

# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук Кафедра прикладной информатики и теории вероятностей

## ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №13 =====

дисциплина: Операционные системы

Студент: Койфман Кирилл Дмитриевич

Группа: НПИбд-01-21

### Введение.

---

### Цель работы.

Приобрести простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования C калькулятора с простейшими функциями. ## Задачи. 1. В домашнем каталоге создать подкаталог `~/work/os/lab_prog`.

2. Создать в нём файлы: `calculate.h`, `calculate.c`, `main.c`. Это будет примитивнейший калькулятор, способный складывать, вычитать, умножать и делить, возводить число в степень, брать квадратный корень, вычислять `sin`, `cos`, `tan`. При запуске он будет запрашивать первое число, операцию, второе число. После этого программа выведет результат и остановится.

3. Выполнить компиляцию программы посредством `gcc`:

```
gcc -c calculate.c
gcc -c main.c
gcc calculate.o main.o -o calcul -lm
```

4. При необходимости исправить синтаксические ошибки.

5. Создать Makefile со следующим содержанием:

```
#
# Makefile
#
```

```
CC = gcc
CFLAGS =
```

```
LIBS = -lm

calcul: calculate.o main.o
gcc calculate.o main.o -o calcul $(LIBS)

calculate.o: calculate.c calculate.h
gcc -c calculate.c $(CFLAGS)

main.o: main.c calculate.h
gcc -c main.c $(CFLAGS)

clean:
-rm calcul *.o *~

# End Makefile
```

Пояснить в отчёте его содержание.

6. С помощью gdb выполнить отладку программы calcul (перед использованием gdb исправить Makefile).
7. С помощью утилиты splint попробовать проанализировать коды файлов calculate.c и main.c.

### Теоретическое введение.

Процесс разработки программного обеспечения обычно разделяется на следующие этапы:

- планирование, включающее сбор и анализ требований к функционалу и другим характеристикам разрабатываемого приложения;
- проектирование, включающее в себя разработку базовых алгоритмов и спецификаций, определение языка программирования;
- непосредственная разработка приложения:
- кодирование — по сути создание исходного текста программы (возможно в нескольких вариантах);
- анализ разработанного кода;
- сборка, компиляция и разработка исполняемого модуля;
- тестирование и отладка, сохранение произведённых изменений;
- документирование.

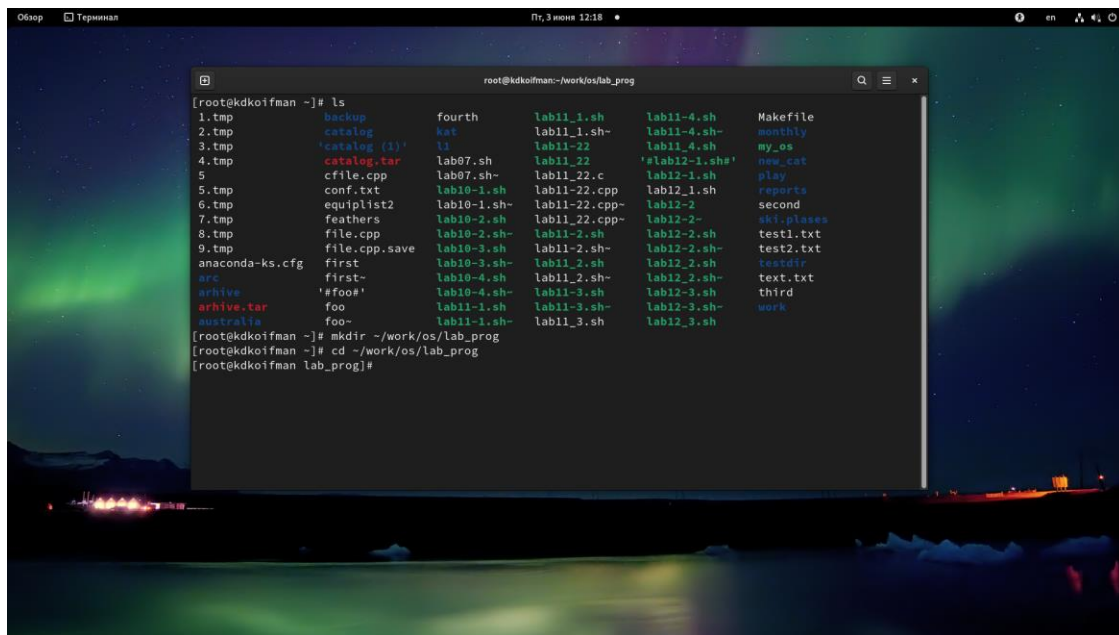
Для создания исходного текста программы разработчик может воспользоваться любым удобным для него редактором текста: vi, vim, mceditor, emacs, geany и др. После завершения написания исходного кода программы (возможно состоящей из

нескольких файлов), необходимо её скомпилировать и получить исполняемый модуль.

## Ход работы.

### 1 задание.

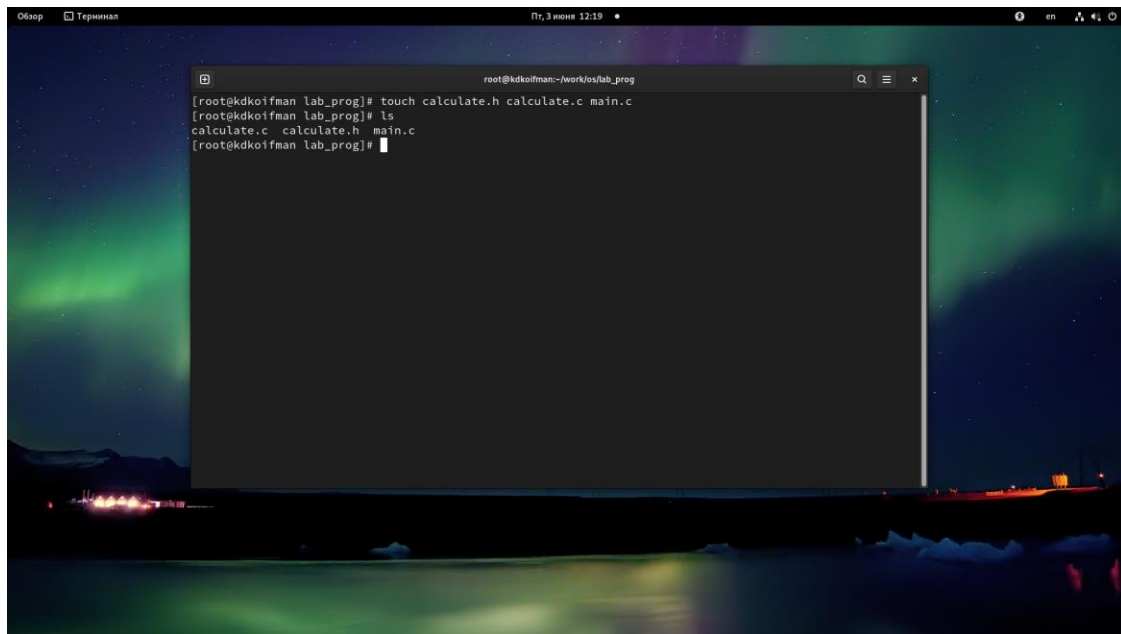
В домашнем каталоге создадим подкаталог `~/work/os/lab_prog`(рис.1)



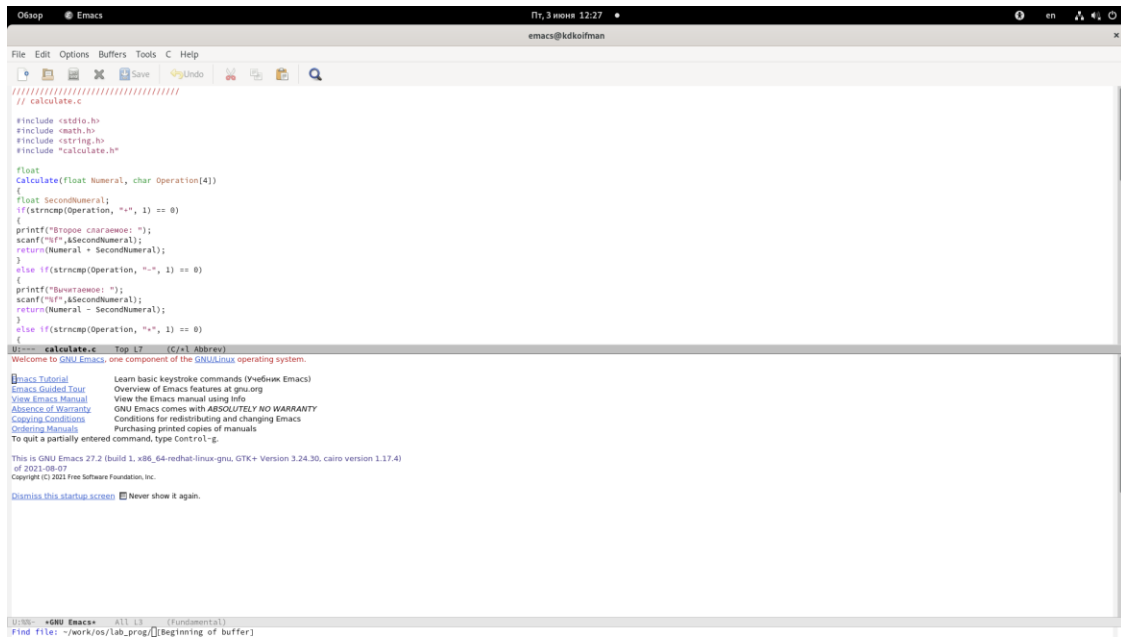
:-: рис.1

### 2 задание.

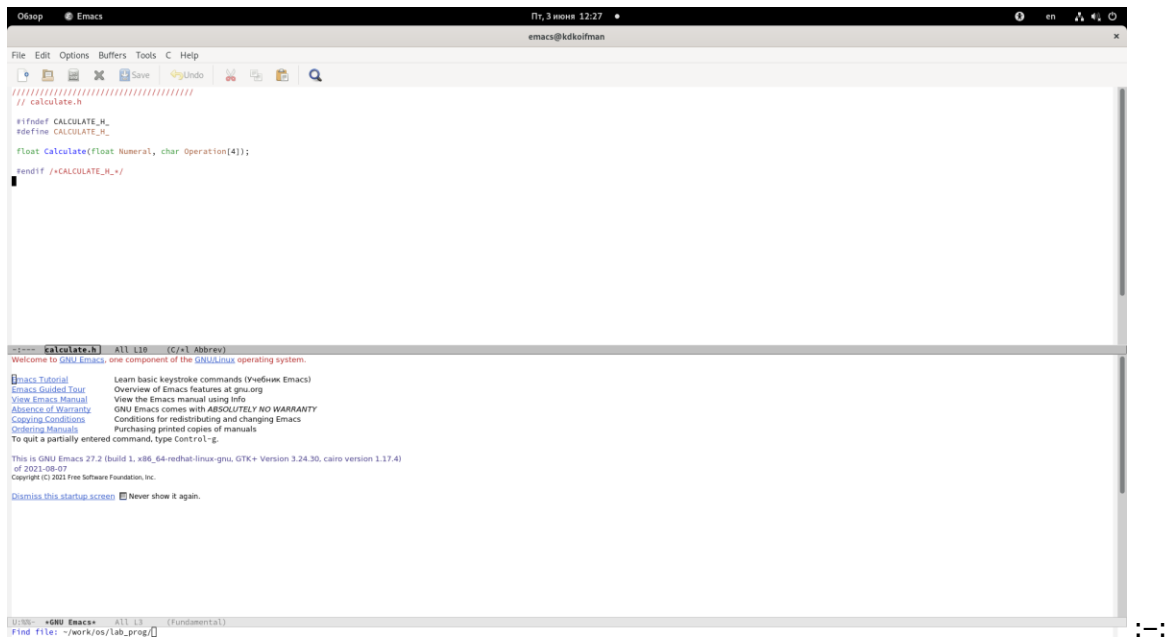
Создадим в нём файлы: `calculate.h`, `calculate.c`, `main.c`(рис.2, 3, 4, 5)



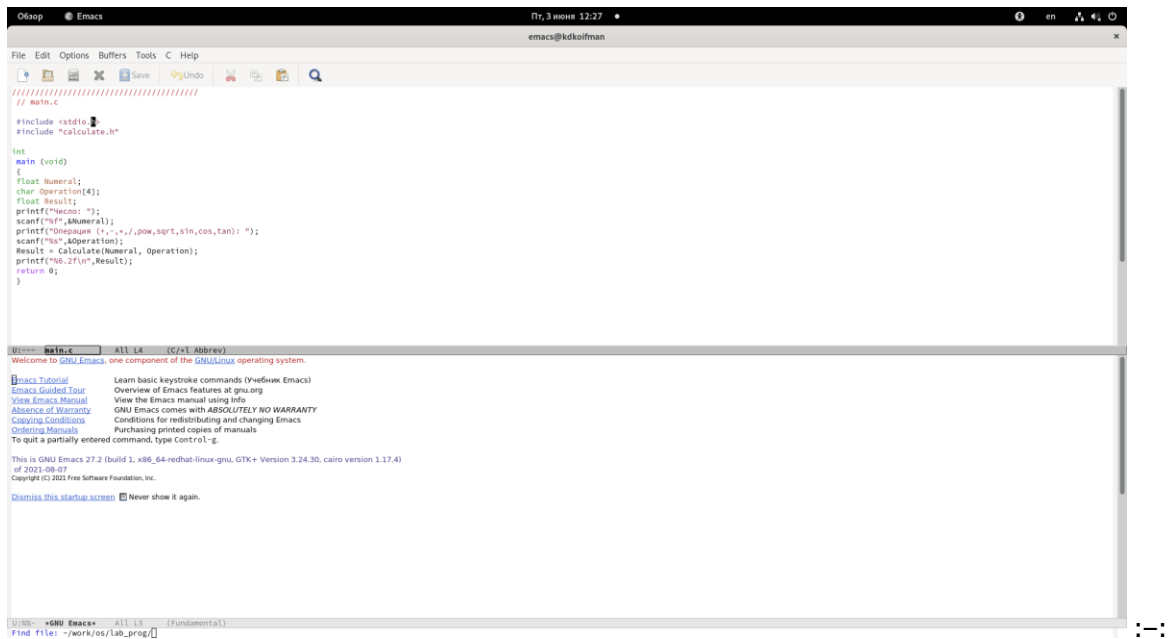
:-: puc.2



puc.3(calculate.c)



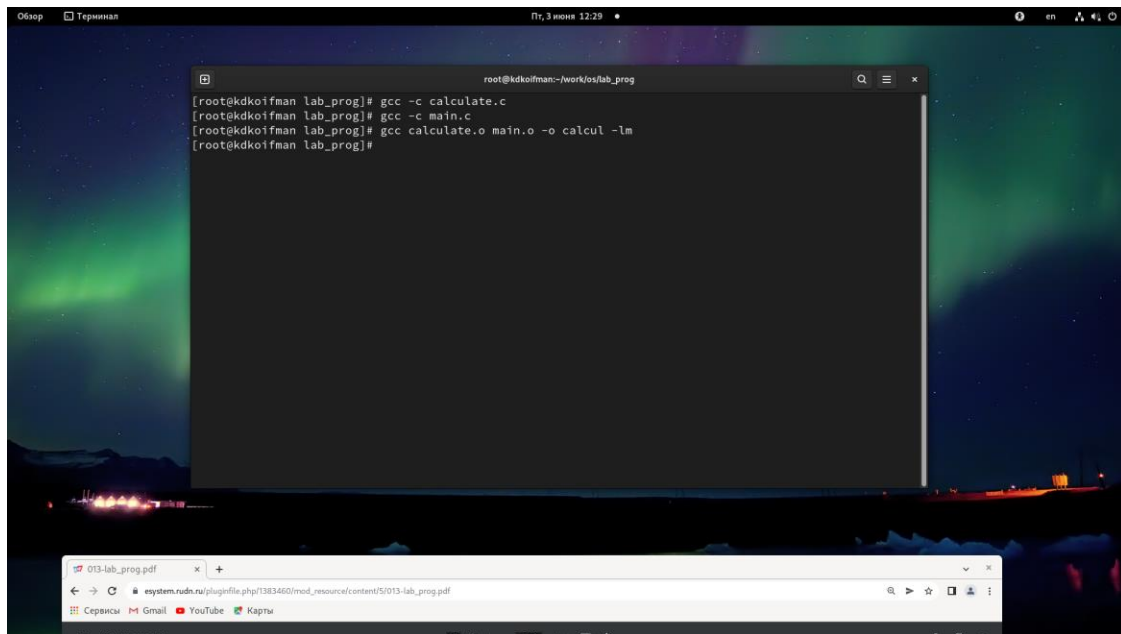
*pic.4(calculate.h)*



*pic.5(main.c)*

## 3 задание.

Выполним компиляцию программы посредством gcc(рис.6)

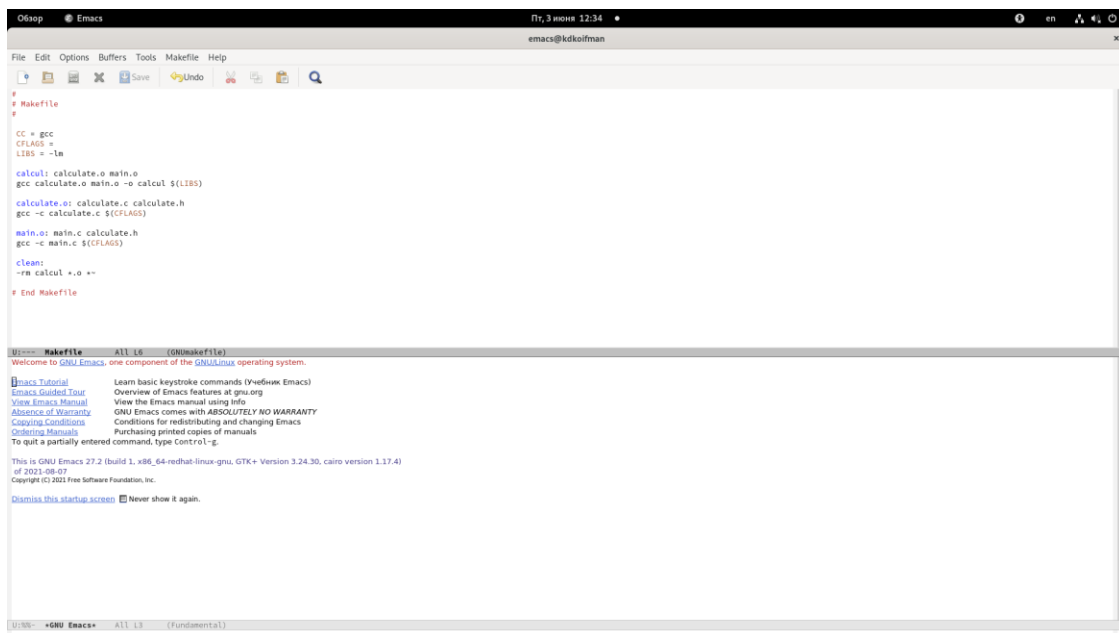


:-: рис.6

## 4 задание. Исправили несколько незначительных синтаксических ошибок.

## 5 задание.

Создадим Makefile(рис.7)

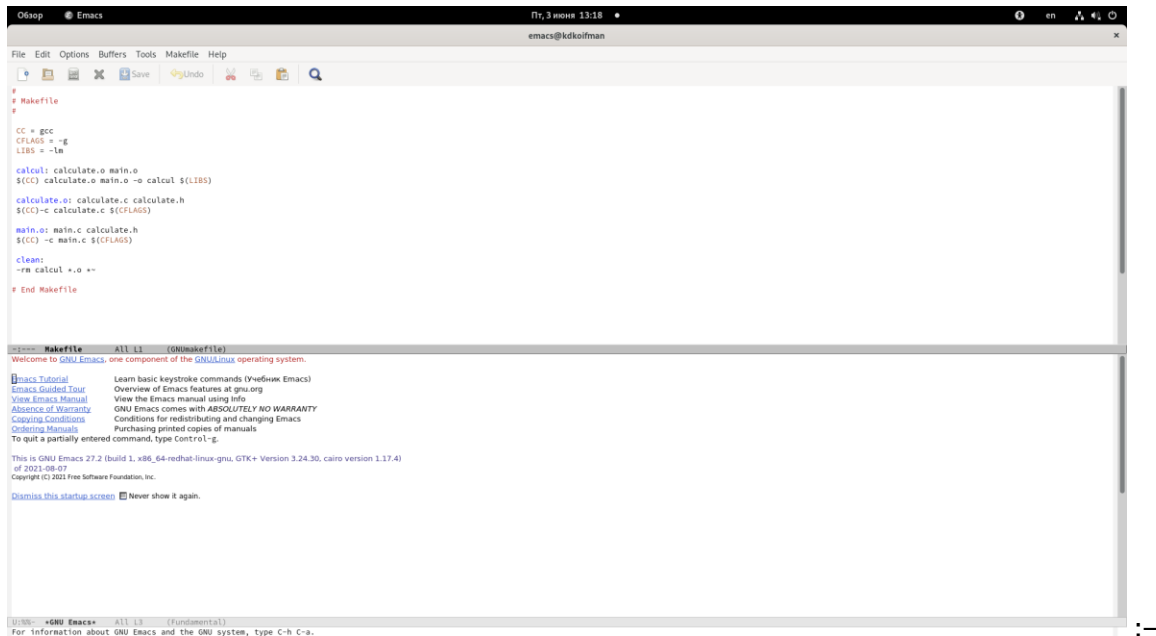


:-: рис.7

Данный Makefile состоит из параметров, определяющих принцип работы с указанными файлами - используемые библиотеки, цели файлов и команды для их выполнения.

## 6 задание.

С помощью gdb выполним отладку программы calcul (перед использованием gdb исправим Makefile)(рис.8, 9, 10)



```
# Makefile
#
CC = gcc
CFLAGS = -g
LIBS = -lm

calcul: calculate.o main.o
$(CC) calculate.o main.o -o calcul $(LIBS)

calculate.o: calculate.c calculate.h
$(CC) -c calculate.c $(CFLAGS)

main.o: main.c calculate.h
$(CC) -c main.c $(CFLAGS)

clean:
-rm calcul *.o *~

# End Makefile
```

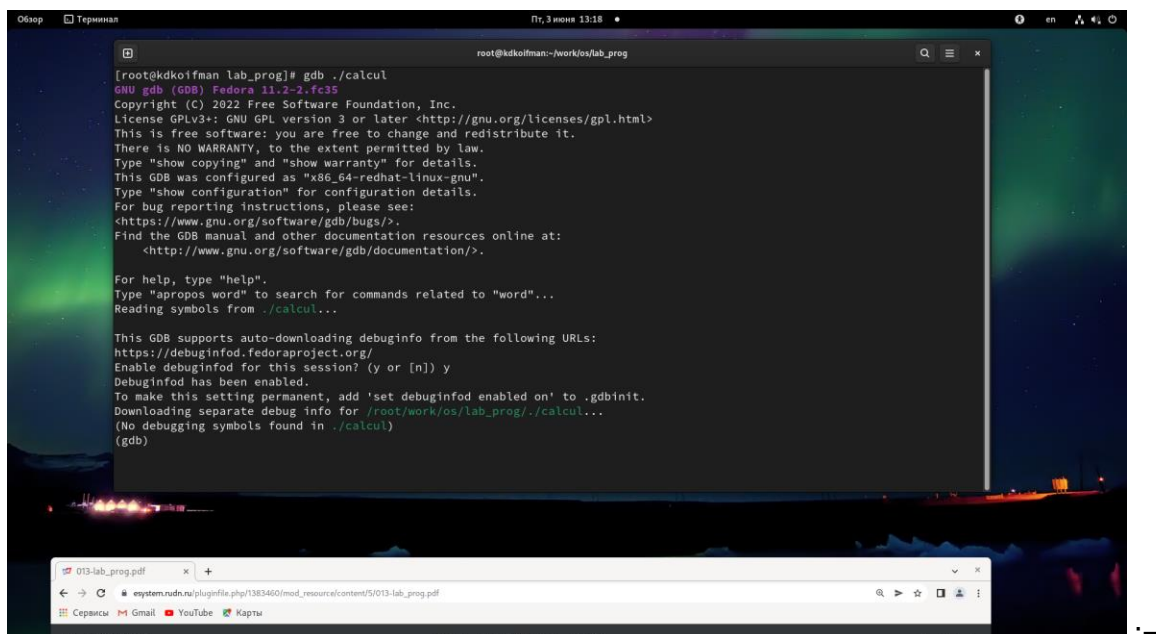
Welcome to GNU Emacs, one component of the GNU/Linux operating system.

[Emacs Tutorial](#) Learn basic keystroke commands (Vueбник Emacs)  
[Emacs Guided Tour](#) Overview of Emacs features at gnu.org  
[View Emacs Manual](#) View the Emacs manual using Info  
[Abstracts of Warnings](#) GNU Emacs comes with ABSOLUTELY NO WARRANTY  
[Copying Conditions](#) Conditions for redistributing and changing Emacs  
[Ordering Manuals](#) Purchasing printed copies of manuals  
To quit a partially entered command, type Control-g.

This is GNU Emacs 27.2 (build 1, x86\_64-redhat-linux-gnu, GTK+ Version 3.24.30, cairo version 1.17.4) of 2021-08-07  
Copyright (C) 2021 Free Software Foundation, Inc.  
[Dismiss this startup screen](#) ☐ Never show it again.

[[USE: <GNU Emacs> All 13 (fundamental)]  
For information about GNU Emacs and the GNU system, type C-h C-a.

рис.8(изменённый Makefile)



```
[root@kdkoifman lab_prog]# gdb ./calcul
GNU gdb (GDB) Fedora 11.2-2.fc35
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./calcul...

This GDB supports auto-downloading debuginfo from the following URLs:
https://debuginfod.fedoraproject.org/
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for /root/work/os/lab_prog/./calcul...
(no debugging symbols found in ./calcul)
(gdb)
```

рис.9(загрузим в отладчике gdb исполняемый файл calcul)

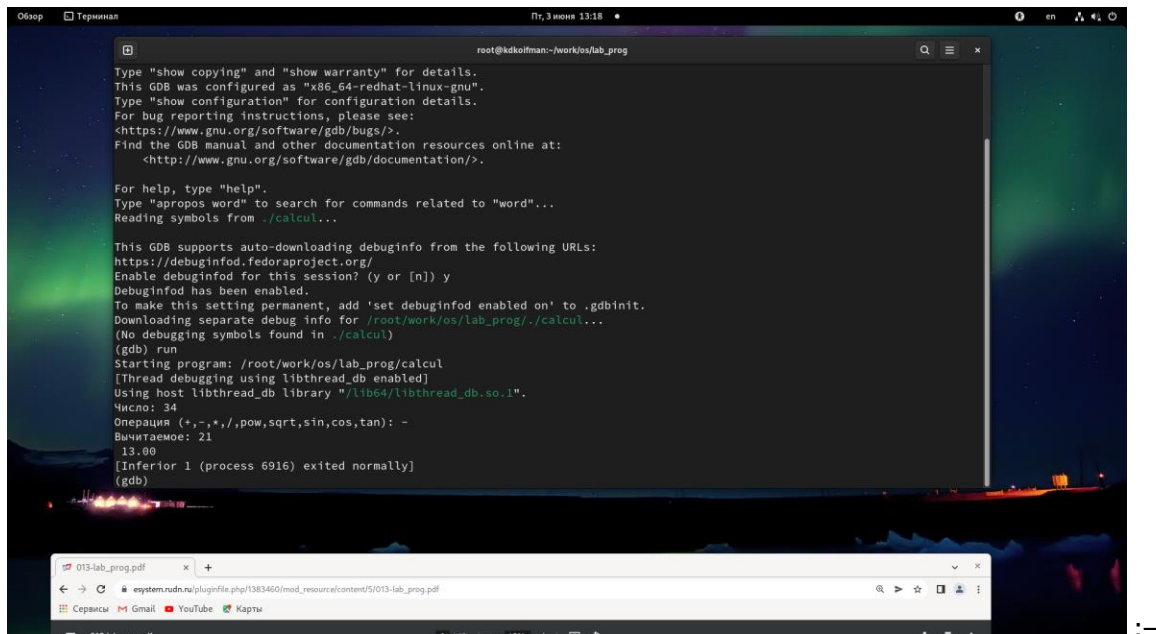


рис.10(протестируем работы программы)

## 7 задание.

С помощью утилиты splint попробуем проанализировать коды файлов calculate.c и main.c.(рис.11, 12)

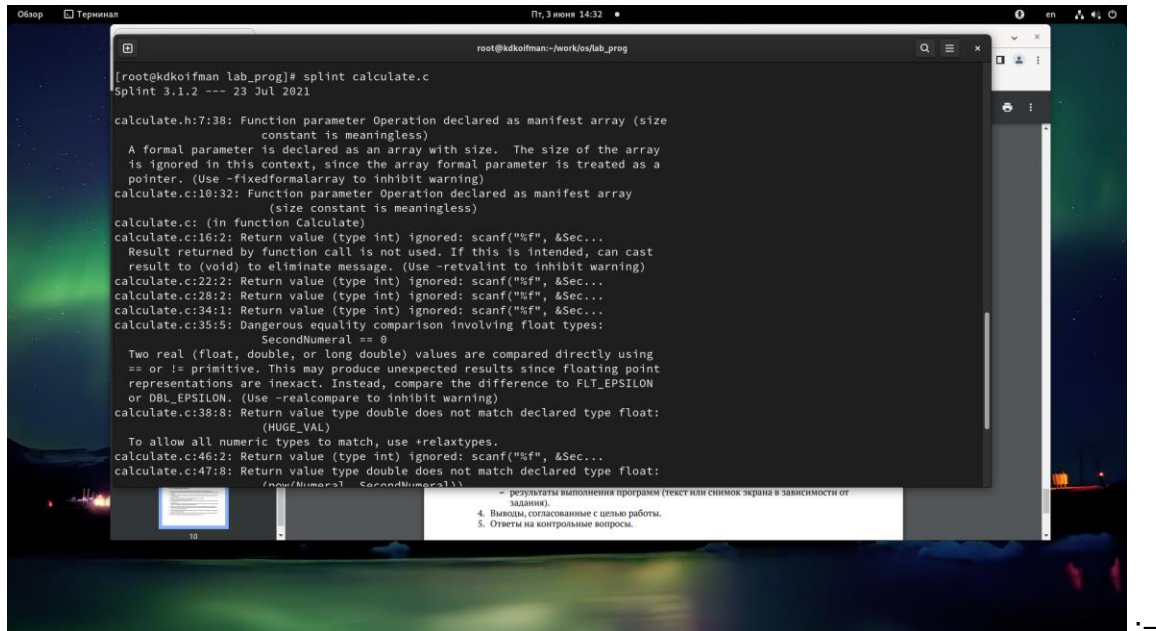


рис.11(файл calculate.c)



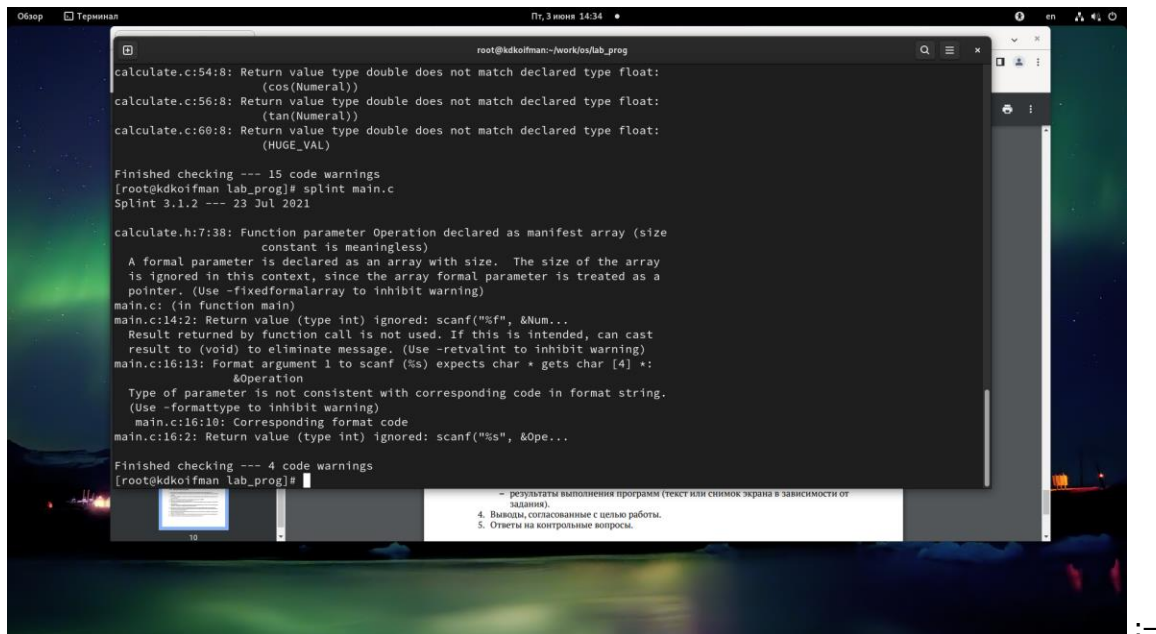


рис.12(main.c)

Просмотрев выведенную справку об этих файлах, можно сказать следующее:

- Функция **calculate.c:10:32** принимает в качестве параметра массив **char[4]**, однако используется как указатель **\*\*\*char\*\*\***, вследствие чего параметр, отвечающий за размер, утрачивает необходимость в использовании.
- При выполнении программы производится сравнение чисел типа **float** с помощью оператора сравнения **==**, что может спровоцировать неточные результаты или вовсе ошибки ввиду погрешностей.

## Вывод.

В ходе проделанной лабораторной работы мной были усвоены основные навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux.

## Контрольные вопросы.

1. Как получить информацию о возможностях программ gcc, make, gdb и др.? Для получения информации о возможностях программ gcc, make, gdb и др. можно использовать команду **man** или флаг **-help**.
2. Назовите и дайте краткую характеристику основным этапам разработки приложений в UNIX.

– планирование, включающее сбор и анализ требований к функционалу и другим характеристикам разрабатываемого приложения;

– проектирование, включающее в себя разработку базовых алгоритмов и спецификаций, определение языка программирования;

– разработка приложения:

- написание кода в текстовом редакторе;
- анализ разработанного кода;
- сборка, компиляция и разработка исполняемого модуля;
- тестирование и отладка, сохранение произведённых изменений;
- документирование.

3. Что такое суффикс в контексте языка программирования? Приведите примеры использования. В языке программирования C++ пользователям предоставляется возможность добавлять различные суффиксы к числовым литеральным константам для обозначения типа данных той или иной переменной (например, запись 21.340f означает, что переменная имеет тип float).
4. Каково основное назначение компилятора языка C в UNIX? Компилятор предназначен для компиляции программ (конвертация исходного кода в объектный файл)
5. Для чего предназначена утилита make? Утилита make позволяет автоматизировать процесс конвертации файлов из одной формы в другую и наоборот.
6. Приведите пример структуры Makefile. Дайте характеристику основным элементам этого файла.
7. Назовите основное свойство, присущее всем программам отладки. Что необходимо сделать, чтобы его можно было использовать? Основным свойством, присущим всем программам отладки, можно назвать остановку выполнения работы программы в определённых точках исходного кода (для этого нужно устанавливать breakpoint на желаемой точке).
8. Назовите и дайте основную характеристику основным командам отладчика gdb.

backtrace - вывод на экран пути к текущей точке останова (по сути вывод названий всех функций)

break - установить точку останова (в качестве параметра может быть указан номер строки или название функции)

clear - удалить все точки останова в функции

continue - продолжить выполнение программы

delete - удалить точку останова

display - добавить выражение в список выражений, значения которых отображаются при достижении точки останова программы

finish - выполнить программу до момента выхода из функции

info breakpoints - вывести на экран список используемых точек останова

info watchpoints - вывести на экран список используемых контрольных выражений

list - вывести на экран исходный код (в качестве параметра может быть указано название файла и через двоеточие номера начальной и конечной строк)

next - выполнить программу пошагово, но без выполнения вызываемых в программе функций

print - вывести значение указываемого в качестве параметра выражения

run - запуск программы на выполнение

set - установить новое значение переменной

step - пошаговое выполнение программы

watch - установить контрольное выражение, при изменении значения которого программа будет остановлена

9. Опишите по шагам схему отладки программы, которую Вы использовали при выполнении лабораторной работы.

1)Установка брейкпоинтов 2)Выполнение программы

10. Прокомментируйте реакцию компилятора на синтаксические ошибки в программе при его первом запуске. Компиляция завершается с указанием на синтаксические ошибки.
11. Назовите основные средства, повышающие понимание исходного кода программы. Грамотные комментарии, хорошо воспринимаемая структура программы.
12. Каковы основные задачи, решаемые программой splint? Основными задачами программы splint являются вывод ошибок и предупреждений, которые указывают на возможные проблемы в исходном коде.