

Отчет по лабораторной работе №11

Дисциплина:Операционные системы

Шишук Владислав Олегович

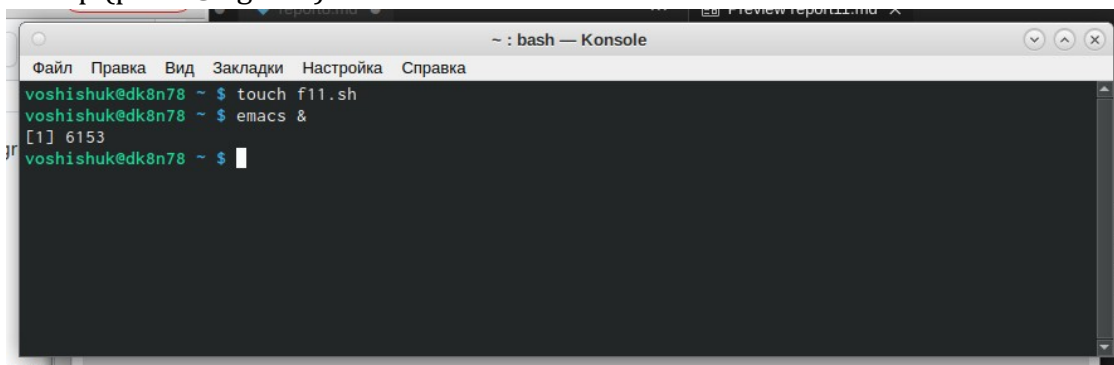
Содержание

Цель работы

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

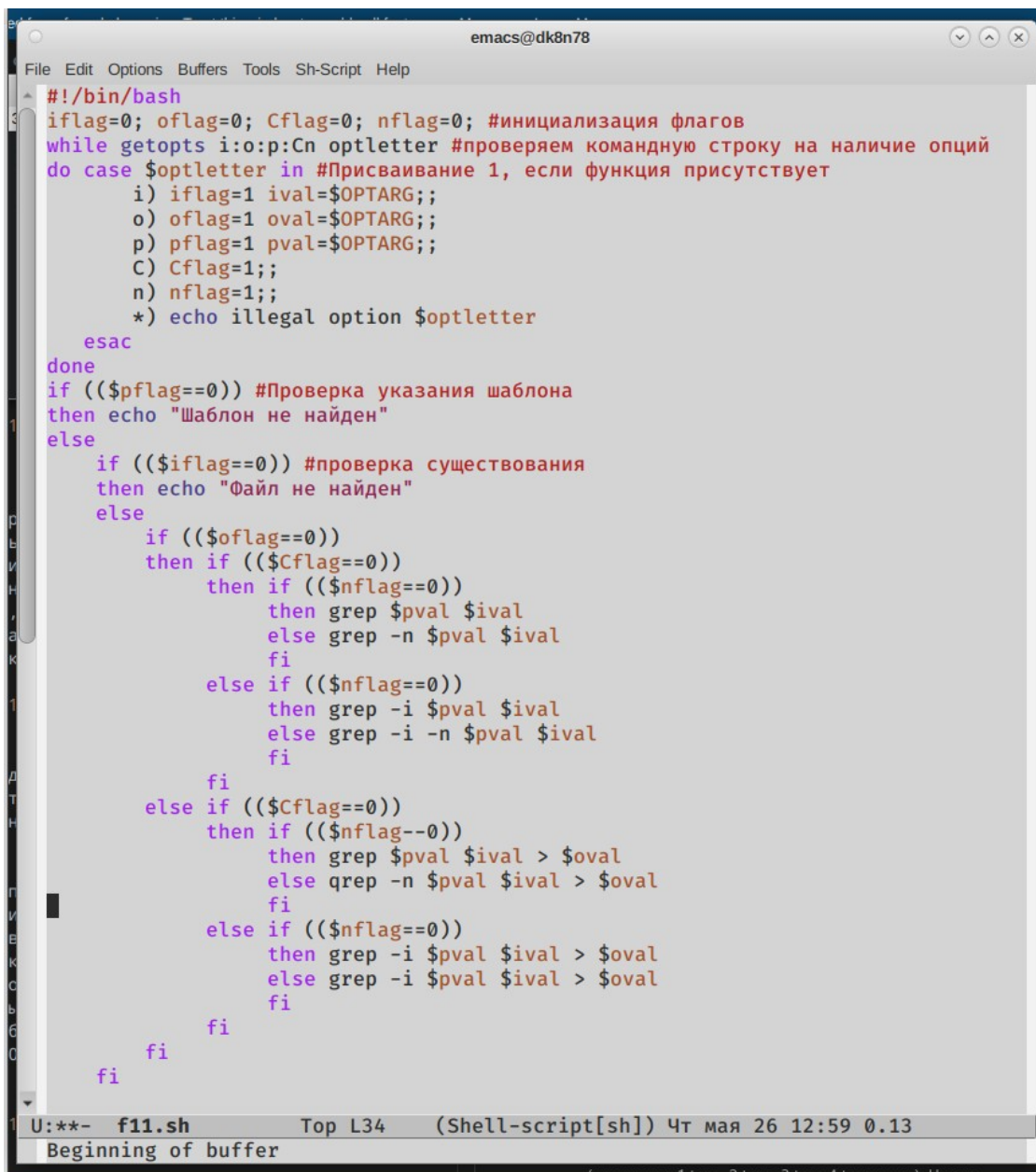
Выполнение лабораторной работы

1. Создаем файл и пишем соответствующие скрипты. (рис. -@fig:001) Пишем командный файл, который анализирует командную строку с ключами:
 - iinputfile — прочитать данные из указанного файла;
 - ooutputfile — вывести данные в указанный файл;
 - ршаблон — указать шаблон для поиска;
 - С — различать большие и малые буквы;
 - n — выдавать номера строк, а затем ищет в указанном файле нужные строки, определяемые ключом
 - p (рис. -@fig:002)



```
~ : bash — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
voshishuk@dk8n78 ~ $ touch f11.sh
voshishuk@dk8n78 ~ $ emacs &
[1] 6153
voshishuk@dk8n78 ~ $
```

рис.1

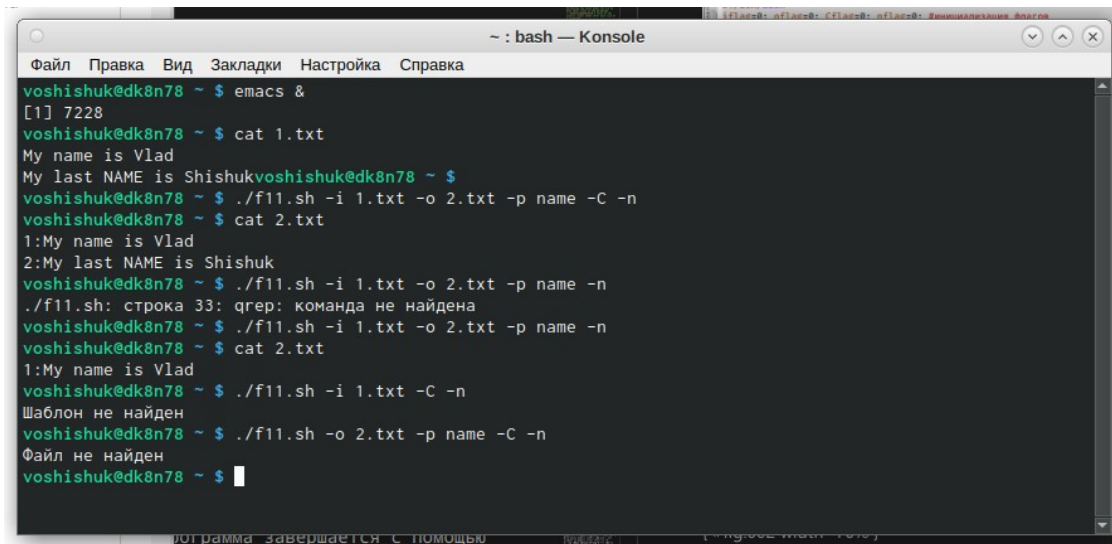


```
#!/bin/bash
iflag=0; oflag=0; Cflag=0; nflag=0; #инициализация флагов
while getopts i:o:p:Cn optletter #проверяем командную строку на наличие опций
do case $optletter in #Присваивание 1, если функция присутствует
    i) iflag=1 ival=$OPTARG;;
    o) oflag=1 oval=$OPTARG;;
    p) pflag=1 pval=$OPTARG;;
    C) Cflag=1;;
    n) nflag=1;;
    *) echo illegal option $optletter
esac
done
if (($pflag==0)) #Проверка указания шаблона
then echo "Шаблон не найден"
else
    if (($iflag==0)) #проверка существования
    then echo "Файл не найден"
    else
        if (($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 qrep -n $pval $ival > $oval
                fi
            else if (($nflag==0))
                then grep -i $pval $ival > $oval
                else grep -i $pval $ival > $oval
                fi
            fi
        fi
    fi
fi
fi
```

U:***- f11.sh Top L34 (Shell-script[sh]) Чт мая 26 12:59 0.13
Beginning of buffer

рис.2

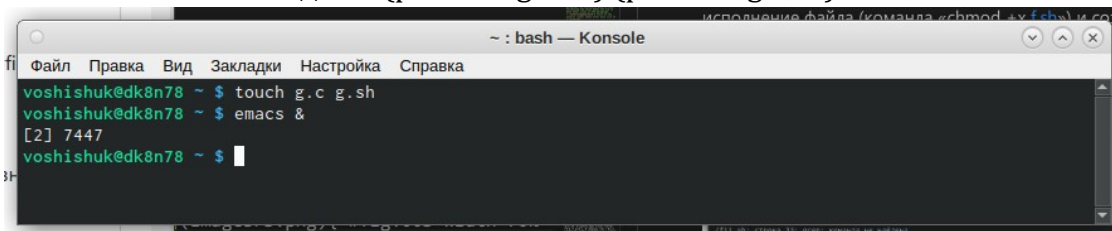
Проверяем работу написанного скрипта, используя различные опции, предварительно добавив право на исполнение файла (команда «chmod +x f.sh») и создав 2 файла, которые необходимы для выполнения программы: 1.txt и 2.txt. Скрипт работает корректно. (рис. -@fig:003)



```
~ : bash — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
voshishuk@dk8n78 ~ $ emacs &
[1] 7228
voshishuk@dk8n78 ~ $ cat 1.txt
My name is Vlad
My last NAME is Shishukvoshishuk@dk8n78 ~ $
voshishuk@dk8n78 ~ $ ./f11.sh -i 1.txt -o 2.txt -p name -C -n
voshishuk@dk8n78 ~ $ cat 2.txt
1:My name is Vlad
2:My last NAME is Shishuk
voshishuk@dk8n78 ~ $ ./f11.sh -i 1.txt -o 2.txt -p name -n
./f11.sh: строка 33: qrep: команда не найдена
voshishuk@dk8n78 ~ $ ./f11.sh -i 1.txt -o 2.txt -p name -n
voshishuk@dk8n78 ~ $ cat 2.txt
1:My name is Vlad
voshishuk@dk8n78 ~ $ ./f11.sh -i 1.txt -C -n
Шаблон не найден
voshishuk@dk8n78 ~ $ ./f11.sh -o 2.txt -p name -C -n
Файл не найден
voshishuk@dk8n78 ~ $
```

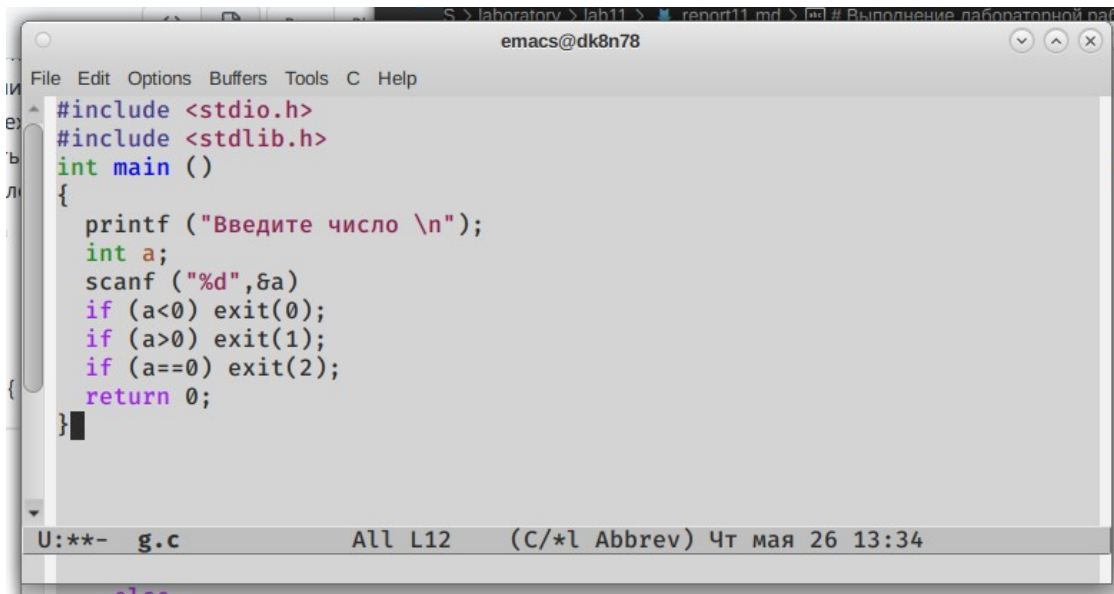
рис.3

2. Создаем файлы g.c и g.sh и пишем соответствующие скрипты. (рис. -@fig:004)
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. (рис. -@fig:005) (рис. -@fig:006)



```
~ : bash — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
voshishuk@dk8n78 ~ $ touch g.c g.sh
voshishuk@dk8n78 ~ $ emacs &
[2] 7447
voshishuk@dk8n78 ~ $
```

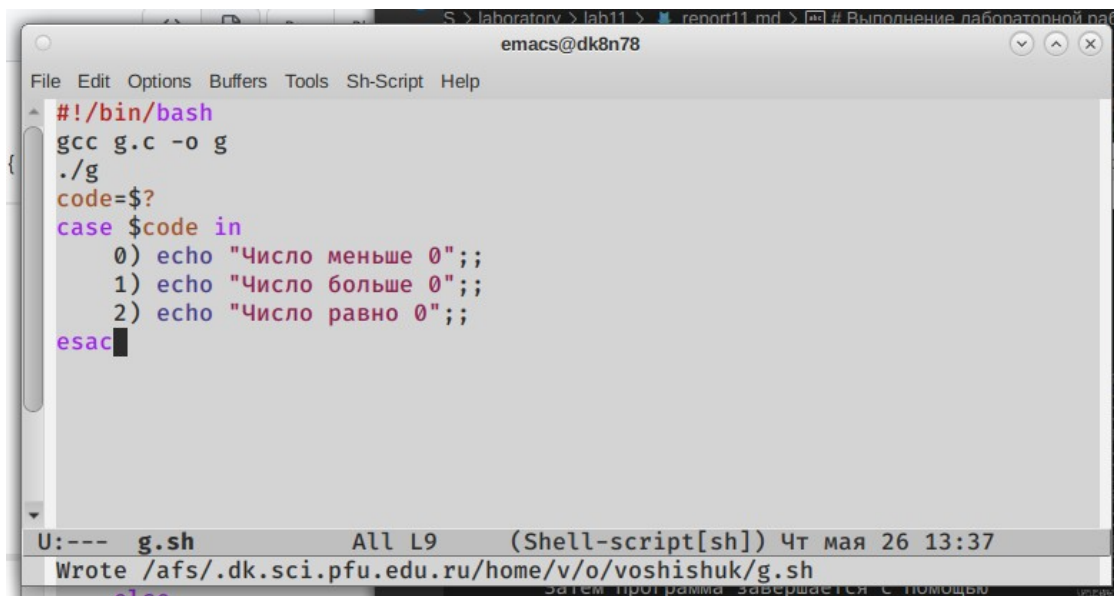
рис.4



```
#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;
}
```

U:*** g.c All L12 (C/*l Abbrev) Чт мая 26 13:34

рис.5



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

U:--- g.sh All L9 (Shell-script[sh]) Чт мая 26 13:37
Wrote /afs/.dk.sci.pfu.edu.ru/home/v/o/voshishuk/g.sh

рис.6

Проверяем работу написанных скриптов (команда «./g.sh»), предварительно добавив право на исполнение файла (команда «chmod +x g.sh»). Скрипты работают корректно. (рис. -@fig:007) (рис. -@fig:010)

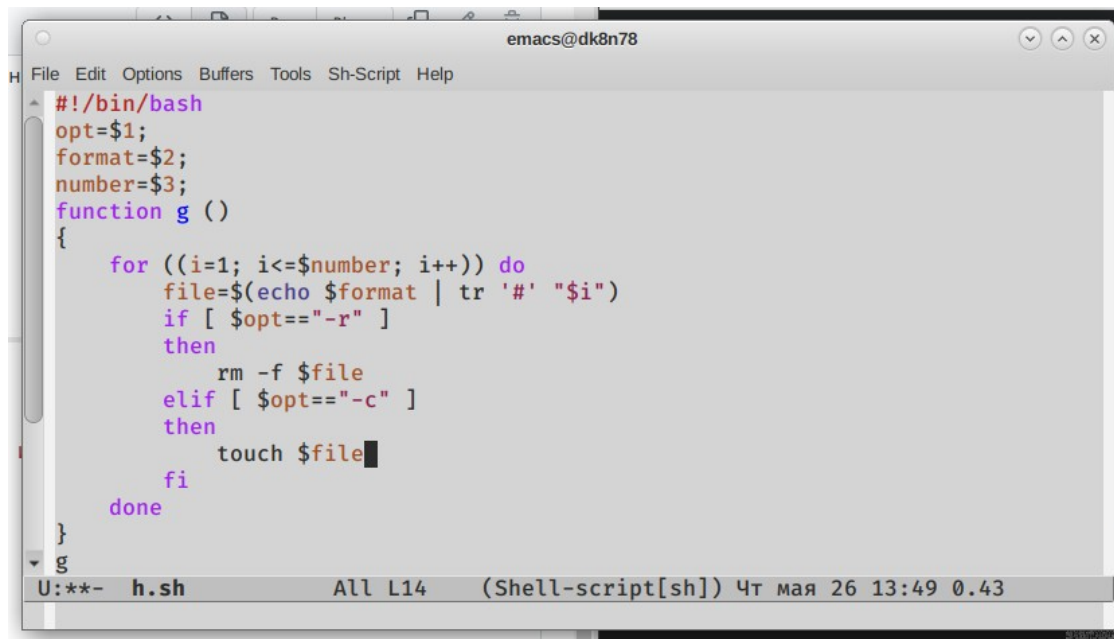
```
strict mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
~ : bash — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
voshishuk@dk8n78 ~ $ chmod +x g.sh
voshishuk@dk8n78 ~ $ ./g.sh
Введите число
3
Число больше 0
voshishuk@dk8n78 ~ $ ./g.sh
Введите число
0
Число равно 0
voshishuk@dk8n78 ~ $ ./g.sh
Введите число
-2
Число меньше 0
voshishuk@dk8n78 ~ $
```

рис.7

3. Создаем файл и пишем соответствующие скрипты. (рис. -@fig:008) Написал 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). (рис. -@fig:009)

```
~ : bash — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
[1]-  Завершён      emacs
[2]+  Завершён      emacs
voshishuk@dk8n78 ~ $ touch h.sh
voshishuk@dk8n78 ~ $ emacs &
[1] 8001
voshishuk@dk8n78 ~ $
```

рис.8



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

U:*- h.sh All L14 (Shell-script[sh]) Чт мая 26 13:49 0.43

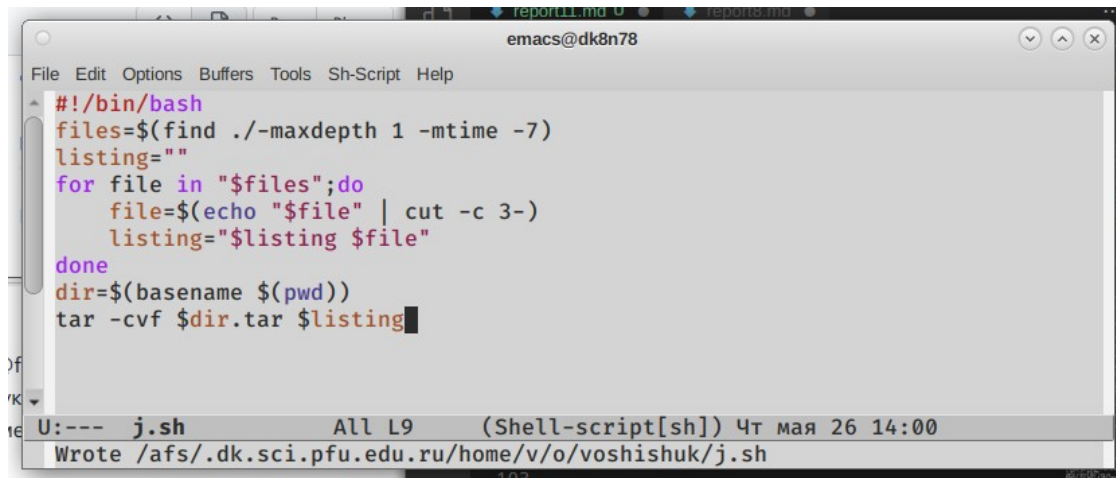
рис.9

Проверил работу написанного скрипта (команда «./h.sh»), предварительно добавив право на исполнение файла (команда «chmod +x h.sh»). Создаем три файла (команда «./h.sh -c et#.txt 3»), удовлетворяющие условию задачи, а потом удаляем их (команда «./h.sh -r et#.txt 3»). (рис. -@fig:010)

```
File Edit Selection View Go Run Terminal Help
~ : bash — Konsole
Файл Правка Вид Закладки Настройка Справка
voshishuk@dk8n78 ~ $ chmod +x h.sh
voshishuk@dk8n78 ~ $ ./h.sh -c et#/txt 3
voshishuk@dk8n78 ~ $ ls
123          f11.sh      g          my_os      Загрузки
1.txt        f11.sh~    g.c        play       Изображения
'2022-05-19 17-08-24.mkv' feathers    g.c~       public     Музыка
2.txt        file1.sh   GNUstep    public_html  Общедоступные
acb1         file1.sh~  g.sh       ski.places  'Рабочий стол'
australia    file2.sh   g.sh~      tmp         Шаблоны
backup       file2.sh~  h.sh       untitled_1.odt
backup.sh    file.sh    h.sh~      work
backup.sh~   file.sh~   io.h       Видео
em1.txt~     file.txt   lab07.sh   Документы
voshishuk@dk8n78 ~ $ ./h.sh -r et#/txt 3
voshishuk@dk8n78 ~ $ ls
123          backup.sh~  file2.sh~  g.sh       public      Загрузки
1.txt        em1.txt~   file.sh    g.sh~      public_html  Изображения
'2022-05-19 17-08-24.mkv' f11.sh     file.sh~   h.sh       ski.places   Музыка
2.txt        f11.sh~   file.txt   h.sh~      tmp          Общедоступные
acb1         feathers   g          io.h       untitled_1.odt 'Рабочий стол'
australia    file1.sh   g.c        lab07.sh   work         Шаблоны
backup       file1.sh~  g.c~       my_os      Видео
backup.sh    file2.sh   GNUstep    play       Документы
voshishuk@dk8n78 ~ $
```

рис.10

4. Создаем файл и пишем соответствующие скрипты. 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). (рис. -@fig:011)

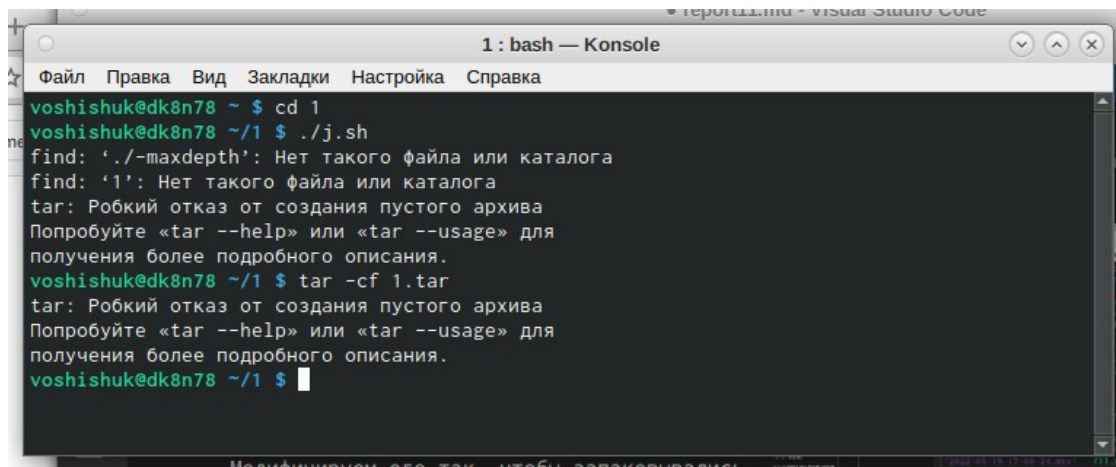


```
emacs@dk8n78
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
files=$(find ./-maxdepth 1 -mtime -7)
listing=""
for file in "$files";do
    file=$(echo "$file" | cut -c 3-)
    listing="$listing $file"
done
dir=$(basename $(pwd))
tar -cvf $dir.tar $listing
```

U:--- j.sh All L9 (Shell-script[sh]) Чт мая 26 14:00
Wrote /afs/.dk.sci.pfu.edu.ru/home/v/o/voshishuk/j.sh

рис.11

Проверяем работу написанного скрипта, предварительно добавив право на исполнение файла (команда «chmod +x j.sh») и создав отдельный каталог 1 с несколькими файлами. К сожалению скрипт не видит каталог и не выполняет свою работу.



```
1: bash — Konsole
Файл Правка Вид Закладки Настройка Справка
voshishuk@dk8n78 ~ $ cd 1
voshishuk@dk8n78 ~/1 $ ./j.sh
find: './-maxdepth': Нет такого файла или каталога
find: '1': Нет такого файла или каталога
tar: Робкий отказ от создания пустого архива
Попробуйте «tar --help» или «tar --usage» для
получения более подробного описания.
voshishuk@dk8n78 ~/1 $ tar -cf 1.tar
tar: Робкий отказ от создания пустого архива
Попробуйте «tar --help» или «tar --usage» для
получения более подробного описания.
voshishuk@dk8n78 ~/1 $
```

рис.12

Выводы

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

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 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 `getopts` также понимает переменные типа массив, следовательно, можно использовать её в функции не только для синтаксического анализа аргументов функций, но и для анализа введенных 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 отрезультатов 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 mans/i.sn` проверяет, существует ли файл `mans/i.sn` 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.