports  s4.sh~   Музыка
aaa5.txt  june    p1.sh    s1.sh    script.cpp  Общедоступные
australia l9       p1.sh~   s1.sh~   two.txt   'Рабочий стол'
backup    lab09.sh p2       s2.sh    work      Шаблоны
conf.txt  lab09.sh~ p2.c     s2.sh~   yulia2
directory may       p2.c~    '#s3.sh#' Видео
[babinayuliaolegovna@yobabina ~]$ ./p3.sh -r aaa#.txt 5
[babinayuliaolegovna@yobabina ~]$ ls
australia games    my_os    p2.c~    s2.sh    script.cpp  Изображения
backup    june     newdir   p2.sh    s2.sh~   two.txt     Музыка
conf.txt  l9       one.txt  p3.sh    '#s3.sh#' work        Общедоступные
directory lab09.sh p1.sh    p3.sh~   s3.sh    yulia2     'Рабочий стол'
etc       lab09.sh~ p1.sh~   reports  s3.sh~   Видео      Шаблоны
feathers   may       p2       s1.sh    s4.sh    Документы
file.txt  monthly  p2.c     s1.sh~   s4.sh~   Загрузки
[babinayuliaolegovna@yobabina ~]$
```

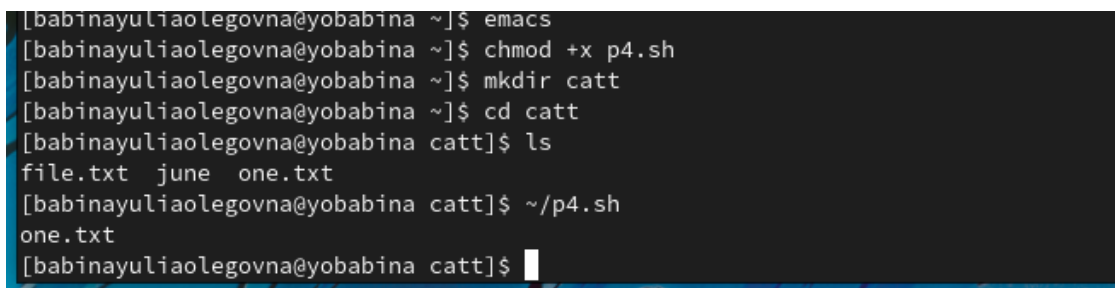
результат выполнения третьей программы

Создадим файл p4.sh для задания номер четыре и откроем в emacs. 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).



```
#!/bin/bash
read ex
read dir
k=0
cd ~/${dir}
for a in *
do
    if test -f ${a}
    then let k=k+1
    fi
done
echo ${k}
```

код четвертой программы



```
[babinayuliaolegovna@yobabina ~]$ emacs
[babinayuliaolegovna@yobabina ~]$ chmod +x p4.sh
[babinayuliaolegovna@yobabina ~]$ mkdir catt
[babinayuliaolegovna@yobabina ~]$ cd catt
[babinayuliaolegovna@yobabina catt]$ ls
file.txt  june  one.txt
[babinayuliaolegovna@yobabina catt]$ ./p4.sh
one.txt
[babinayuliaolegovna@yobabina catt]$
```

результат выполнения четвертой программы

Ответы на контрольные вопросы

Вопрос 1

Getopts - это встроенная команда оболочки Unix для анализа аргументов командной строки. Она предназначен для обработки аргументов командной строки, которые следуют рекомендациям синтаксиса утилиты POSIX, основанным на интерфейсе C getopt.

Вопрос 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, - 1.1 echo – выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды ls; - 1.2. ls.c–выведет все файлы с последними двумя символами, совпадающими с.c. - 1.3. echorprog.?–выведет все файлы, состоящие 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.

Вопрос 3

1) Точка с запятой (;)

Вы можете разместить две и более команд в одной и той же строке, разделив эти команды с помощью символа точки с запятой ;. Командная оболочка будет исследовать строку команды до момента достижения символа точки с запятой. Все аргументы перед этим символом точки с запятой будут рассматриваться как аргументы, не относящиеся к команде, находящейся после символа точки с запятой. Все команды с наборами аргументов будут выполнены последовательно, причем командная оболочка будет ожидать завершения исполнения каждой из команд перед исполнением следующей команды.

2) Амперсанд (&)

В том случае, если строка команды оканчивается символом амперсанда &, командная оболочка не будет ожидать завершения исполнения этой команды. Сразу же после ввода команды будет выведено новое приглашение командной оболочки, а сама команда будет исполняться в фоновом режиме. В момент завершения исполнения команды в фоновом режиме вы получите соответствующее сообщение.

3) Символ доллара со знаком вопроса (\$?)

Код завершения предыдущей команды сохраняется в переменной командной оболочки с именем \$? . На самом деле \$? является параметром командной оболочки, а не ее переменной, так как вы не можете присвоить значение переменной \$? .

4) Двойной амперсанд (&&)

Командная оболочка будет интерпретировать последовательность символов && как логический оператор “И”. При использовании оператора && вторая команда будет исполняться только в том случае, если исполнение первой команды успешно завершится (будет возвращен нулевой код завершения).

5) Двойная вертикальная черта (||)

Оператор || представляет логическую операцию “ИЛИ”. Вторая команда выполняется только тогда, когда исполнение первой команды заканчивается неудачей (возвращается ненулевой код завершения).

6) Знак фунта (#)

Все написанное после символа фунта (#) игнорируется командной оболочкой. Это обстоятельство оказывается полезным при возникновении необходимости в написании комментариев в сценариях командной оболочки, причем комментарии ни коим образом не будут влиять на процесс исполнения команд или процесс раскрытия команд командной оболочкой.

7) экранирование специальных символов (\)

Символ обратного слэша позволяет использовать управляющие символы без их интерпретации командной оболочкой; процедура добавления данного символа перед управляющими символами называется экранированием символов.

Вопрос 4

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.

Вопрос 5

`True` и `false` - это значения, которые может принять логическая переменная. По сути `true` и `false` эквивалентно да и нет.

Вопрос 6

Инструкция `if test -fman$s/$i.$s` проверяет, существует ли файл `mans/i.$s` и является ли этот объект обычным файлом.

Вопрос 7

Оператор `while` выполняет тело цикла, пока какое-то условие истинно, т.е. выражение или команда возвращают нулевой код. Оператор `until` наоборот, выполняет тело цикла, пока условие ложно, т.е. код возврата выражения или команды отличен от нуля.

Вывод

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