

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 12

Дисциплина: Операционные системы

Студент:

Гаглов Олег Мелорович

Преподаватель:

Велиева Т.В.

Группа: НПИбд-01-20

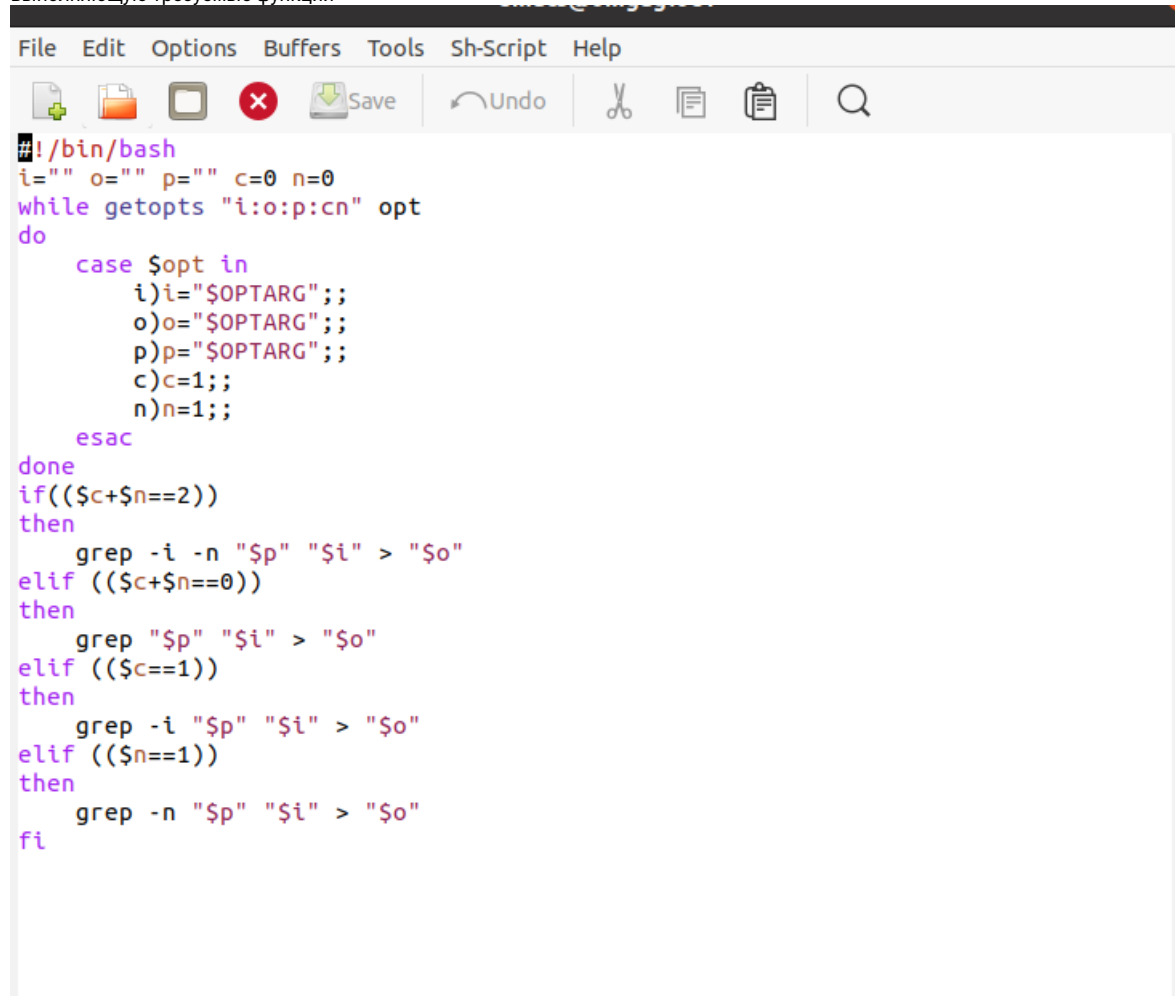
МОСКВА 2021 г.

Цель работы:

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

Выполнение работы :

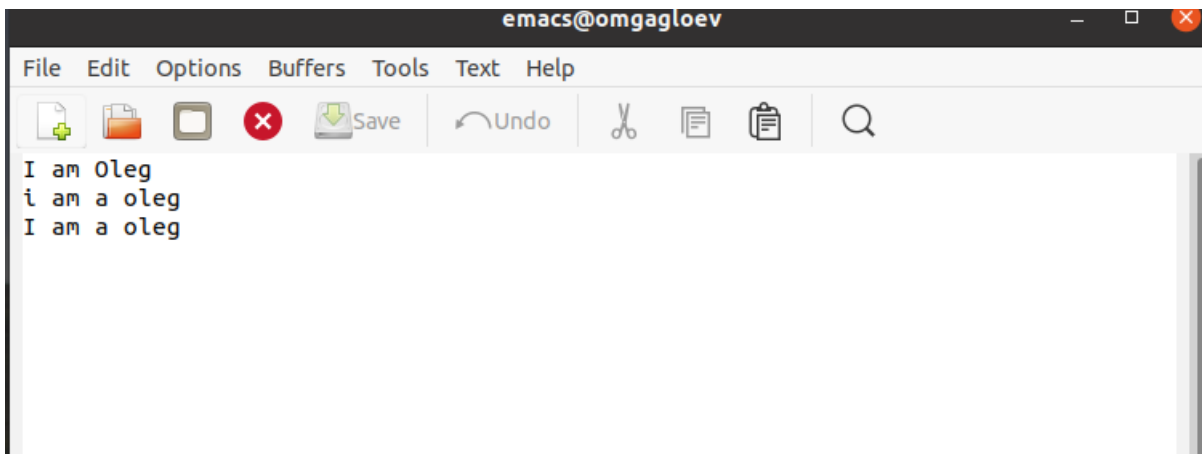
Задание 1: 1. Создаю файл для выполнения данного задания и сразу открываю его с помощью команды `emacs sr1.sh`. Написал программу, выполняющую требуемые функции



```
#!/bin/bash
i="" o="" p="" c=0 n=0
while getopts "i:o:p:cn" opt
do
    case $opt in
        i)i="$OPTARG";;
        o)o="$OPTARG";;
        p)p="$OPTARG";;
        c)c=1;;
        n)n=1;;
    esac
done
if(($c+$n==2))
then
    grep -i -n "$p" "$i" > "$o"
elif (($c+$n==0))
then
    grep "$p" "$i" > "$o"
elif (($c==1))
then
    grep -i "$p" "$i" > "$o"
elif (($n==1))
then
    grep -n "$p" "$i" > "$o"
fi
```

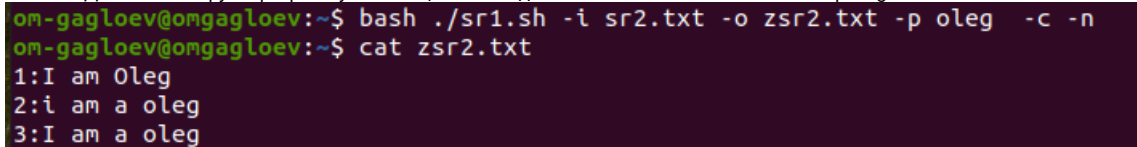
Теперь опишу её: В

первой строке мы вызываем интерпретатор `bash`. Далее идет блок объявления нужных переменных, которые изначально пусты или равны нулю. Затем, используя оператор `getopts`, а также циклы `if` и `elif`, которые будут помогать распознать, какие именно действия нам нужно выполнить в зависимости от упоминания ключей `-c` и `-n`. Сами действия выполняются в строках `grep`. Далее создаю текстовый файл `sc2.txt` с помощью команды `emacs sc2.txt`



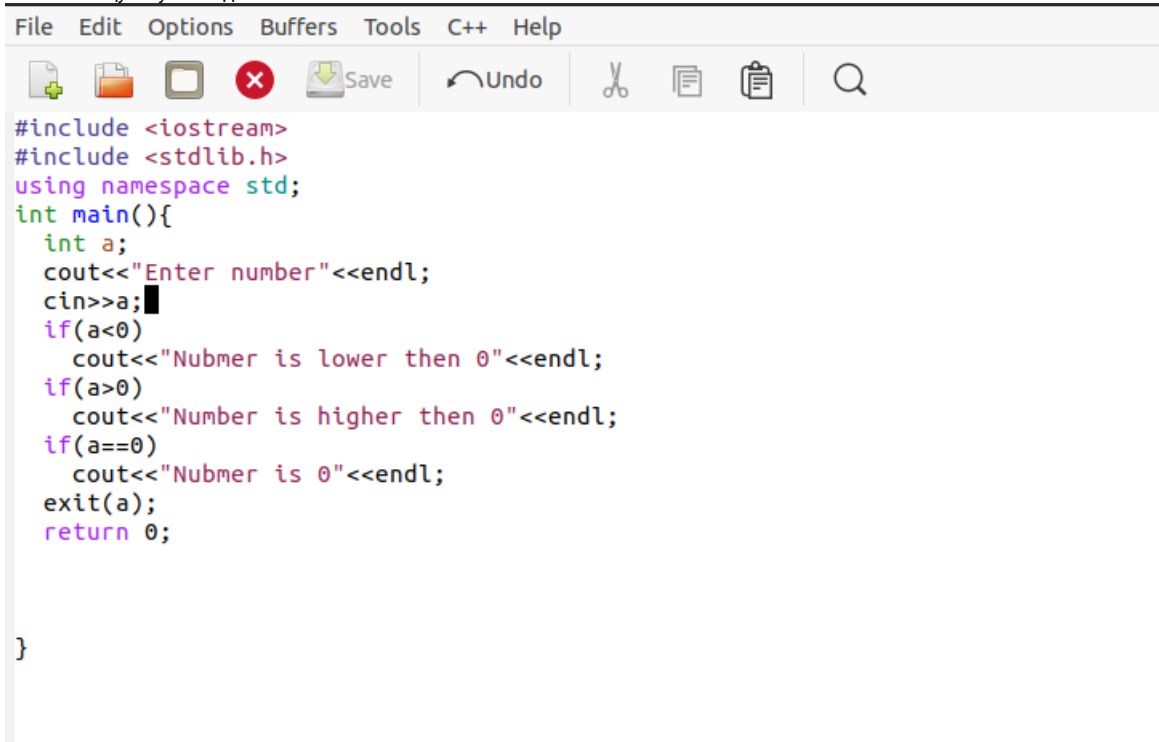
и заполняю его

текстом Далее активирую программу с помощью команды `bash ./sr1.sh -i sr2.txt -o zsr2.txt -p oleg -C -n`.



Здесь sc1.sh - название

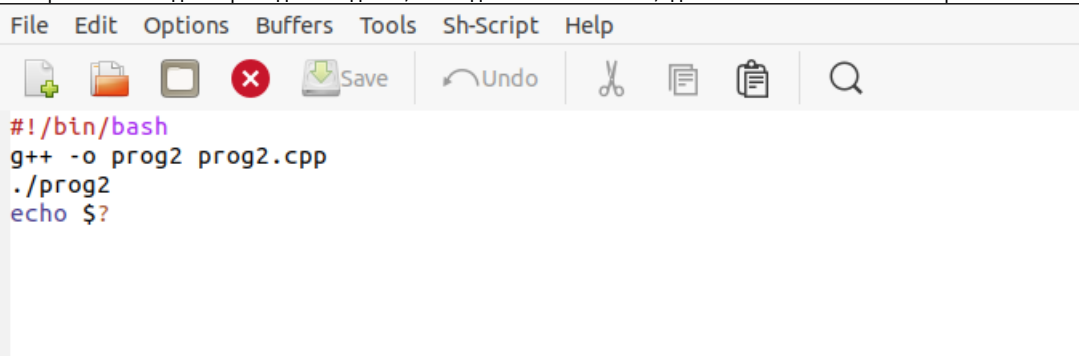
командного файла, sc2.txt - название файла, с которого мы будем считывать, zsc2.txt файл, в который мы будем записывать данные, oleg - слово для поиска. Обозначаем сразу две опции. Далее используем команды `cat zsc2.txt` чтобы просмотреть содержимое файла и убедиться в правильности выполнения программы ##### Задание 2 1.С помощью команды `emacs prog2.cpp`, создаю файл. Написал программу на языке программирования C++, выполняющую нужные действия



```
#include <iostream>
#include <stdlib.h>
using namespace std;
int main(){
    int a;
    cout<<"Enter number"<<endl;
    cin>>a;
    if(a<0)
        cout<<"Nubmer is lower then 0"<<endl;
    if(a>0)
        cout<<"Number is higher then 0"<<endl;
    if(a==0)
        cout<<"Nubmer is 0"<<endl;
    exit(a);
    return 0;
}
```

Структура : в первых трех

строках я подключаю необходимые библиотеки и инициализирую пространство имен. Далее приступаем к основной части программы. Инициализируем переменную a для хранения числа int a. Считываем значение переменной с клавиатуры командой `cin>>a`. Далее мы с помощью `if` и разных условий `<`, `>`, `=` сравниваем число с нулем и выводим `cout` нужную информацию. Затем программа завершается с помощью функции `exit(a)`. 2. Создаю и открываю командный файл для 2 задания, командой `emacs z2kom.sh`, где `z2kom.sh` - название этого файла. 3. Написал командный файл



```
#!/bin/bash
g++ -o prog2 prog2.cpp
./prog2
echo $?
```

doc2.sh

Его структура: в первой

строке мы вызываем интерпретатор `bash`. Во второй строке мы компилируем файл, а в третьей вызываем на выполнение `./prog2` В конце анализируем и передаем на экран с помощью `echo $?`, какое число было введено для сравнения с нулем. 4. Запускаю командный файл командой `bash ./doc2.sh`. У нас появляется запрос на ввод числа, куда я ввожу несколько цифр для проверки

```
om-gagloev@omgagloev:~$ bash ./doc2.sh
Enter number
0
Nubmer is 0
0
om-gagloev@omgagloev:~$ bash ./doc2.sh
Enter number
5
Number is higher then 0
5
om-gagloev@omgagloev:~$
```

5. Видим, что программа выводит нужную фразу и число, т.е работает верно

Задание 3 :

1. Создаю командный файл prog3.sh с помощью команды emacs prog3.sh и пишу код

```
#!/bin/bash
n=""
a=""
echo "How much files should i create?"
read n
for((i=1;i<=n;i++))
do
    touch $i.tmp
done
echo "Created files:"
ls
echo "Should i cleane the created files? Type(y/n)"
read a
if(a=="y")
then
    for((i=1;i<=n;i++))
    do
        rm $i.tmp
    done
    echo " files was deleted"
    ls
fi
```

2. Теперь опишу его структуру: в первой строке мы вызываем интерпретатор bash. Далее инициализируем переменные для хранения количества файлов n и считывания ответа a. Далее идет строка, спрашивающая у пользователя сколько файлов создать и считывает это количество с клавиатуры (read n). Далее циклом, который выполняется n раз, мы создаем файлы с нужным названием. После этого программа выводит содержимое каталога, чтобы можно было увидеть, что файлы созданы (команда ls). Далее программа спрашивает, нужно ли удалить файлы и считывает ответ с клавиатуры. Если ответ у, то циклом мы удаляем все файлы. После чего снова выводится содержимое каталога, чтобы можно было увидеть, что файлы удалены ls
3. Запускаю файл командой bash ./prog3.sh

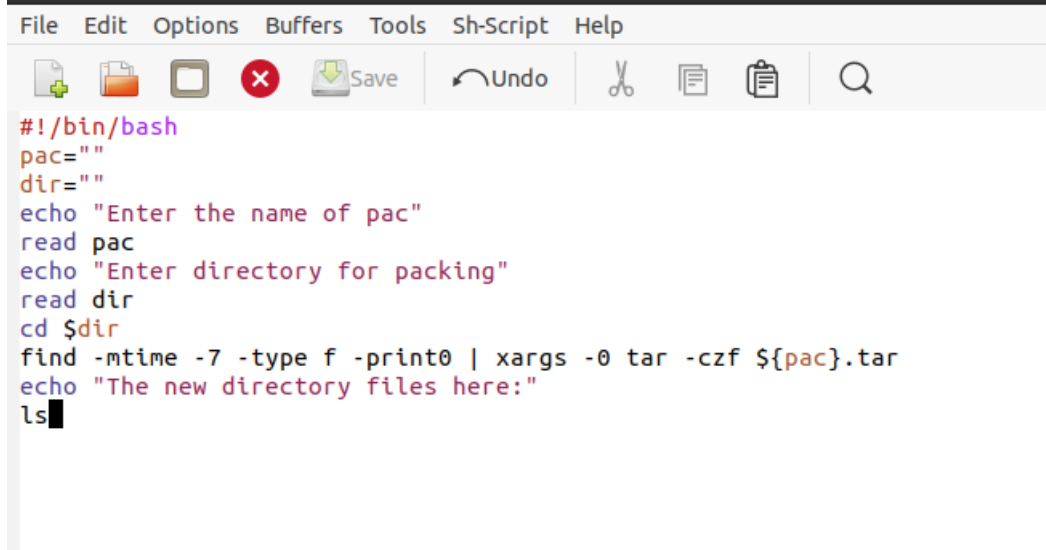
```
om-gagloev@omgagloev:~$ bash ./prog3.sh
How much files should i create?
5
Created files:
1.tmp      doc2.sh~      prog2.cpp     sc2.sh~      sr2.txt~      Общедоступные
2.tmp      file.txt      prog2.cpp~    sc3.sh~      work          'Рабочий стол'
3.tmp      '#lab10.sh#'  prog3.sh     sc3.sh~      zsr2.txt      Шаблоны
4.tmp      lab10.sh     prog3.sh~    sc4.sh~      Видео
5.tmp      newdir       README.md    snap         Документы
backup     '#newfile.txt#'  sc1.sh       sr1.sh       Загрузки
cong.txt   OPTARG       sc1.sh~      sr1.sh~      Изображения
doc2.sh    prog2        sc2.sh       sr2.txt      Музыка
Should i cleane the created files? Type(y/n)
y
files was deleted
backup     '#newfile.txt#'  sc1.sh       sr1.sh       Загрузки
cong.txt   OPTARG          sc1.sh~      sr1.sh~      Изображения
doc2.sh    prog2          sc2.sh       sr2.txt      Музыка
doc2.sh~   prog2.cpp      sc2.sh~      sr2.txt~     Общедоступные
file.txt   prog2.cpp~     sc3.sh       work         'Рабочий стол'
'#lab10.sh#' prog3.sh      sc3.sh~      zsr2.txt     Шаблоны
lab10.sh   prog3.sh~     sc4.sh       Видео
newdir     README.md      snap         Документы
om-gagloev@omgagloev:~$
```

и убеждаюсь, что

программа работает

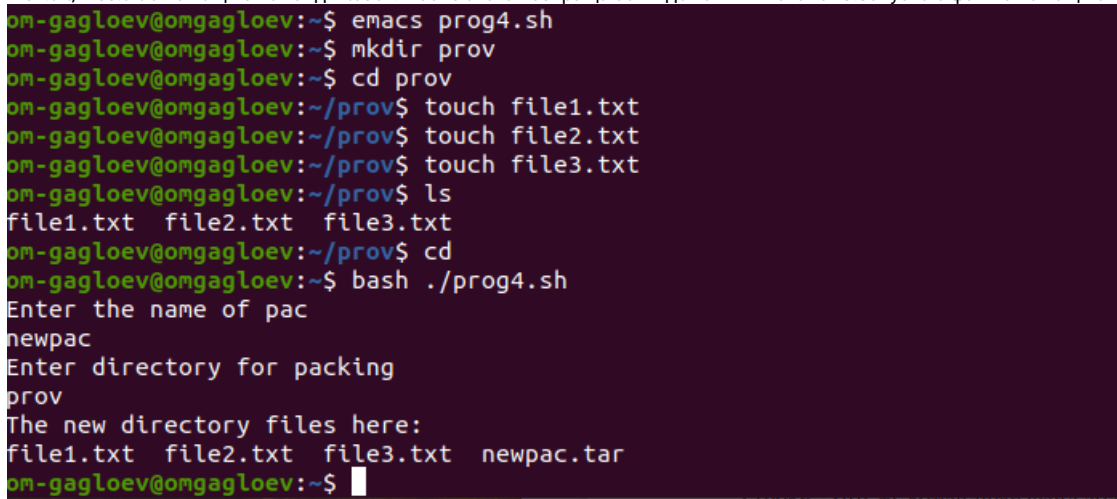
Задание 4:

1. Командой `emacs prog4.sh` создаю файл. Затем написал командный файл



```
File Edit Options Buffers Tools Sh-Script Help
Save Undo
#!/bin/bash
pac=""
dir=""
echo "Enter the name of pac"
read pac
echo "Enter directory for packing"
read dir
cd $dir
find -mtime -7 -type f -print0 | xargs -0 tar -czf ${pac}.tar
echo "The new directory files here:"
ls
```

Теперь опишу его структуру: в первой строке мы вызываем интерпретатор `bash`. Далее мы инициализируем переменные для имени директории `dir` и имени архива `pac`. Запрашиваем имя архива, который будем создавать, и имя директории, в которой будем работать. И вводим их с клавиатуры `read`. переходим в нужный каталог `cd $dir`. Далее мы начинаем поиск `find`. В этой строке: `find -mtime -7 -type f -print0` - поиск осуществляется в текущем каталоге, `-mtime -7` - файлы, редактированные не позднее чем 7 дней назад, `-type f` - поиск именно файлов, `-print0` - позволяет выводить полный путь к файлу на стандартном выходе, за которым следует нулевой символ. Далее используем конвейер и создаем архив с заданным с клавиатуры именем при помощи команды `tar`, `xargs` - флаг `-0` `xargs` используем, чтобы поместить все найденные файлы в архив. Ключи `-czf` помогут создать архив в `linux`, После чего выводим содержимое каталога, чтобы убедиться в том, что все выполнено 2. С помощью команды `mkdir` создаю каталог `prov`, перехожу в него с помощью команды `cd`. Далее создаю в нем файлы `file1.txt`, `file2.txt`, `file3.txt` с помощью команды `touch`. После этого возвращаюсь в домашний каталог 3. Запускаю файл с помощью команды `bash ./prog4.sh`



```
on-gagloev@omgagloev:~$ emacs prog4.sh
on-gagloev@omgagloev:~$ mkdir prov
on-gagloev@omgagloev:~$ cd prov
on-gagloev@omgagloev:~/prov$ touch file1.txt
on-gagloev@omgagloev:~/prov$ touch file2.txt
on-gagloev@omgagloev:~/prov$ touch file3.txt
on-gagloev@omgagloev:~/prov$ ls
file1.txt file2.txt file3.txt
on-gagloev@omgagloev:~/prov$ cd
on-gagloev@omgagloev:~$ bash ./prog4.sh
Enter the name of pac
newpac
Enter directory for packing
prov
The new directory files here:
file1.txt file2.txt file3.txt newpac.tar
on-gagloev@omgagloev:~$
```

строки. Это же командный файл, и убеждаюсь, что он исправно работает

Вывод:

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

Контрольные вопросы :

Она осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. При генерации имен файлов используют метасимволы:

" " - произвольная (возможно пустая) последовательность символов; "?" - один произвольный символ; "[...]" - любой из символов, указанных в скобках перечислением и/или с указанием диапазона; "cat f" - выдаст все файлы каталога, начинающиеся с "f"; "cat f" - выдаст все файлы, содержащие "f"; "cat program.?" выдаст файлы данного каталога с однобуквенными расширениями, скажем "program.c" и "program.o", но не выдаст "program.com"; "cat [a-d]" выдаст файлы, которые начинаются с "a", "b", "c", "d". Аналогичный эффект дадут и команды "cat [abcd]" и "cat [bdac]*". for, case, if, while Break, continue 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. Означает условие существования файла `man/s/i.$s` 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.