

# **Лабораторная работа №13**

**Дисциплина: Операционные системы**

Комягин Андрей Николаевич

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
<b>3</b>	<b>Контрольные вопросы</b>	<b>16</b>
<b>4</b>	<b>Вывод</b>	<b>17</b>

## Список иллюстраций

2.1	Создание файлов . . . . .	6
2.2	Код на с# . . . . .	7
2.3	Программа 2 . . . . .	7
2.4	Запуск 2 . . . . .	8
2.5	Программа 1 . . . . .	9
2.6	Запуск 1 . . . . .	12
2.7	Создание файла и наделение правами . . . . .	13
2.8	Программа 3 . . . . .	13
2.9	Запуск 3 . . . . .	14
2.10	Программа 4 . . . . .	15
2.11	Запуск 4 . . . . .	15

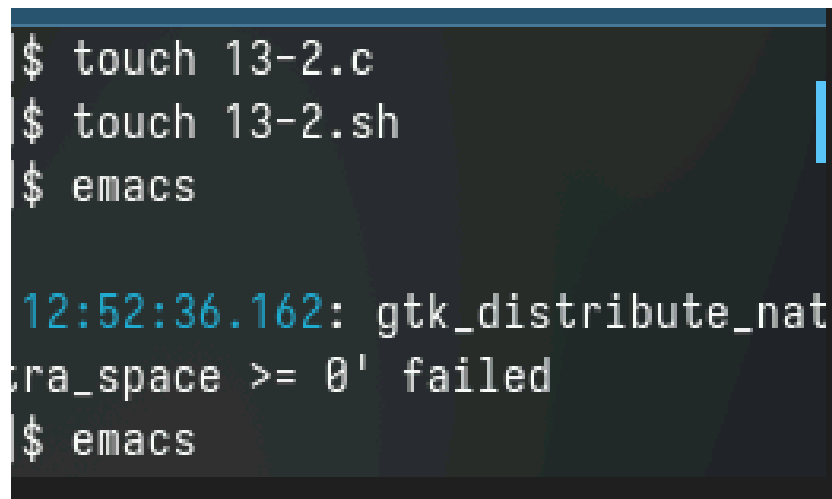
## **Список таблиц**

# 1 Цель работы

Изучение основ программирования в оболочке ОС UNIX. Приобретение навыков в написании более сложных командных файлов с использованием логических управляющих конструкций и циклов.

## 2 Выполнение лабораторной работы

Начнем выполнение задания со 2 пункта. Для этого создадим 2 файла - назовем их 13-2.sh и 13-2.c, запустим редактор emacs для написания самого кода(рис. 2.1). Не забудем прописать команду `chmod +x` для наделения файла правом на исполнение.

A terminal window with a dark background and light blue text. The commands entered are: \$ touch 13-2.c, \$ touch 13-2.sh, and \$ emacs. The output shows a timestamp and a message: 12:52:36.162: gtk\_distribute\_nat...ra\_space >= 0' failed, followed by another \$ emacs command.

```
$ touch 13-2.c
$ touch 13-2.sh
$ emacs

12:52:36.162: gtk_distribute_nat
ra_space >= 0' failed
$ emacs
```

Рис. 2.1: Создание файлов

Реализуем задание - напишем код на `c#` и создадим командный файл, анализирующий этот код (рис. 2.2) (рис. 2.3).

```

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

int main(){
    printf("Введите число");
    int x;
    scanf("%d", &x);
    if (x<0) exit (2);
    if (x>0) exit (1);
    if (x==0) exit (0);
    return 0;
}

```

Рис. 2.2: Код на с#

```

#!/bin/bash
gcc 13-2.c -o 13-2
./13-2
case $? in
    0) echo "Число равно нулю";;
    1) echo "Число больше нуля";;
    2) echo "Число меньше нуля";;
esac

```

Рис. 2.3: Программа 2

## Программа 2

```

#!/bin/bash
gcc 13-2.c -o 13-2
./13-2
case $? in
    0) echo "Число равно нулю";;
    1) echo "Число больше нуля";;

```

```
2) echo "Число меньше нуля";;  
esac
```

Запустим файл и проверим работу программы(рис. 2.4).

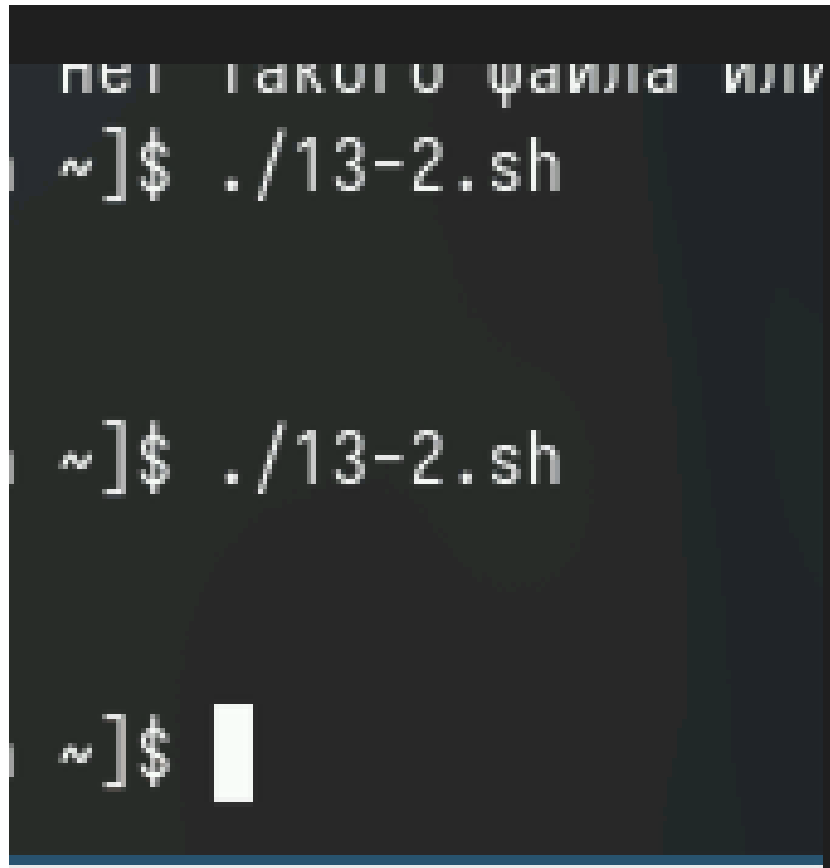


Рис. 2.4: Запуск 2

Далее перейдем к пункту 1.

Также создадим файл, назовем его 13-1.sh и наделим его правами на исполнение. Далее создадим 2 текстовых файла(в один добавим текст, а второй нужен для записи) и перейдем к написанию программы(рис. 2.5).



```
#!/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 "Not found"
else
    if (($iflag==0))
    then echo "File not found"
    else
        if((oflag==0))
        then if (($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))
```

Рис. 2.5: Программа 1

### Программа 1

```
#!/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 (($pflag==0))
then echo "Not found"
else
    if (($iflag==0))
    then echo "File not found"
    else
        if((oflag==0))
        then if (($Cflag==0))
            then if ((nflag==0))
                then grep $pval $inal
                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 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
    fi
fi
fi
fi
```

Запустим наш файл и посмотрим на вывод(рис. 2.6).

```
$ touch f1.txt f2.  
  
$ gedit f1.txt  
$ chmod +x 13-1.sh  
$ ./13-1.sh -i f1.  
  
$ cat f2.txt  
п корабль на мель,  
мели ели.  
  
$ gedit f1.txt  
$ ./13-1.sh -i f1.  
  
$ cat f2.txt  
  
$
```

Рис. 2.6: Запуск 1

Затем создадим файл для 3й программы. Наделим правами доступ и перейдем к написанию кода(рис. 2.7) и (рис. 2.8).

```
touch 13-3.sh
emacs
chmod +x 13-3.sh
./13-3.sh 3
```

Рис. 2.7: Создание файла и наделение правами

```
#!/bin/bash
for((i=1; i<=$*; i++))
do
    if test -f "$i".tmp
    then rm "$i".tmp
    else touch "$i".tmp
    fi
done
```

Рис. 2.8: Программа 3

### Программа 3

```
#!/bin/bash
for((i=1; i<=$*; i++))
do
```

```

if test -f "$i".tmp
then rm "$i".tmp
else touch "$i".tmp
fi
done

```

Запустим программу, убедимся, что все работает корректно - файлы создаются и удаляются(рис. 2.9).

```

13-1.sh 13-2.sh~ abc1 Downloads '#lab06.sh#
13-1.sh~ 13-3.sh backup f1.txt '#lab07.sh#
13-2 13-3.sh~ code.cpp f2.txt lab07.sh
13-2.c 1.tmp config feathers LICENSE
13-2.c~ 2.tmp conf.txt file.txt may
13-2.sh 3.tmp Documents git-extended monthly
edzhibitskaya@edzhibitskaya ~]$ ./13-3.sh 3
edzhibitskaya@edzhibitskaya ~]$ ls
13-1.sh 13-2.sh~ config feathers LICENSE
13-1.sh~ 13-3.sh conf.txt file.txt may
13-2 13-3.sh~ Documents git-extended monthly
13-2.c abc1 Downloads '#lab06.sh# my_os
13-2.c~ backup f1.txt '#lab07.sh# newdir
13-2.sh code.cpp f2.txt lab07.sh noname.gv

```

Рис. 2.9: Запуск 3

Наконец перейдем к 4 пункту. Выполним те же действия, создав и написав программу для исполняемого файла(рис. 2.10) и (рис. 2.11).



Рис. 2.10: Программа 4

#### Программа 4

```
#!/bin/bash  
find $* -mtime -7 -type f > FILES.txt  
tar -cf archive.tar -T FILES.txt
```



Рис. 2.11: Запуск 4

В результате запуска создается архив, с данными из указанного каталога, причем с изменениями до 7 дней.

### 3 Контрольные вопросы

1. Команда `getopts` используется для обработки аргументов командной строки в скриптах Shell, чтобы сделать их более гибкими и управляемыми.
2. Метасимволы используются для шаблонного поиска и обработки файлов в командной оболочке UNIX. Они позволяют задавать шаблоны для поиска файлов с определенными именами или расширениями.
3. Операторы управления действиями включают в себя условные операторы (`if-then-else`), операторы цикла (`for`, `while`, `until`), операторы выбора (`case`).
4. Для прерывания цикла используются операторы `break` и `continue`.
5. Команда `false` возвращает ложное значение (код ошибки), а команда `true` - истинное значение (код успеха). Они могут быть использованы для управления потоком выполнения в скриптах.
6. Строка `if test -f mans/i.$s` проверяет, существует ли файл с именем, указанным в переменных `$s` и `$i`. Опция `-f` утилиты `test` указывает на проверку существования обычного файла.
7. Конструкция `while` выполняет цикл до тех пор, пока условие истинно, а конструкция `until` выполняет цикл до тех пор, пока условие ложно. Разница заключается в том, что в `while` условие проверяется перед выполнением тела цикла, а в `until` - после.



## 4 Вывод

В ходе работы мы изучили основные программирования в оболочке UNIX, написали несколько более сложных командных файлов.