

# Российский университет дружбы народов

факультет физико-математических и естественных наук

## Отчет по лабораторной работе № 12

дисциплина : *Операционные системы*

студент Блохин александр НКН

Москва

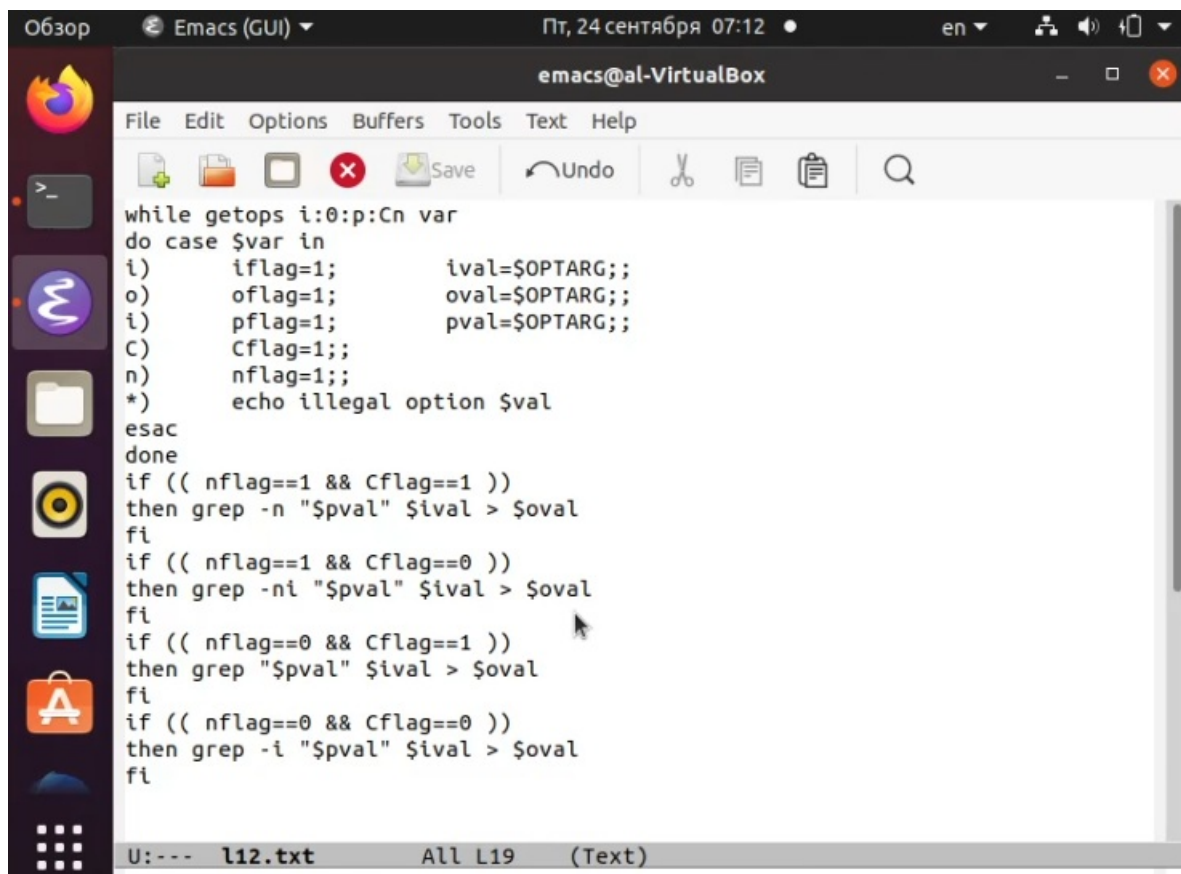
2021

## Цель работы

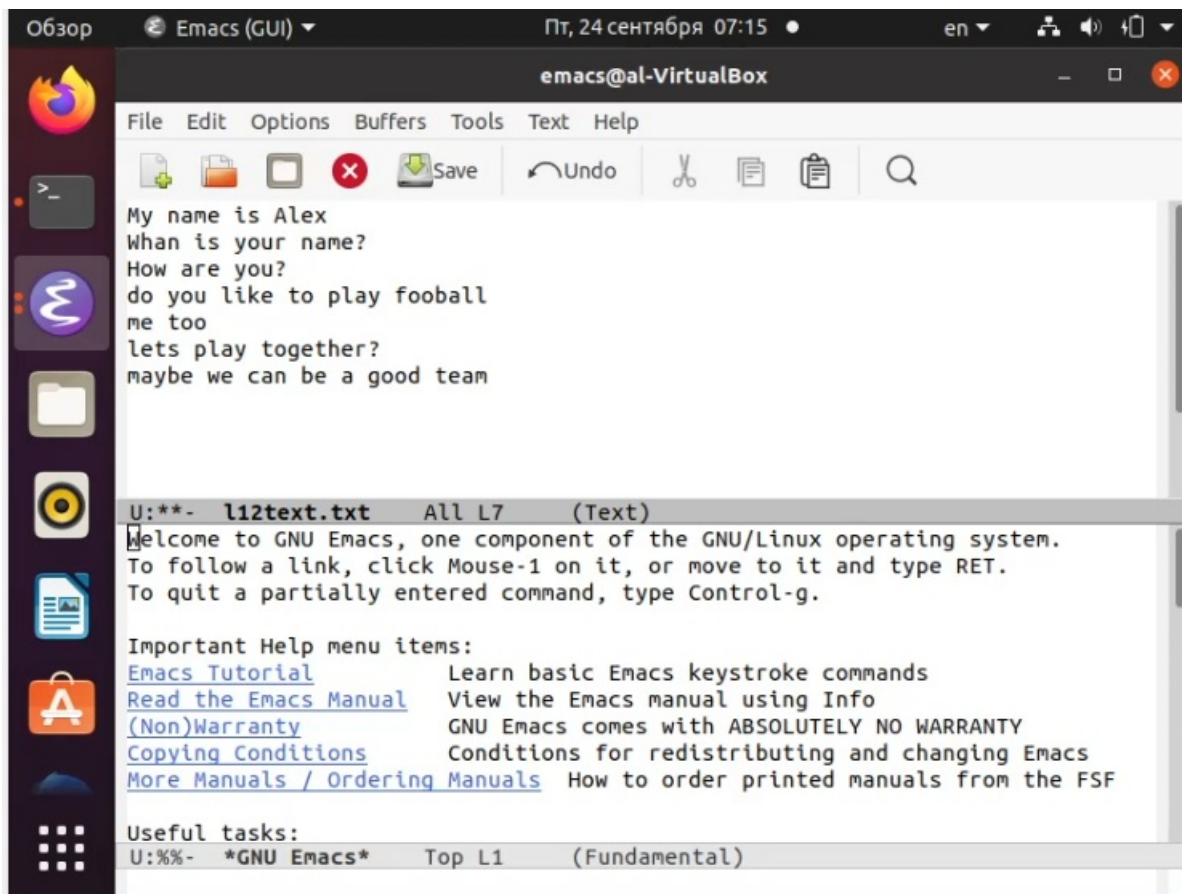
Изучить основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов

## Ход работы

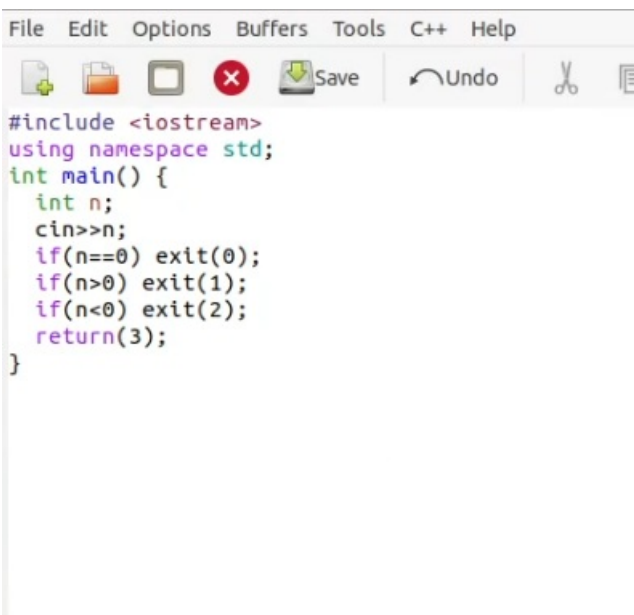
1. Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами: `-i` — прочитать данные из указанного файла; `-o` — вывести данные в указанный файл; `-r` — шаблон — указать шаблон для поиска; `-C` — различать большие и малые буквы; `-n` — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом `-p`.



```
File Edit Options Buffers Tools Text Help
while getopts i:0:p:Cn var
do case $var in
i)      iflag=1;          ival=$OPTARG;;
o)      oflag=1;          oval=$OPTARG;;
i)      pflag=1;          pval=$OPTARG;;
C)      Cflag=1;;
n)      nflag=1;;
*)      echo illegal option $val
esac
done
if (( nflag==1 && Cflag==1 ))
then grep -n "$pval" $ival > $oval
fi
if (( nflag==1 && Cflag==0 ))
then grep -ni "$pval" $ival > $oval
fi
if (( nflag==0 && Cflag==1 ))
then grep "$pval" $ival > $oval
fi
if (( nflag==0 && Cflag==0 ))
then grep -i "$pval" $ival > $oval
fi
U: --- l12.txt All L19 (Text)
```



2. Написать 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.



```

Обзор Терминал ▼ Пт, 24 сентября 07:48
al@al-VirtualBox: ~

al@al-VirtualBox:~$ sh 2l12.txt
3
positive
al@al-VirtualBox:~$ sh 2l12.txt
0
0
al@al-VirtualBox:~$ sh 2l12.txt
-1
negative
al@al-VirtualBox:~$

```

3. Написать 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).

```

x=1
while getopts r var
do case $var in
r)      rflag=1;;
esac
done
if (( rflag==1 ))
then
    while (( $x !=$1 ))
    do
        rm -r "$x".tmp
        let x+=1
    done
else
    while (( $x !=$1 ))
    do
        echo " " > "$x".tmp
        let x+=1
    done
fi

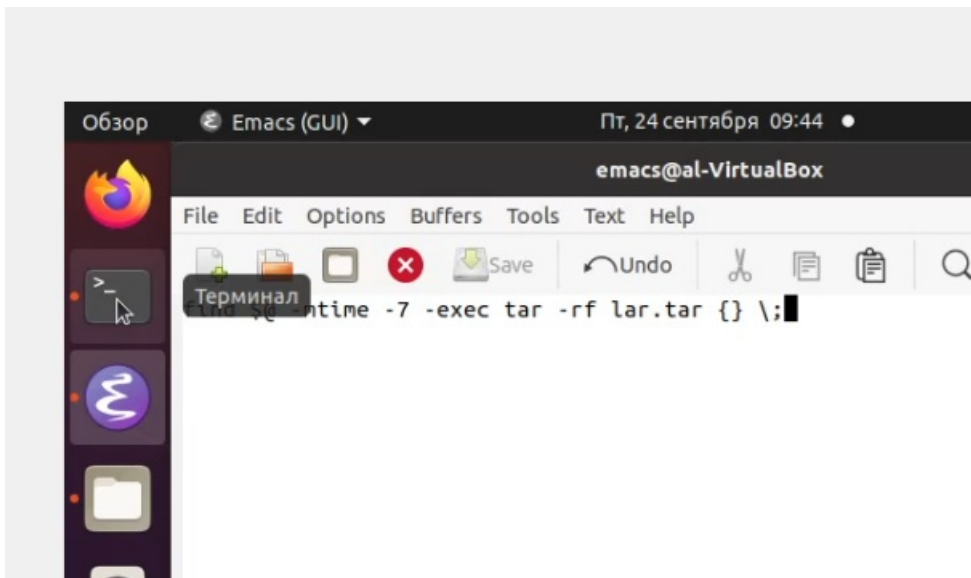
```

```

al@al-VirtualBox:~$ bash n3.txt 7
al@al-VirtualBox:~$ ls
1                '#abc#'          l12r.txt         monthly          s4.tx
1.tmp            abc              l12r.txt~        n3.txt           s4.tx
2l12             '#abc1#'         l12text.txt      n3.txt~          ski
'#2l12.cpp#'    abc1             l12text.txt~     oslab            test
2l12.cpp         abc1~           l12.txt          ostest           test1
2l12.cpp~        abc2            l12.txt~         os.txt           tmp
2l12.txt         al              l13.sh           play             usr
2l12.txt~        ans.txt         lab12            r1.txt           work
2.tmp            backup          lab1.txt         r1.txt~          Видео
3.tmp            conf.txt        lab1.txt~        random.txt       Докум
4.tmp            equipment        lockfile         random.txt~      Загру
5.tmp            etc              lw.sh            reports           Изобр
6.tmp            f               may              s2.txt           Музык

```

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



## Вывод

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

## Контрольные вопросы

1. Весьма необходимой при программировании является команда `getopts`, которая 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, `-F` является флагом для команды `ls -F`. Иногда эти флаги имеют аргументы, связанные с ними. Программы интерпретируют эти флаги, соответствующим образом изменяя свое поведение. Строка опций `option-string` – это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за этой буквой должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введенные данные с помощью оператора `case`. Предположим, необходимо распознать командную строку следующего формата: `testprog -ifile_in.txt -ofile_out.doc -L -t -r`. Вот как выглядит использование оператора `getopts` в этом случае: `while getopts o:ltr optletter do case $optletter in o) oflag=1; oval=$OPTARG;; i) iflag=1; ival=$OPTARG;; L) Lflag=1;; t) tflag=1;; r) rflag=1;; *) echo Illegal option $optletter esac done`. Функция `getopts` включает две специальные переменные среды – `OPTARG` и `OPTIND`. Если ожидается дополнительное значение, то `OPTARG` устанавливается в значение этого аргумента (будет равно `file_in.txt` для опции `i` и `file_out.doc` для опции `o`). `OPTIND` является числовым индексом на упомянутый аргумент. Функция `getopts` также понимает переменные типа массив, следовательно, можно использовать ее в функции не только для синтаксического анализа аргументов функций, но и для анализа введенных пользователем данных.

2. При перечислении имен файлов текущего каталога можно использовать следующие символы:

– соответствует произвольной, в том числе и пустой строке; `?` – соответствует любому одному символу;

`[c1-c1]` – соответствует любому символу, лексикографически находящемуся между символами `c1` и `c2`.

`echo *` – выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`;

`ls *.c` – выведет все файлы с последними двумя символами, равными `.c`.

`echo prog.?` – выдаст 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 от результатов проверки некоторого условия. Для решения подобных задач язык программирования `bash` предоставляет Вам 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 дела являются операторами языка программирования `bash`. Поэтому при описании языка программирования `bash` термин оператор будет использоваться наравне с термином команда.

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 перестает быть правильным. Пример бесконечного цикла `while`, с прерыванием в момент, когда файл перестает существовать:

```
while true do if [! -f $file] then break fi sleep 10 done
```

5. 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

6. Введенная строка означает условие существования файла `man$s/$i.$s`

7. Если речь идет о 2-х параллельных действиях, то это `while`. когда мы показываем, что сначала делается 1-е действие. потом оно заканчивается при наступлении 2-го действия, применяем `until`.