

Лабораторная работа номер 11

Программирование в командном процессоре ОС UNIX. Ветвления и
циклы

Мальков Роман

Содержание

Цель работы	3
Задание	4
Ход работы	6
Выводы	11

Цель работы

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

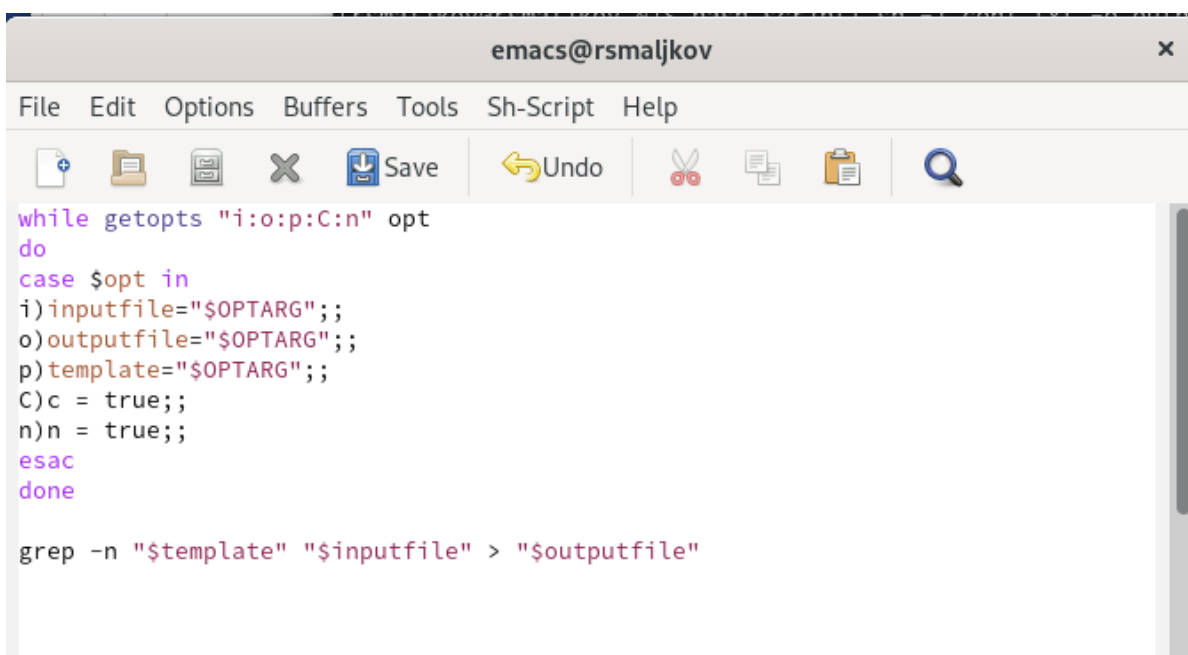
Задание

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

Ход работы

1. Код и результаты выполнения смотреть в скриншотах 1-2.

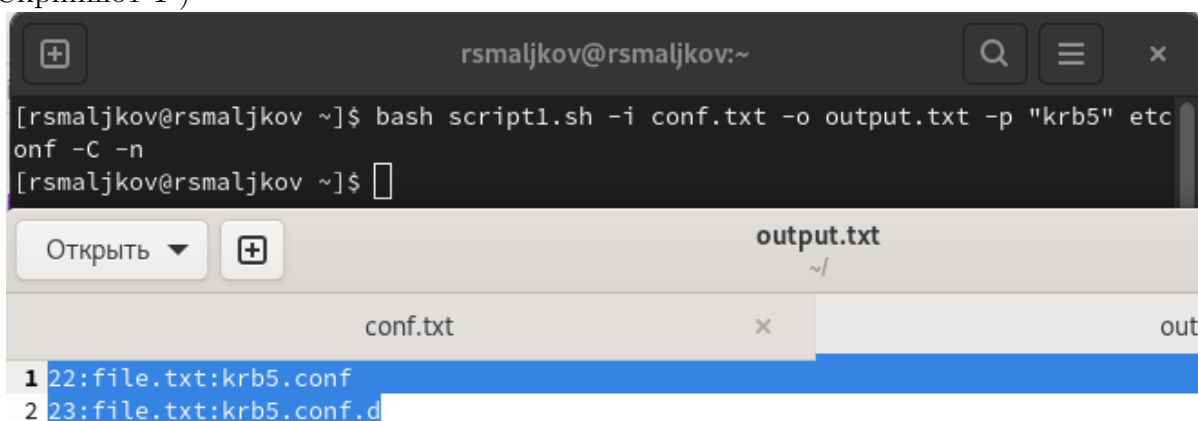


The screenshot shows an Emacs editor window titled "emacs@rsmaljkov". The menu bar includes "File", "Edit", "Options", "Buffers", "Tools", "Sh-Script", and "Help". The toolbar contains icons for opening files, saving, undo, redo, and search. The main text area contains the following shell script:

```
while getopts "i:o:p:C:n" opt
do
case $opt in
i)inputfile="$OPTARG";;
o)outputfile="$OPTARG";;
p)template="$OPTARG";;
C)c = true;;
n)n = true;;
esac
done

grep -n "$template" "$inputfile" > "$outputfile"
```

(Скриншот 1)

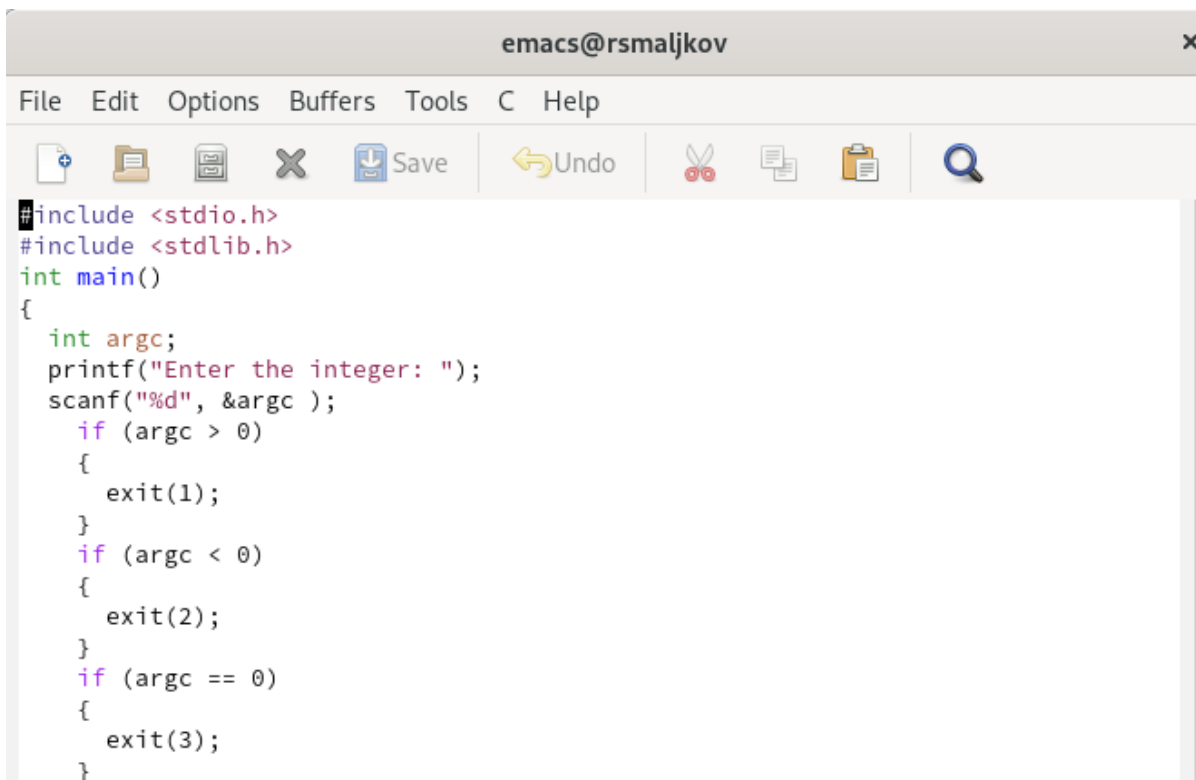


The screenshot shows a terminal window titled "rsmaljkov@rsmaljkov:~". The command executed is `bash script1.sh -i conf.txt -o output.txt -p "krb5" etc onf -C -n`. Below the terminal, a file manager window shows the contents of `output.txt`. The file contains two lines:

```
1 22:file.txt:krb5.conf
2 23:file.txt:krb5.conf.d
```

(Скриншот 2)

2. Код и результаты выполнения смотреть в скриншотах 3-5.



```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int argc;
    printf("Enter the integer: ");
    scanf("%d", &argc );
    if (argc > 0)
    {
        exit(1);
    }
    if (argc < 0)
    {
        exit(2);
    }
    if (argc == 0)
    {
        exit(3);
    }
}
```

(Скриншот 3)

```

1 #!/bin/bash
2
3 gcc script2.c
4
5 ./a.out
6
7 ec=$?
8
9 if [ $ec == 1 ]
10 then
11     echo "input > 0"
12 fi
13
14 if [ $ec == 2 ]
15 then
16     echo "input < 0"
17 fi
18
19 if [ $ec == 3 ]
20 then
21     echo "input = 0"
22 fi

```

(Скриншот 4)

```

[rsmaljkov@rsmaljkov ~]$ bash script2.sh
Enter the integer: 3
input > 0
[rsmaljkov@rsmaljkov ~]$

```

(Скриншот 5)

3. Код и результаты выполнения смотреть в скриншотах 6-8.

```

[rsmaljkov@rsmaljkov ~]$ bash script3.sh -c 4
[rsmaljkov@rsmaljkov ~]$ ls
1.tmp      catalogmc  lab07.sh~  output.txt  script2.c~  text.txt
2.tmp      conf.txt   main.c     Pictures    script2.sh  tmp
3.tmp      Desktop   main.lua   play        script2.sh~ Videos
4.tmp      Documents main.py    Public      script3.sh  work
abcl       Downloads may        reports     script3.sh~
a.out      features  monthly   script1.sh  scripts
australia  '#lab07.sh#' Music     script1.sh~ ski.places
backup     lab07.sh  my_os.bz2 script2.c   Templates

```

(Скриншот 6)


```
[rsmaljkov@rsmaljkov ~]$ bash script3.sh -r
find: './play/games/play': Отказано в доступе
[rsmaljkov@rsmaljkov ~]$ ls
abcl      Documents  main.lua   Pictures   script2.c~  Templates
a.out     Downloads  main.py    play       script2.sh  text.txt
australia features    may        Public     script2.sh~ tmp
backup    '#lab07.sh#' monthly    reports    script3.sh  Videos
catalogmc lab07.sh   Music      script1.sh script3.sh~ work
conf.txt  lab07.sh~ my_os.bz2  script1.sh~ scripts
Desktop   main.c     output.txt script2.c   ski.places
[rsmaljkov@rsmaljkov ~]$
```

(Скриншот 7)

```
Открыть  script3.sh ~/
1 #!/bin/bash
2
3 while getopts c:r opt
4 do
5 case $opt in
6 c)n="$OPTARG"; for i in $(seq 1 $n); do touch "$i.tmp"; done;;
7 r)for i in $(find -name "*.tmp"); do rm $i; done;;
8 esac
9 done
```

Скриншот 8)

4. Код и результаты выполнения смотреть в скриншотах 9 - 10.

```
Открыть  script4.sh ~/ Сохранить
1 while getopts ":p:" opt;
2 do
3 case $opt in
4 p)dir="$OPTARG"
5 esac
6 done
7 |
8 touch data.txt
9 find $dir -mtime -7 -mtime +0 -type f > data.txt
10 tar -cvf archive.tar -T data.txt
```

(Скриншот 9)

```
[rsmaljkov@rsmaljkov ~]$ bash script4.sh -p
find: './play/games/play': Отказано в доступе
./cache/mesa_shader_cache/8f/27cbf3bfed5ef11eaee6c84927e76d6453ea79
./cache/mesa_shader_cache/9b/044571a28c2d2079aa302a9906896ef6e5b4f5
./cache/mesa_shader_cache/6d/2c366683b541c4ae184bab4696fbb321eb8705
./cache/mesa_shader_cache/c0/c27bf446c1124e75b5832729b34a8e3a851b4b
./cache/mesa_shader_cache/90/a67a710bd82657e14dc609cd60dedc10140fd6
./cache/mesa_shader_cache/ab/1de095bd43ddf464a644bbd3c525c9bc322e1c
./cache/mesa_shader_cache/93/c9a9fbbb64fc6eeb9a4c38312164bea430a9eb
./cache/mesa_shader_cache/9a/44d0b9087bfaf1eef1141e1eb07307fc27692
./cache/mesa_shader_cache/64/cab41ec1a8a2a79267749ea8b413e0934f4d64
```

(Скриншот 10)

Выводы

Мы изучили основы программирования в оболочке ОС UNIX. Научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.