

Лабораторная работа № 13

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Мальянц Виктория Кареновна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Задание № 1	7
3.2	Задание № 2	11
3.3	Задание № 3	14
3.4	Задание № 4	15
4	Выводы	17
5	Контрольные вопросы	18
	Список литературы	19

Список иллюстраций

3.1	Создание файлов input.txt и output.txt, открытие файла input.txt . .	7
3.2	Редактирование файла	8
3.3	Создание файла lab13-1.sh и открытие его	8
3.4	Редактирование файла	9
3.5	Право на исполнение lab13-1.sh и запуск этого файла	10
3.6	Открытие файла output.txt	10
3.7	Файл output.txt, открытый в редакторе gedit	11
3.8	Создание файла lab13_2.c и открытие его	11
3.9	Редактирование файла	12
3.10	Создание файла lab13-2.sh и открытие его	12
3.11	Редактирование файла	13
3.12	Право на исполнение lab13-2.sh и запуск этого файла	14
3.13	Создание файла lab13-3.sh и открытие его	14
3.14	Редактирование файла	14
3.15	Право на исполнение lab13-3.sh и запуск этого файла	15
3.16	Создание файла lab13-4.sh и открытие его	15
3.17	Редактирование файла	16
3.18	Право на исполнение lab13-3.sh и запуск этого файла	16
3.19	Список содержимого домашнего каталога	16

Список таблиц

1 Цель работы

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

2 Задание

1. 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. Задание № 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. Задание № 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. Задание № 4. Написать командный файл, который с помощью команды `tag` 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`).
5. Контрольные вопросы

3 Выполнение лабораторной работы

3.1 Задание № 1

Создаю файлы input.txt и output.txt, открываю файл input.txt (рис. 3.1).

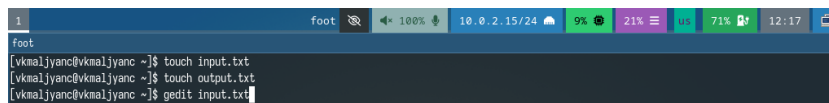
A screenshot of a terminal window. The window has a title bar with 'foot' and system status icons. The terminal shows three commands being executed: 'touch input.txt', 'touch output.txt', and 'gedit input.txt'. The prompt is '[vkmajjanc@vkmajjanc ~]\$'.

Рис. 3.1: Создание файлов input.txt и output.txt, открытие файла input.txt

Ввожу текст в файл input.txt (рис. 3.2).

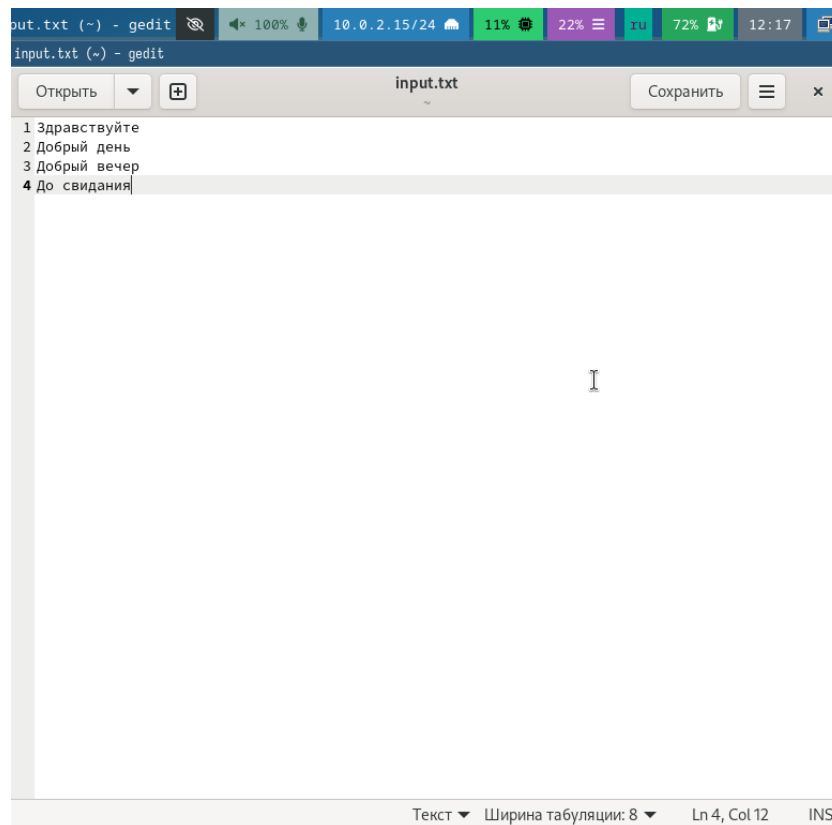


Рис. 3.2: Редактирование файла

Создаю файл lab13-1.sh и открываю его (рис. 3.3).



Рис. 3.3: Создание файла lab13-1.sh и открытие его

Ввожу код в файл lab13-1.sh (рис. 3.4).

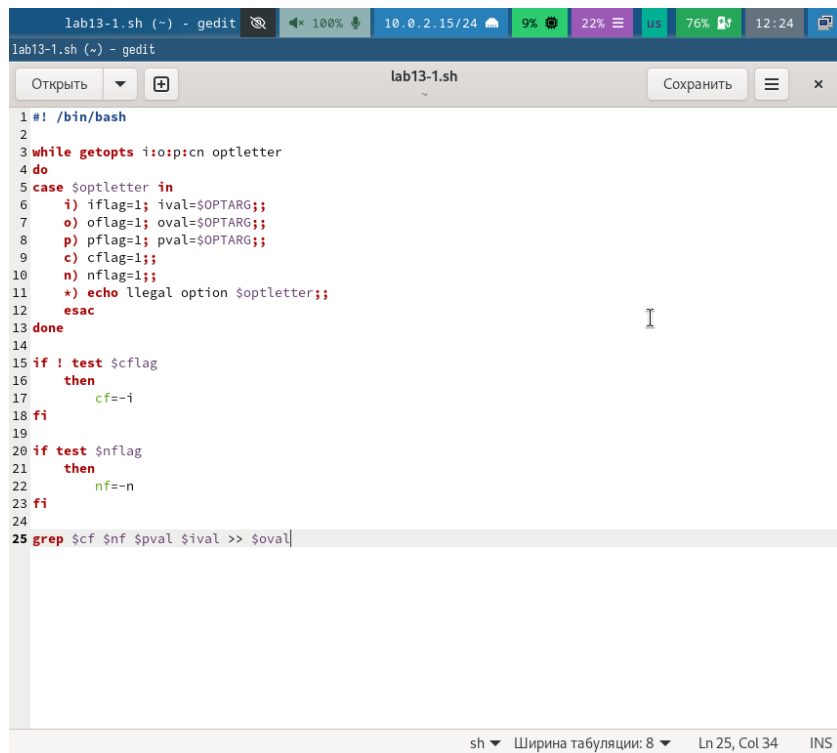


Рис. 3.4: Редактирование файла

Листинг программы lab13-1.sh:

```

#!/bin/bash

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

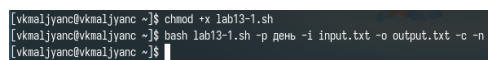
```

```
if ! test $cflag
then
    cf=-i
fi
```

```
if test $nflag
then
    nf=-n
fi
```

```
grep $cf $nf $pval $ival >> $oval
```

Даю право на исполнение файла lab13-1.sh и запускаю его (рис. 3.5).



```
[vkmajyanc@vkmajyanc ~]$ chmod +x lab13-1.sh
[vkmajyanc@vkmajyanc ~]$ bash lab13-1.sh -p день -i input.txt -o output.txt -c -n
[vkmajyanc@vkmajyanc ~]$
```

Рис. 3.5: Право на исполнение lab13-1.sh и запуск этого файла

Открываю файл output.txt (рис. 3.6).



```
[vkmajyanc@vkmajyanc ~]$ gedit output.txt
```

Рис. 3.6: Открытие файла output.txt

Убеждаюсь в том, что программа работает корректно (рис. 3.7).

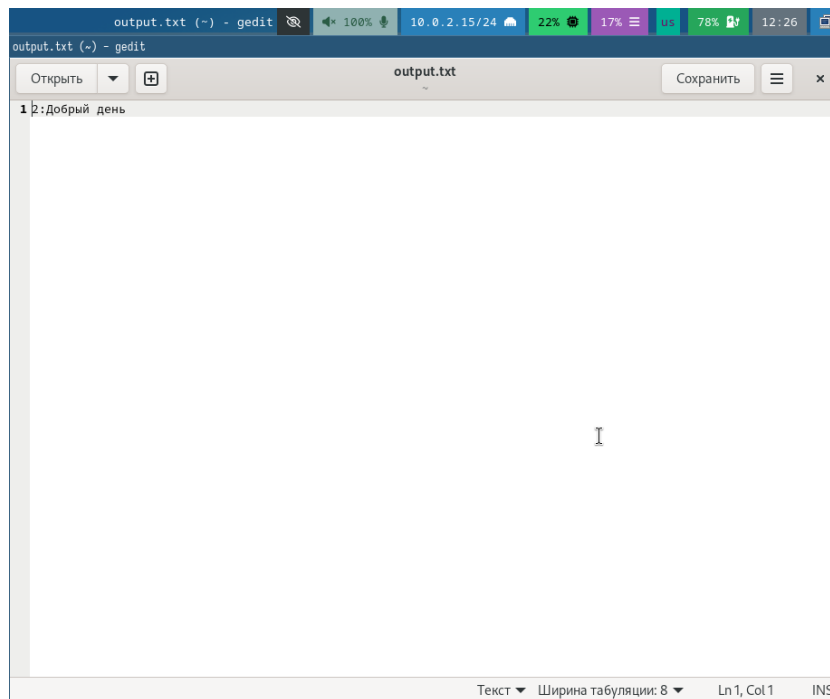


Рис. 3.7: Файл output.txt, открытый в редакторе gedit

3.2 Задание № 2

Создаю файл lab13_2.c и открываю его (рис. 3.8).

```
[vkmajjyanc@vkmajjyanc ~]$ touch lab13_2.c  
[vkmajjyanc@vkmajjyanc ~]$ gedit lab13_2.c
```

Рис. 3.8: Создание файла lab13_2.c и открытие его

Ввожу код в файл lab13_2.c (рис. 3.9).

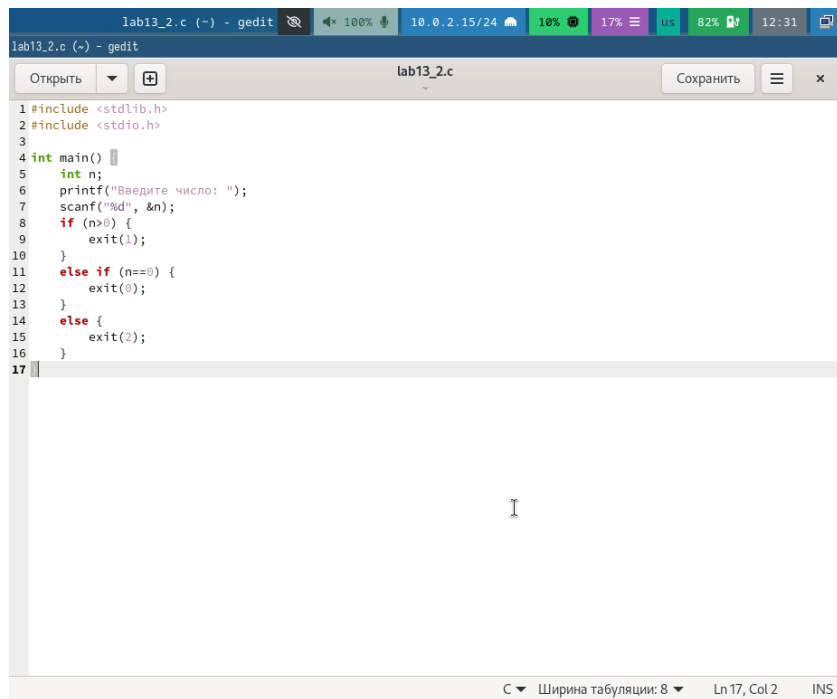


Рис. 3.9: Редактирование файла

Создаю файл lab13-2.sh и открываю его (рис. 3.10).



Рис. 3.10: Создание файла lab13-2.sh и открытие его

Ввожу код в файл lab13-2.sh (рис. 3.11).

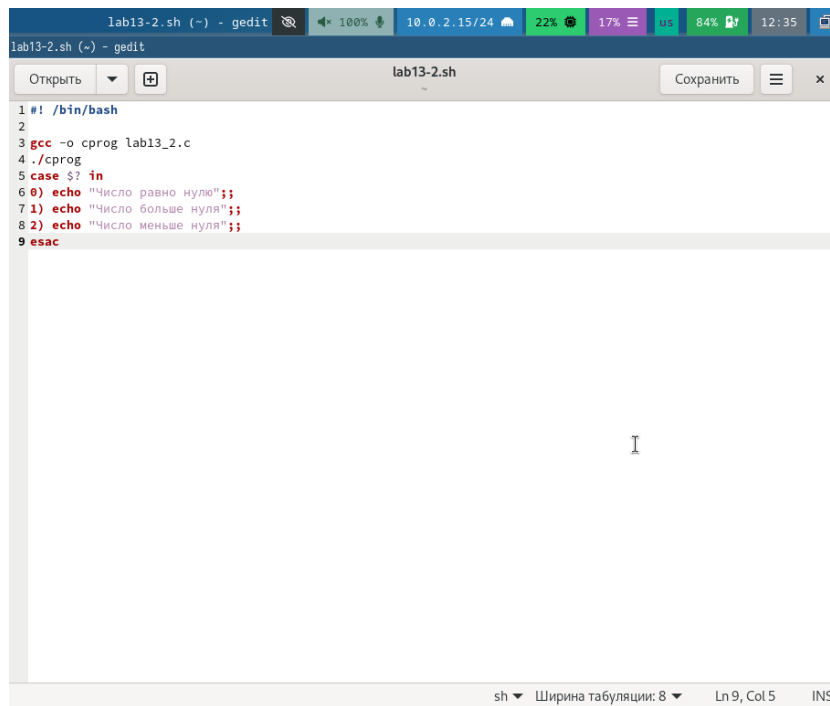


Рис. 3.11: Редактирование файла

Листинг программы lab13-2.sh:

```
#!/bin/bash

gcc -o cprog lab13_2.c
./cprog
case $? in
0) echo "Число равно нулю";;
1) echo "Число больше нуля";;
2) echo "Число меньше нуля";;
esac
```

Даю право на исполнение файла lab13-2.sh и запускаю его. Ввожу число 10. Убеждаюсь в том, что программа работает корректно (рис. 3.12).

```
[vkmajjyanc@vkmajjyanc ~]$ chmod +x lab13-2.sh
[vkmajjyanc@vkmajjyanc ~]$ bash lab13-2.sh
Введите число: 10
Число больше нуля
[vkmajjyanc@vkmajjyanc ~]$
```

Рис. 3.12: Право на исполнение lab13-2.sh и запуск этого файла

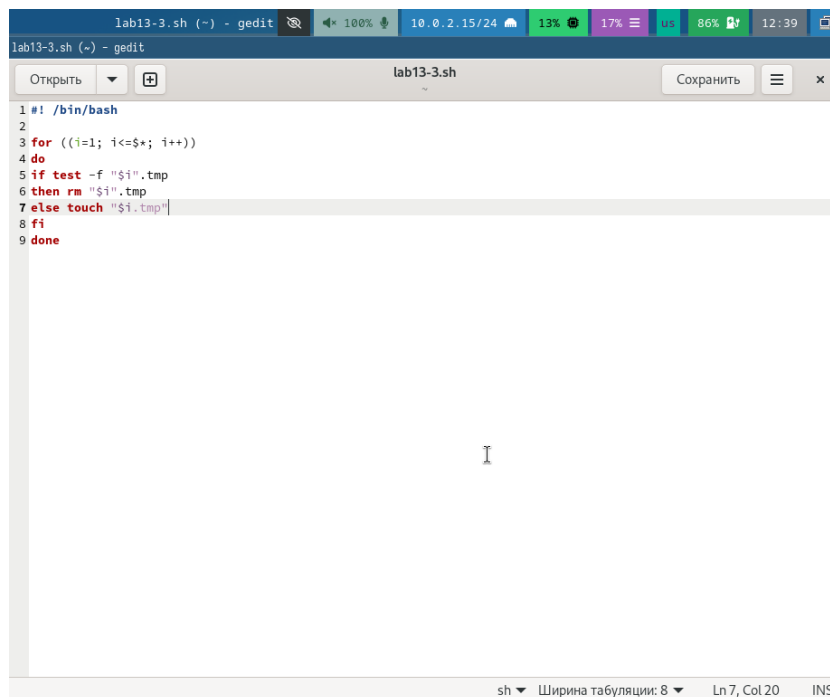
3.3 Задание № 3

Создаю файл lab13-3.sh и открываю его (рис. 3.13).

```
[vkmajjyanc@vkmajjyanc ~]$ touch lab13-3.sh
[vkmajjyanc@vkmajjyanc ~]$ gedit lab13-3.sh
```

Рис. 3.13: Создание файла lab13-3.sh и открытие его

Ввожу код в файл lab13-3.sh (рис. 3.14).



```
lab13-3.sh (~) - gedit
lab13-3.sh (~) - gedit
Открыть lab13-3.sh Сохранить x
1 #! /bin/bash
2
3 for ((i=1; i<=20; i++))
4 do
5 if test -f "$i".tmp
6 then rm "$i".tmp
7 else touch "$i.tmp"
8 fi
9 done
sh Ширина табуляции: 8 Ln 7, Col 20 INS
```

Рис. 3.14: Редактирование файла

Листинг программы lab13-3.sh:

```
#!/bin/bash
```

```

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

```

Даю право на исполнение файла lab13-3.sh и запускаю его. Программа создала пять файлов: 1.tmp, 2.tmp, 3.tmp, 4.tmp, 5.tmp. После повторного ввода команды программа удалила созданные файлы. Убеждаюсь в том, что программа работает корректно (рис. 3.15).

Рис. 3.15: Право на исполнение lab13-3.sh и запуск этого файла

3.4 Задание № 4

Создаю файл lab13-4.sh и открываю его (рис. 3.16).

Рис. 3.16: Создание файла lab13-4.sh и открытие его

Ввожу код в файл lab13-4.sh (рис. 3.17).

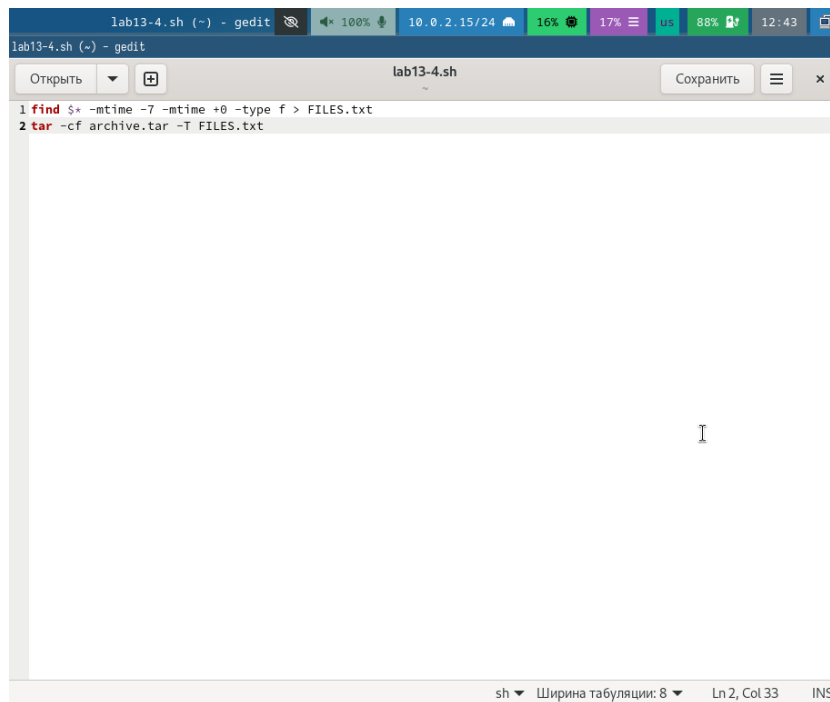


Рис. 3.17: Редактирование файла

Листинг программы lab13-4.sh:

```
find $* -mtime -7 -mtime +0 -type f > FILES.txt
tar -cf archive.tar -T FILES.txt
```

Даю право на исполнение файла lab13-3.sh и запускаю его. Программа создает архив с файлами из каталога ski.places (рис. 3.18).

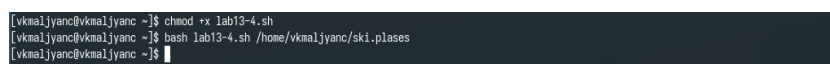


Рис. 3.18: Право на исполнение lab13-3.sh и запуск этого файла

Убеждаюсь в том, что программа работает корректно (рис. 3.19) [1].

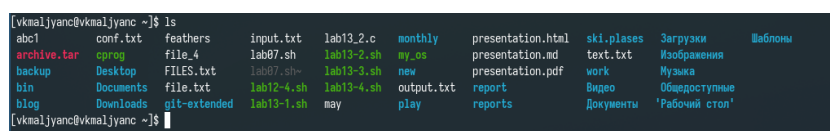


Рис. 3.19: Список содержимого домашнего каталога

4 Выводы

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

5 Контрольные вопросы

1. Команда `getopts` используется в скриптах оболочки для разбора опций командной строки.
2. Метасимволы используются оболочкой для сопоставления шаблонов имен файлов.
3. Условные операторы (`if`, `then`, `else`, `fi`, `case`, `esac`), циклы (`for`, `while`, `until`), операторы объединения команд (`;`, `&&`, `||`).
4. Для прерывания цикла используются `break` и `continue`.
5. `false` - используется для команды, которая завершается всегда неудачно.
`true` - используется для команды, которая всегда выполняется успешно.
6. Строка `if test -f mans/i.$$` проверяет, существует ли файл.
7. `while` - выполняется блок кода, пока условие истинно. `until` - выполняет блок кода, пока условие ложно. Различие `while` и `until` состоит в том, что они логически противоположны друг другу.

Список литературы

1. Лабораторная работа № 13.