

Отчет к лабораторной работе №11

Common information

discipline: Операционные системы

group: НПМбд-01-21

author: Ермолаев А.М.

Цель работы

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

Выполнение работы

Используя команды `getopts` и `grep`, напомним 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.

а затем ищет в указанном файле нужные строки, определяемые ключом -p.

```
#!/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
            fi
        fi
    fi
fi
```

листинг первой программы

```
        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
```

листинг первой программы

Проверим корректность написанного файла.

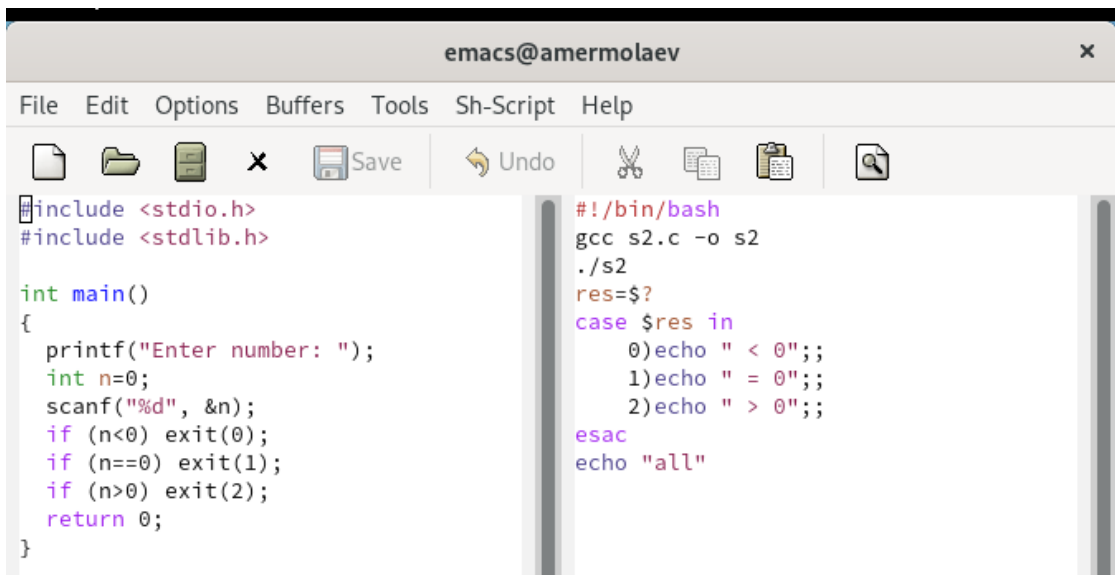
```
[amermolaev@amermolaev ~]$ ./s1.sh -i f1.txt -o f2.txt -p you -C -n
[amermolaev@amermolaev ~]$ cat f1.txt
hello
is
it
me
you
're
looking
for

never
gonna
give
you
up

[amermolaev@amermolaev ~]$ cat f2.txt
5:you
13:you
[amermolaev@amermolaev ~]$
```

работа первой программы

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 о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено.



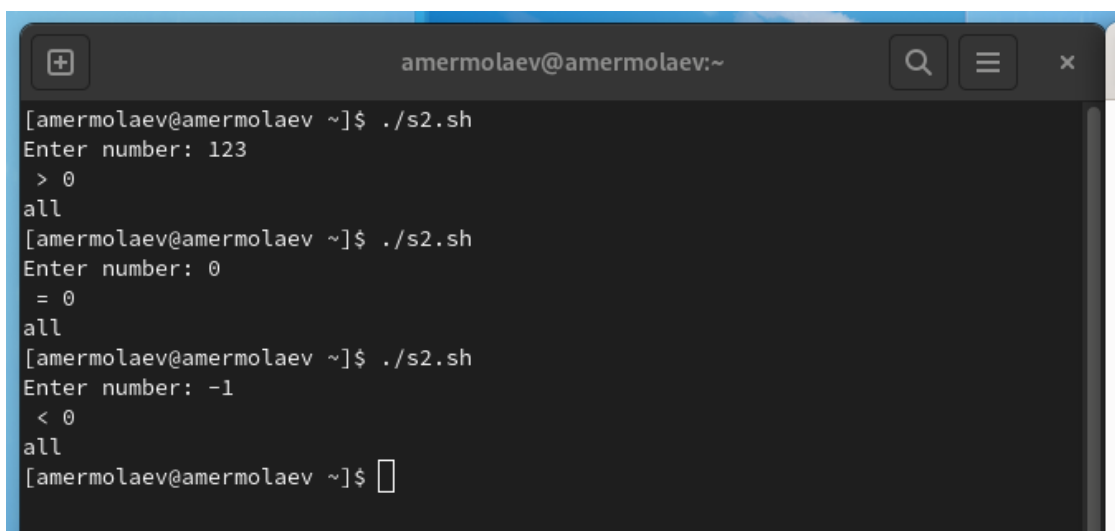
```
emacs@amermolaev
File Edit Options Buffers Tools Sh-Script Help
[Icons: File, Folder, Save, Undo, Cut, Copy, Paste, Find]

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

int main()
{
    printf("Enter number: ");
    int n=0;
    scanf("%d", &n);
    if (n<0) exit(0);
    if (n==0) exit(1);
    if (n>0) exit(2);
    return 0;
}

#!/bin/bash
gcc s2.c -o s2
./s2
res=$?
case $res in
    0)echo " < 0";;
    1)echo " = 0";;
    2)echo " > 0";;
esac
echo "all"
```

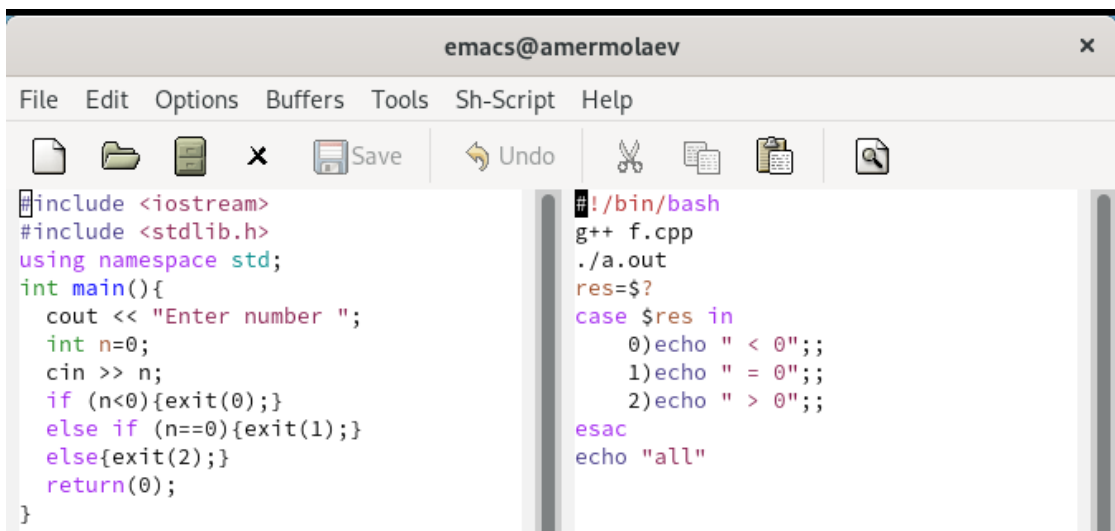
листинг второй программы



```
amermolaev@amermolaev:~$ ./s2.sh
Enter number: 123
> 0
all
amermolaev@amermolaev:~$ ./s2.sh
Enter number: 0
= 0
all
amermolaev@amermolaev:~$ ./s2.sh
Enter number: -1
< 0
all
amermolaev@amermolaev:~$
```

работа первой программы

Для сравнения данное задание можно выполнить при помощи языка C++.

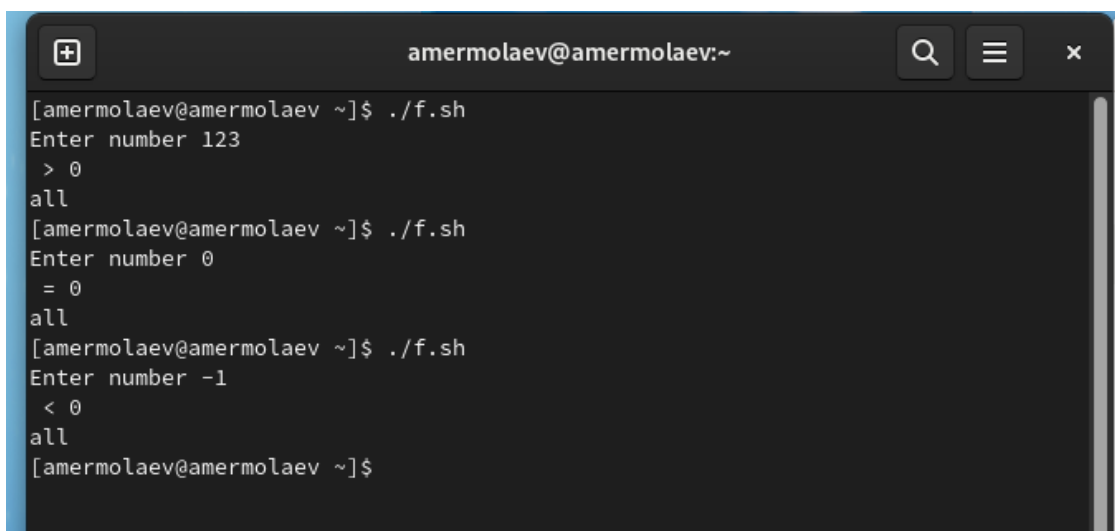


```
emac@amermolaev
File Edit Options Buffers Tools Sh-Script Help
Save Undo

#include <iostream>
#include <stdlib.h>
using namespace std;
int main(){
    cout << "Enter number ";
    int n=0;
    cin >> n;
    if (n<0){exit(0);}
    else if (n==0){exit(1);}
    else{exit(2);}
    return(0);
}

#!/bin/bash
g++ f.cpp
./a.out
res=$?
case $res in
    0)echo " < 0";;
    1)echo " = 0";;
    2)echo " > 0";;
esac
echo "all"
```

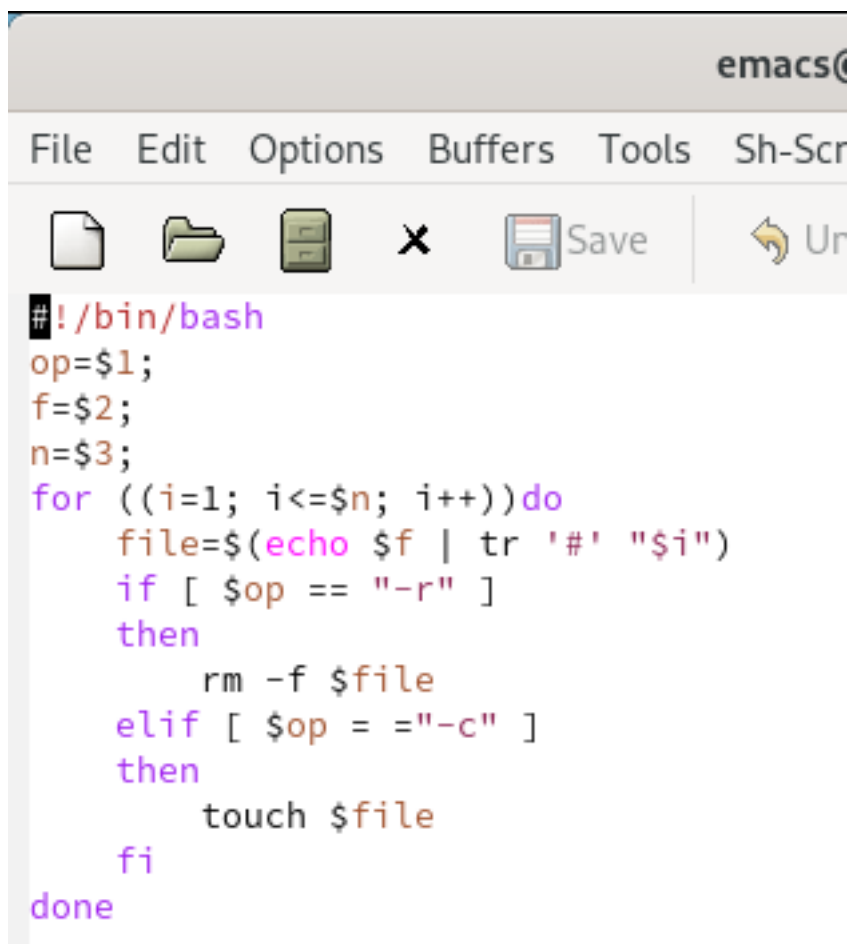
листинг второй программы



```
amermolaev@amermolaev:~  
[amermolaev@amermolaev ~]$ ./f.sh  
Enter number 123  
> 0  
all  
[amermolaev@amermolaev ~]$ ./f.sh  
Enter number 0  
= 0  
all  
[amermolaev@amermolaev ~]$ ./f.sh  
Enter number -1  
< 0  
all  
[amermolaev@amermolaev ~]$
```

работа первой программы

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).

The image shows a screenshot of an Emacs editor window. The title bar at the top right says "emacs@". Below it is a menu bar with "File", "Edit", "Options", "Buffers", "Tools", and "Sh-Scr". Under the menu bar is a toolbar with icons for a file, a folder, a document, a close button (X), a save button (floppy disk), and an undo button (curved arrow). The main text area contains a shell script in bash. The script starts with a shebang line, followed by variable assignments for operation, file, and count. It then enters a loop that iterates from 1 to the count, creating a file with a specific name based on the file variable and the loop index. Depending on the operation, it either removes or creates the file. The script ends with a 'done' keyword.

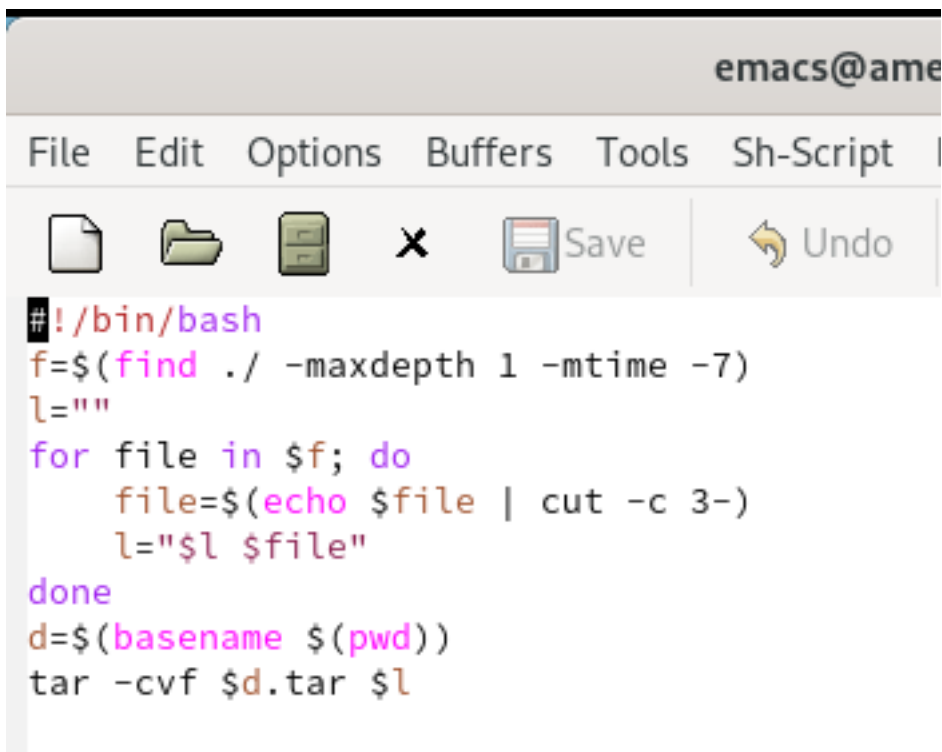
```
#!/bin/bash
op=$1;
f=$2;
n=$3;
for ((i=1; i<=$n; i++))do
    file=$(echo $f | tr '#' "$i")
    if [ $op == "-r" ]
    then
        rm -f $file
    elif [ $op == "-c" ]
    then
        touch $file
    fi
done
```

листинг третьей программы

```
[+]  
ame  
[amermolaev@amermolaev ~]$ ./s3.sh -c foo#.txt 3  
[amermolaev@amermolaev ~]$ ls  
abc1      file.txt      play          script1.sh~   Видео  
alex      f.sh          s1.sh         script2.sh~   Документы  
a.out     f.sh~         s1.sh~        script2.sh~   Загрузки  
australia l1_report.odt s2            script3.sh~   Изображения  
backup    l9            s2.c          script3.sh~   Музыка  
dom.txt   lab09.sh      s2.c~         script4.sh~   Общедоступные  
f1.txt    lab09.sh~     s2.sh         script4.sh~   'Рабочий стол'  
f2.txt    may           s2.sh~        script.py      Шаблоны  
f.cpp     monthly       s3.sh         ski.places  
f.cpp~    monthly.00    s3.sh~        text.txt  
feathers  my_os         script1.sh     work  
[amermolaev@amermolaev ~]$ ./s3.sh -r foo#.txt 3  
[amermolaev@amermolaev ~]$ ls  
abc1      file.txt      play          script1.sh~   Видео  
alex      f.sh          s1.sh         script2.sh~   Документы  
a.out     f.sh~         s1.sh~        script2.sh~   Загрузки  
australia l1_report.odt s2            script3.sh~   Изображения  
backup    l9            s2.c          script3.sh~   Музыка  
dom.txt   lab09.sh      s2.c~         script4.sh~   Общедоступные  
f1.txt    lab09.sh~     s2.sh         script4.sh~   'Рабочий стол'  
f2.txt    may           s2.sh~        script.py      Шаблоны  
f.cpp     monthly       s3.sh         ski.places  
f.cpp~    monthly.00    s3.sh~        text.txt  
feathers  my_os         script1.sh     work  
[amermolaev@amermolaev ~]$
```

работа третьей программы

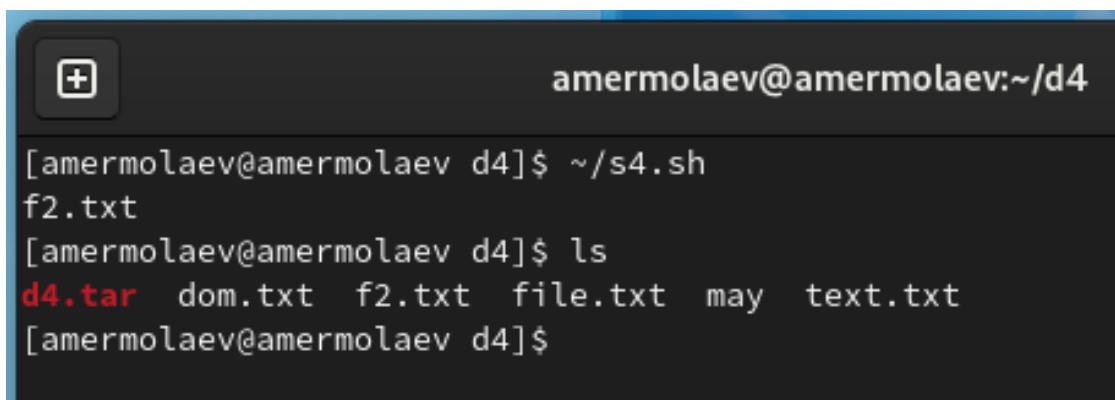
Напишем командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицируем его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find).



The screenshot shows the Emacs editor interface. The title bar at the top reads 'emacs@ame'. Below it is a menu bar with 'File', 'Edit', 'Options', 'Buffers', 'Tools', and 'Sh-Script'. A toolbar contains icons for file operations and buttons for 'Save' and 'Undo'. The main text area contains a shell script with the following code:

```
#!/bin/bash
f=$(find ./ -maxdepth 1 -mtime -7)
l=""
for file in $f; do
    file=$(echo $file | cut -c 3-)
    l="$l $file"
done
d=$(basename $(pwd))
tar -cvf $d.tar $l
```

листинг четвертой программы



The screenshot shows a terminal window with the prompt 'amermolaev@amermolaev:~/d4'. The user has executed the command '~/.s4.sh'. The output shows 'f2.txt' and then the result of 'ls', which lists 'd4.tar', 'dom.txt', 'f2.txt', 'file.txt', 'may', and 'text.txt'.

```
amermolaev@amermolaev:~/d4
[amermolaev@amermolaev d4]$ ~/.s4.sh
f2.txt
[amermolaev@amermolaev d4]$ ls
d4.tar  dom.txt  f2.txt  file.txt  may  text.txt
[amermolaev@amermolaev d4]$
```

работа 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. 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.

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 исполняется 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 рамках выполнения работы я изучил основы программирования в оболочке ОС UNIX и научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.