

## Отчёт по лабораторной работе №11

### Цель работы

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

## Ход выполнения работы

1. Написали командный файл, который анализирует командную строку с данными ключами, а затем ищет в указанном файле нужные строки, определенные ключом

```
pr1.sh [-M--] 0 L:[ 1+ 7 8/ 42] *(214 / 978b) 0032 0x020[*][X]
#!/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 (($Cflag==0))
1По~щъ 2Сох~ть 3Блок 4Замена 5Копия 6Пе~ть 7Поиск 8Уда~ть 9МенюMC10Выход
1По~щъ 2Сох~ть 3Блок 4Замена 5Копия 6Пе~ть 7Поиск 8Уда~ть 9МенюMC10Выход
        then if (($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
1По~щъ 2Сох~ть 3Блок 4Замена 5Копия 6Пе~ть 7Поиск 8Уда~ть 9МенюMC10Выход
```

2. Написали на языке Си программу, которая при вводе число выдает код в соответствии с числом(число меньше, равно или больше 0), а затем написали командный файл, который обращается к программе и выводит на экран информацию о числе

mc [iasyuzev@fedora]:~

pr2.cpp [----] 1 L: [ 1+11 12/ 12] \*(210 / 210b) <EOF> [\*] [X]

#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;  
}

1По~щъ 2Сох~ть 3Блок 4Замена 5Копия 6Пе~ть 7Поиск 8Уда~ть 9МенюМС10Выход

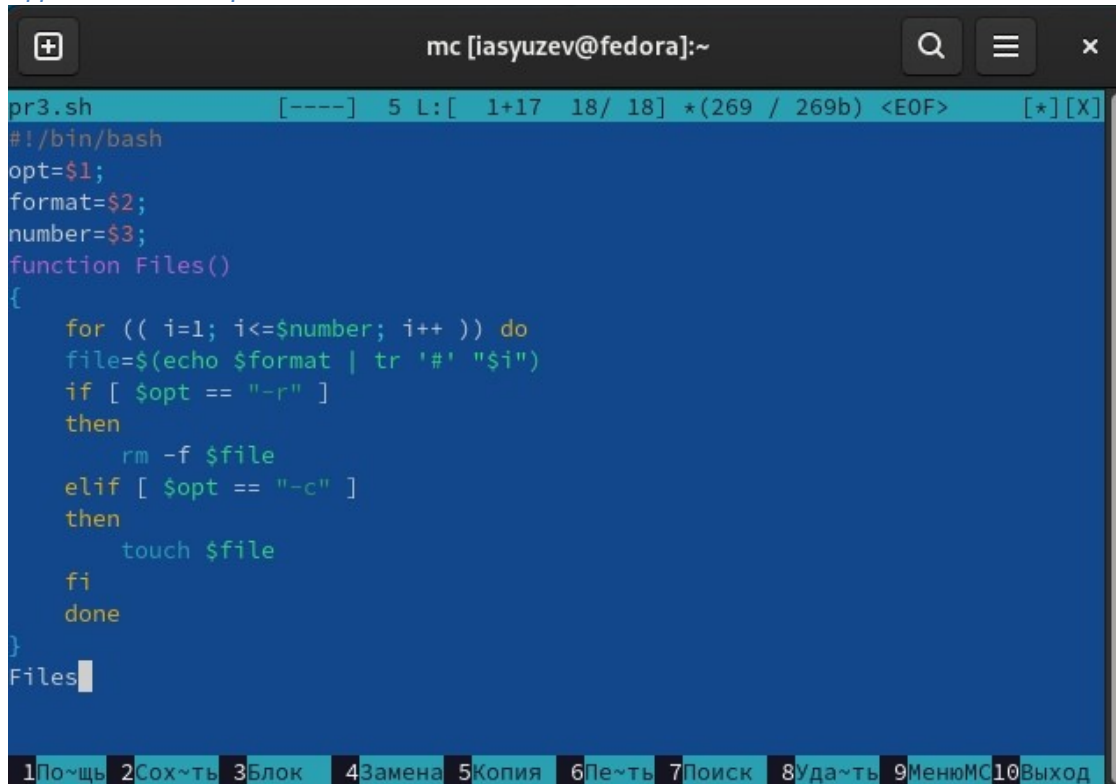
mc [iasyuzev@fedora]:~

pr2.sh [----] 4 L: [ 1+ 8 9/ 9] \*(187 / 187b) <EOF> [\*] [X]

#!/bin/bash  
gcc pr2.cpp -o pr2  
./pr2  
code=\$?  
case \$code in  
 0) echo "Число меньше 0";;  
 1) echo "Число равно 0";;  
 2) echo "Число больше 0";;  
esac

1По~щъ 2Сох~ть 3Блок 4Замена 5Копия 6Пе~ть 7Поиск 8Уда~ть 9МенюМС10Выход

3. Написали командный файл, создающий указанное число файлов или удаляющий ранее созданные этим файлом

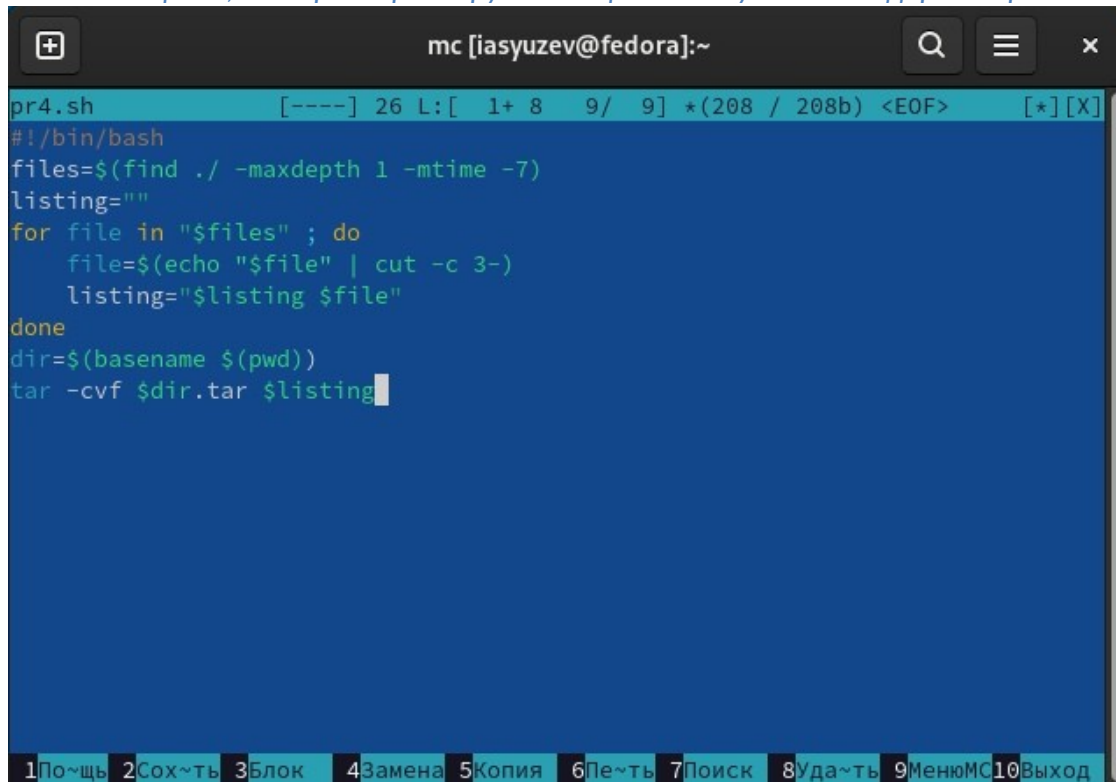


```
mc [iasyuzev@fedora]:~
pr3.sh [-----] 5 L:[ 1+17 18/ 18] *(269 / 269b) <EOF> [*] [X]
#!/bin/bash
opt=$1;
format=$2;
number=$3;
function Files()
{
    for (( i=1; i<=$number; i++ )) do
        file=$(echo $format | tr '#' "$i")
        if [ $opt == "-r" ]
        then
            rm -f $file
        elif [ $opt == "-c" ]
        then
            touch $file
        fi
    done
}
Files
```

1По~щъ 2Сох~ть 3Блок 4Замена 5Копия 6Ле~ть 7Поиск 8Уда~ть 9МенюМС10Выход

photo5

#### 4. Написали файл, который архивирует все файлы из указанной директории



```
mc [iasyuzev@fedora]:~
pr4.sh [-----] 26 L:[ 1+ 8 9/ 9] *(208 / 208b) <EOF> [*] [X]
#!/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
```

photo6

#### Выводы

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

#### 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 описок 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 введенные 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 getopt 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, 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. echo prog.? – 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. ##### 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, 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 -f mans/i.sn` проверяет, существует ли файл `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.