

Лабораторная работа №13

Презентация

Филиппьева К.Д.

04 мая 2024

Российский университет дружбы народов, Москва, Россия

Информация

- Филиппева Ксения Дмитриевна
- Студент
- Российский университет дружбы народов
- 1132230795@pfur.ru

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

1. Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами: `-i` `inputfile` — прочитать данные из указанного файла; `-o` `outputfile` — 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 `-r`.

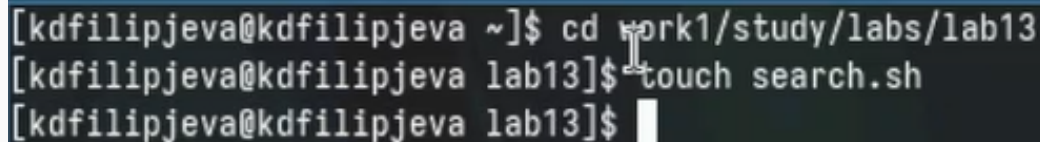
2. Написать на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции `exit(n)`, передавая информацию в о коде завершения в оболочку. Команд- ный файл должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено.

3. Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до ∞ (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`).

Первое задание

Создадим файл для первого задания

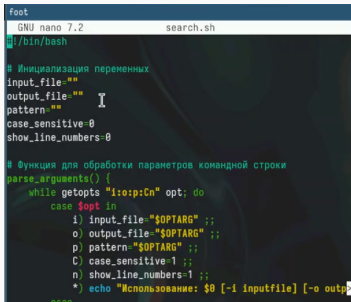
A terminal window with a dark background and light gray text. It shows three lines of commands and their outputs. The first line shows a user changing the directory to 'work1/study/labs/lab13'. The second line shows the user creating a file named 'search.sh' using the 'touch' command. The third line shows the prompt after the command has executed successfully. A white cursor is visible at the end of the third line.

```
[kdfilipjeva@kdfilipjeva ~]$ cd work1/study/labs/lab13  
[kdfilipjeva@kdfilipjeva lab13]$ touch search.sh  
[kdfilipjeva@kdfilipjeva lab13]$
```

Рис. 1: создание файла

Промежуточный результат

Введем в него код для первого задания



```
foot
GNU nano 7.2      search.sh
#!/bin/bash

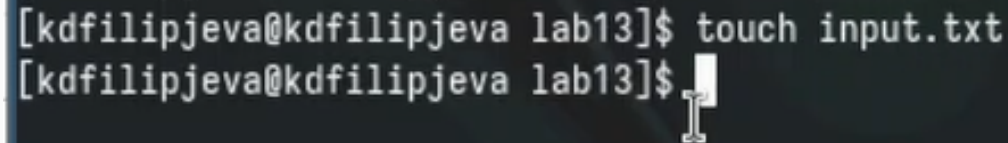
# Инициализация переменных
input_file=""
output_file=""
pattern=""
case_sensitive=0
show_line_numbers=0

# Функция для обработки параметров командной строки
parse_arguments() {
    while getopts "i:o:p:Cn" opt; do
        case $opt in
            i) input_file="$OPTARG" ;;
            o) output_file="$OPTARG" ;;
            p) pattern="$OPTARG" ;;
            C) case_sensitive=1 ;;
            n) show_line_numbers=1 ;;
            *) echo "Использование: $0 [-i inputfile] [-o outputfile] [-p pattern] [-C] [-n]" ;;
        esac
    done
}
```

Рис. 2: код

Промежуточный результат

Создадим файл в который будем вводить текст, с которым будет работать программа

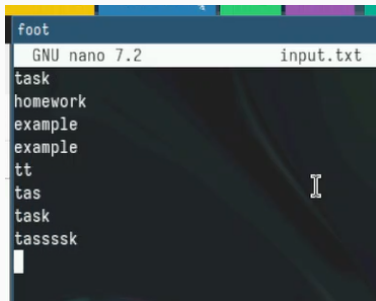
A screenshot of a terminal window with a dark background and light-colored text. The prompt is '[kdfilipjeva@kdfilipjeva lab13]\$'. The command 'touch input.txt' has been entered. The prompt is repeated on the next line, and a white cursor is visible at the end of the second line.

```
[kdfilipjeva@kdfilipjeva lab13]$ touch input.txt  
[kdfilipjeva@kdfilipjeva lab13]$
```

Рис. 3: создание файла

Промежуточный результат

Текст для работы программы

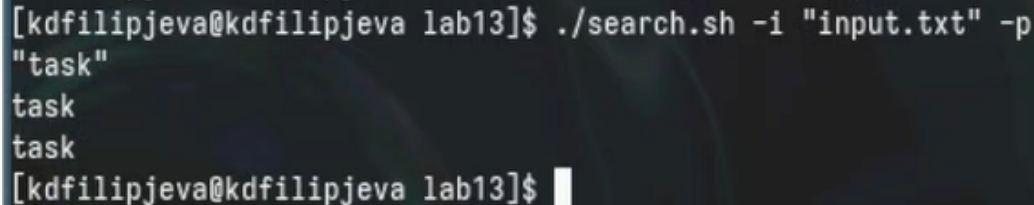


```
foot
GNU nano 7.2 input.txt
task
homework
example
example
tt
tas
task
tasssk
|
```

Рис. 4: код

Промежуточный результат

Вывод найденного текста по установленному шаблону в командную строку

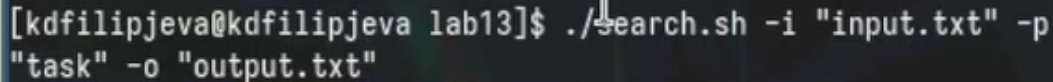
A terminal window with a dark background and light-colored text. The prompt is [kdfilipjeva@kdfilipjeva lab13]\$. The command entered is ./search.sh -i "input.txt" -p "task". The output shows the word "task" on two separate lines. The prompt returns to [kdfilipjeva@kdfilipjeva lab13]\$.

```
[kdfilipjeva@kdfilipjeva lab13]$ ./search.sh -i "input.txt" -p  
"task"  
task  
task  
[kdfilipjeva@kdfilipjeva lab13]$
```

Рис. 5: вывод результата

Промежуточный результат

Выведем найденный текст в отдельный файл



```
[kdfilipjeva@kdfilipjeva lab13]$ ./search.sh -i "input.txt" -p  
"task" -o "output.txt"
```

Рис. 6: вывод результата

Промежуточный результат

Выведенный текст в отдельном файле

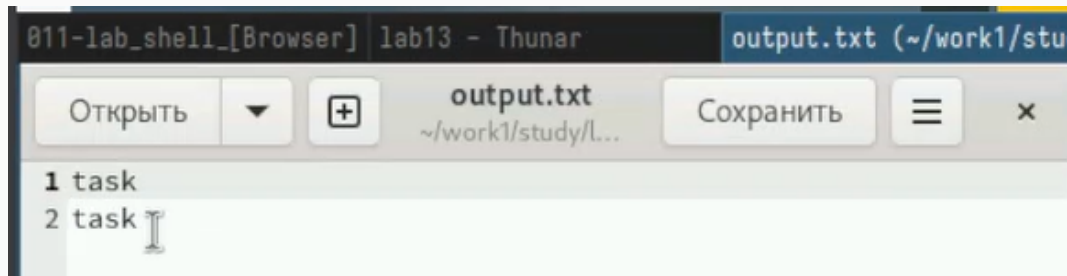
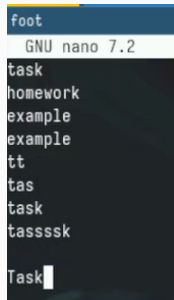


Рис. 7: вывод результата

Промежуточный результат

Отредактируем текст для работы программы

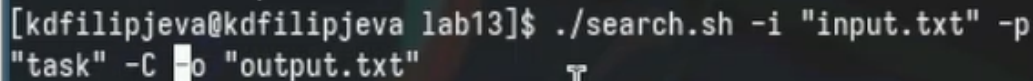


```
foot
GNU nano 7.2
task
homework
example
example
tt
tas
task
tasssk
Task
```

Рис. 8: вывод результата

Промежуточный результат

Выведем текст в файл с учетом регистра



```
[kdfilipjeva@kdfilipjeva lab13]$ ./search.sh -i "input.txt" -p  
"task" -C -o "output.txt"
```

Рис. 9: вывод результата

Промежуточный результат

Вывод текста с учетом регистра(видно, что вывело только с маленькой буквы, а заглавную не тронуло)

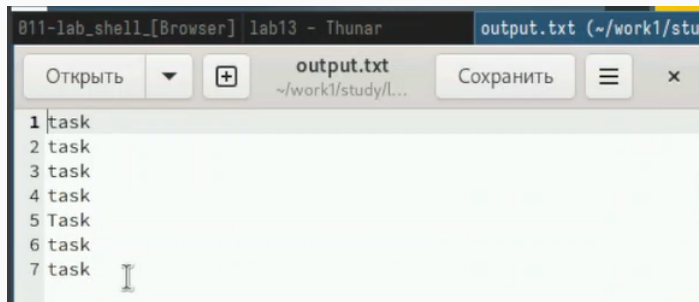
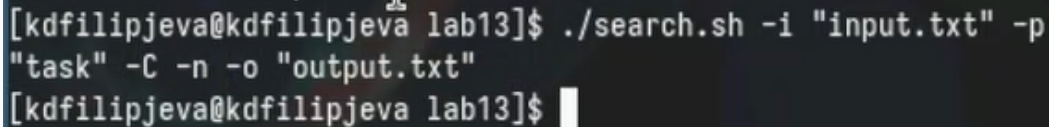


Рис. 10: вывод результата

Промежуточный результат

Выведем текст с учетом регистра и нумерацией строк, из которых было взято слово

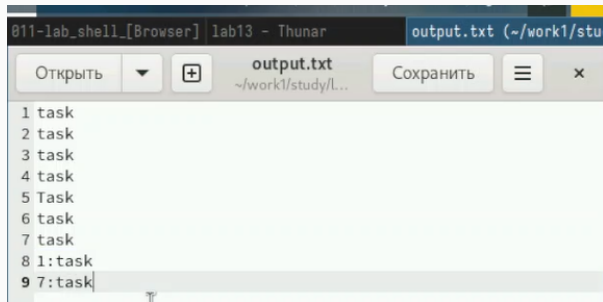
A terminal window with a dark background and light-colored text. The prompt is [kdfilipjeva@kdfilipjeva lab13]\$. The command ./search.sh -i "input.txt" -p "task" -C -n -o "output.txt" is entered on the first line. The second line shows the prompt again with a cursor, indicating the command has finished execution.

```
[kdfilipjeva@kdfilipjeva lab13]$ ./search.sh -i "input.txt" -p  
"task" -C -n -o "output.txt"  
[kdfilipjeva@kdfilipjeva lab13]$
```

Рис. 11: вывод результата

Промежуточный результат

Вывод текста с нумерацией



The screenshot shows a text editor window titled "output.txt (~/.work1/stu)". The window has a menu bar with "Открыть", "Сохранить", and a hamburger menu icon. The text area contains the following content:

```
1 task
2 task
3 task
4 task
5 Task
6 task
7 task
8 1:task
9 7:task
```

Рис. 12: вывод результата

Второе задание

Создадим файл для второго задания и выдадим права на выполнение

foot

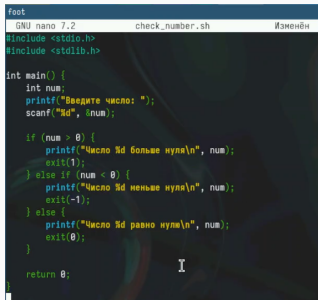
```
[kdfilipjeva@kdfilipjeva lab13]$ touch check_number.sh
```

```
[kdfilipjeva@kdfilipjeva lab13]$ chmod 777 check_number.sh
```

Рис. 13: создание файла

Промежуточный результат

Вставим код программы, который ответственен за определение числа



```
foot
GNU nano 7.2      check_number.sh      Изменён
#include <stdio.h>
#include <stdlib.h>

int main() {
    int num;
    printf("Введите число: ");
    scanf("%d", &num);

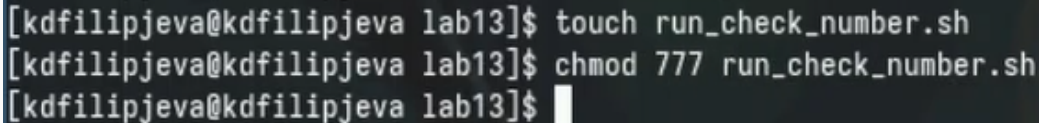
    if (num > 0) {
        printf("Число %d больше нуля\n", num);
        exit(1);
    } else if (num < 0) {
        printf("Число %d меньше нуля\n", num);
        exit(-1);
    } else {
        printf("Число %d равно нулю\n", num);
        exit(0);
    }

    return 0;
}
```

Рис. 14: код программы

Промежуточный результат

Создадим файл для второго задания, который будет “общаться с пользователем”

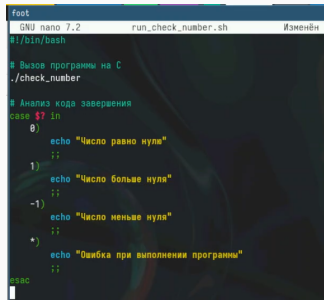
A terminal window with a dark background and light-colored text. It shows three lines of commands being executed in a shell. The first line creates a file named 'run_check_number.sh' using the 'touch' command. The second line sets permissions of '777' for the same file using the 'chmod' command. The third line shows the prompt with a cursor, indicating the user is ready for the next command.

```
[kdfilipjeva@kdfilipjeva lab13]$ touch run_check_number.sh  
[kdfilipjeva@kdfilipjeva lab13]$ chmod 777 run_check_number.sh  
[kdfilipjeva@kdfilipjeva lab13]$
```

Рис. 15: создание файла

Промежуточный результат

Вставим в него код программы



```
foot
GNU nano 7.2      run_check_number.sh      Изменён
#!/bin/bash

# Вызов программы на C
./check_number

# Анализ кода завершения
case $? in
  0)
    echo "Число равно нулю"
    ;;
  1)
    echo "Число больше нуля"
    ;;
  -1)
    echo "Число меньше нуля"
    ;;
  *)
    echo "Ошибка при выполнении программы"
    ;;
esac
```

Рис. 16: код программы

Промежуточный результат

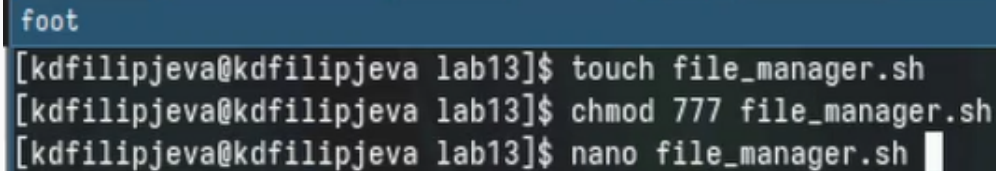
Скомпилируем наш код на языке Си и проверим работоспособность

```
[kdfilipjeva@kdfilipjeva lab13]$ gcc -o check_number check_number.c
[kdfilipjeva@kdfilipjeva lab13]$ ./run_check_number.sh
Введите число: 10
Число 10 больше нуля
Число больше нуля
[kdfilipjeva@kdfilipjeva lab13]$ ./run_check_number.sh
Введите число: 0
Число 0 равно нулю
Число равно нулю
[kdfilipjeva@kdfilipjeva lab13]$ ./run_check_number.sh
Введите число: -10
Число -10 меньше нуля
Ошибка при выполнении программы
[kdfilipjeva@kdfilipjeva lab13]$
```

Рис. 17: вывод результата

Третье задание

Создадим файл для третьего задания и выдадим ему права на выполнение

A terminal window with a dark background and light blue header. The header contains the text 'foot'. The terminal shows three lines of commands being executed in a shell. The first line is '[kdfilipjeva@kdfilipjeva lab13]\$ touch file_manager.sh'. The second line is '[kdfilipjeva@kdfilipjeva lab13]\$ chmod 777 file_manager.sh'. The third line is '[kdfilipjeva@kdfilipjeva lab13]\$ nano file_manager.sh' followed by a white cursor block.

```
foot
[kdfilipjeva@kdfilipjeva lab13]$ touch file_manager.sh
[kdfilipjeva@kdfilipjeva lab13]$ chmod 777 file_manager.sh
[kdfilipjeva@kdfilipjeva lab13]$ nano file_manager.sh
```

Рис. 18: создание файла

Промежуточный результат

Вставим в него необходимый код для выполнения задания

```
foot
GNU nano 7.2      file_manager.sh
#!/bin/bash

# Функция для создания файлов
create_files() {
    local num_files=$1
    for i in $(seq 1 $num_files); do
        touch "$i.tmp"
    done
    echo "Создано $num_files файлов"
}

# Функция для удаления файлов
delete_files() {
    local num_files=$1
    for i in $(seq 1 $num_files); do
        rm -f "$i.tmp"
    done
    echo "Удалено $num_files файлов"
}
```

Рис. 19: код программы

Работоспособность кода

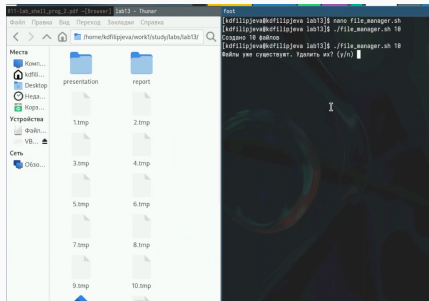


Рис. 20: вывод результата

Работоспособность кода в обратную сторону

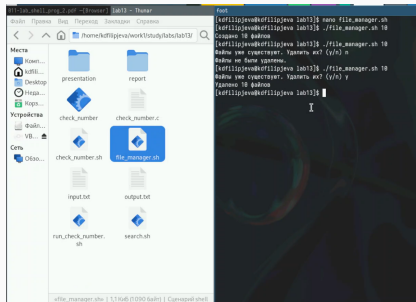


Рис. 21: вывод результата

Четвертое задание

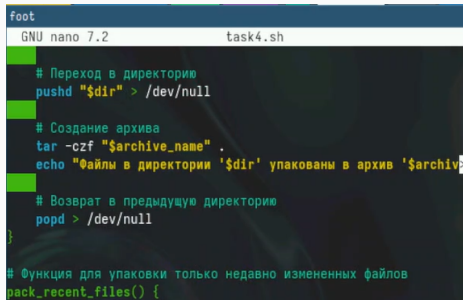
Создадим файл для четвертого задания и выдадим ему права на выполнение

```
foot
[kdfilipjeva@kdfilipjeva lab13]$ touch task4.sh
[kdfilipjeva@kdfilipjeva lab13]$ chmod 777 task4.sh
[kdfilipjeva@kdfilipjeva lab13]$ nano task4.sh
```

Рис. 22: создание файла

Промежуточный результат

Вставим в него необходимый код программы



```
foot
GNU nano 7.2 task4.sh

# Переход в директорию
pushd "$dir" > /dev/null

# Создание архива
tar -czf "$archive_name" .
echo "Файлы в директории '$dir' упакованы в архив '$archive_name'"

# Возврат в предыдущую директорию
popd > /dev/null
}

# Функция для упаковки только недавно измененных файлов
pack_recent_files() {
```

Рис. 23: код программы

Промежуточный результат

Работоспособность кода

```
[kdfilipjeva@kdfilipjeva lab13]$ nano task4.sh
[kdfilipjeva@kdfilipjeva lab13]$ ./task4.sh ~/work1/study/labs/lab13/report/ backup.tar.gz 7
tar: .: файл изменился во время чтения
Файлы в директории '/home/kdfilipjeva/work1/study/labs/lab13/report/' упакованы в архив 'backup.tar.gz'.
Файлы в директории '/home/kdfilipjeva/work1/study/labs/lab13/report/', измененные менее 7 дней назад, упакованы в архив 'backup.tar_recent_20240504223949.tar.gz'.
[kdfilipjeva@kdfilipjeva lab13]$
```

Рис. 24: вывод результата

Промежуточный результат

Созданные 2 архива: всей папки и только файлов, которые были изменены менее чем неделю назад

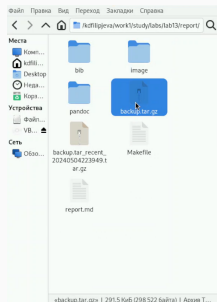


Рис. 25: вывод результата

Мы получили новые и отработали у##е имеющиеся навыки программирования в оболочке ОС Linux.