

# Отчёт по лабораторной работе №13

Дугина Виктория Игоревна

**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.

## Выполнение работы

1. В домашнем каталоге создайте подкаталог ~/work/os/lab\_prog.

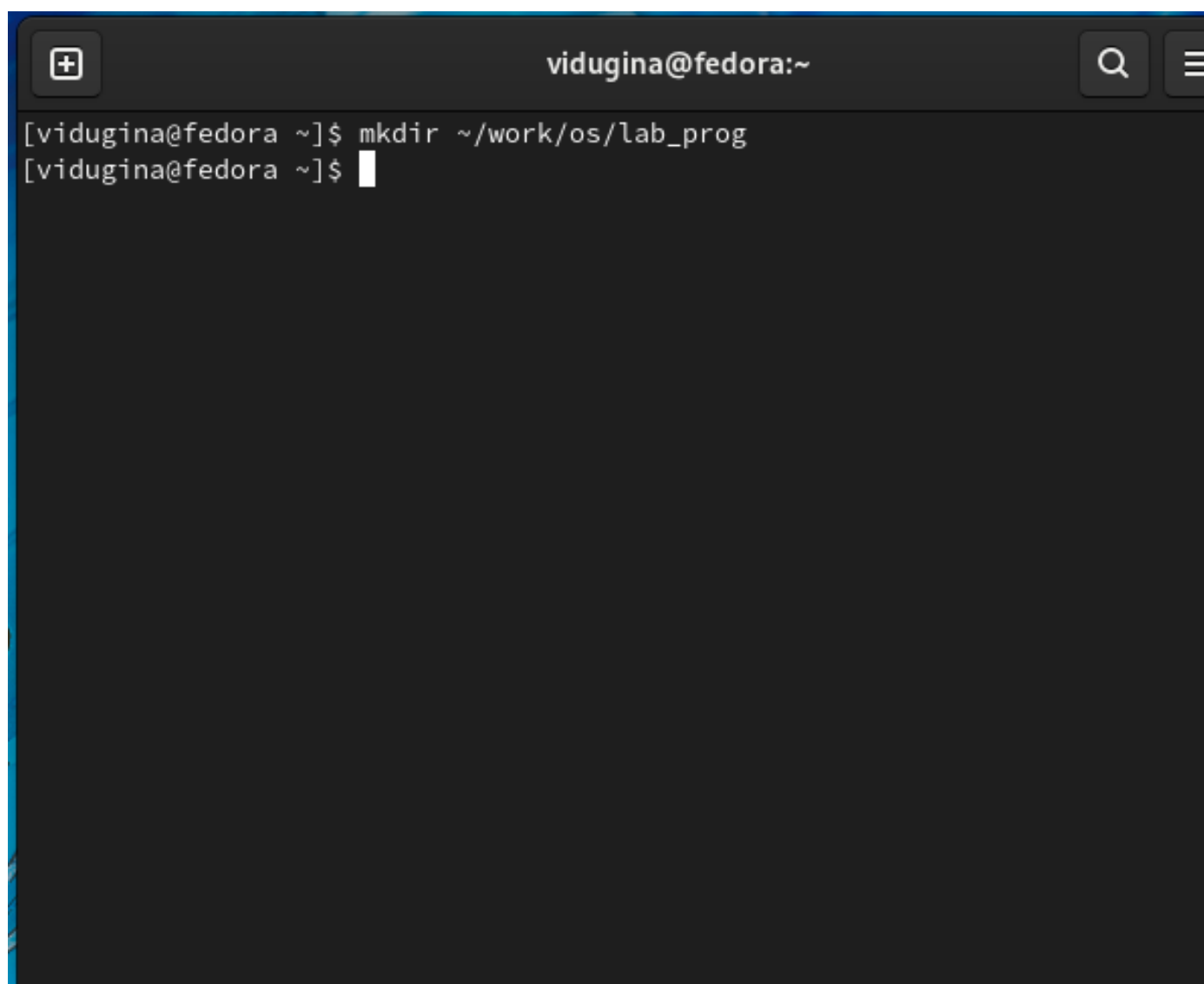
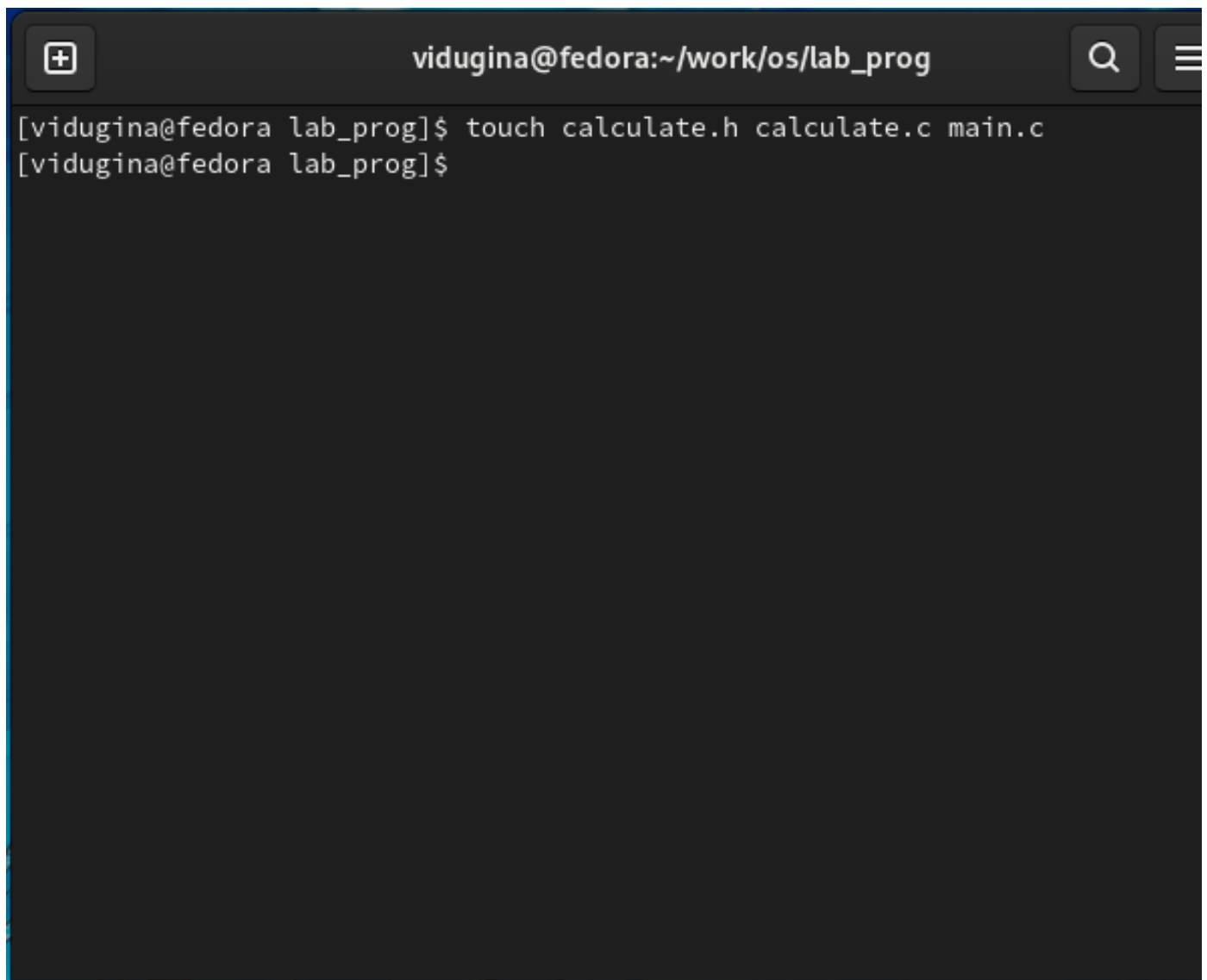


Рис 1.1

2. Создайте 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.

Используя команду touch, создаём файлы, после записывая в них нужный код.

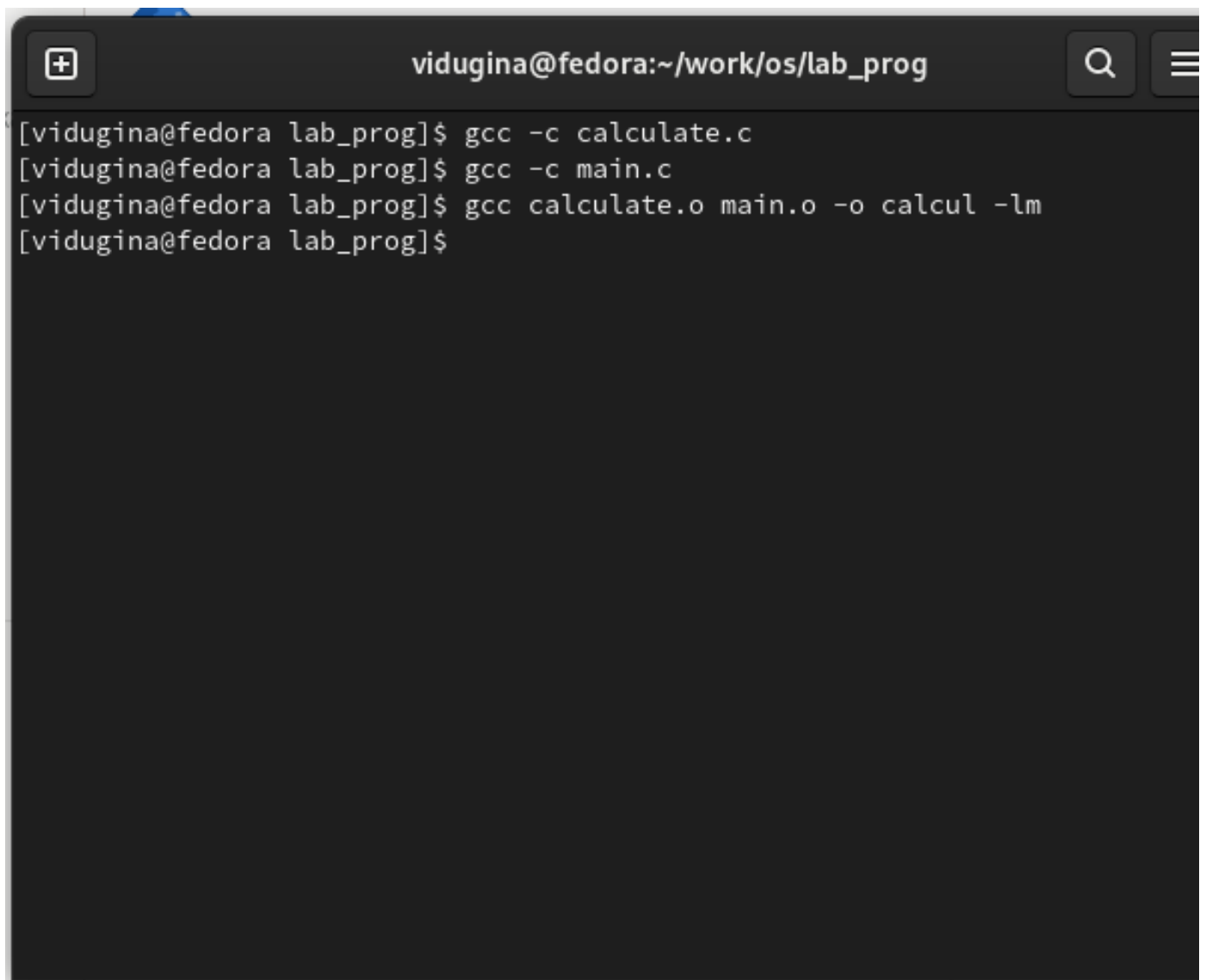
A terminal window with a dark background. The title bar at the top shows a plus icon on the left, the text "vidugina@fedora:~/work/os/lab\_prog" in the center, and a magnifying glass icon and a hamburger menu icon on the right. The terminal content shows two lines of text: the first line is "[vidugina@fedora lab\_prog]\$ touch calculate.h calculate.c main.c" and the second line is "[vidugina@fedora lab\_prog]\$".

```
vidugina@fedora:~/work/os/lab_prog  
[vidugina@fedora lab_prog]$ touch calculate.h calculate.c main.c  
[vidugina@fedora lab_prog]$
```

Рис 2.1

### 3. Выполните компиляцию программы посредством gcc.

Компилируем программу.

A terminal window with a dark background. The title bar at the top shows a plus icon on the left, the text "vidugina@fedora:~/work/os/lab\_prog" in the center, and a magnifying glass icon and a hamburger menu icon on the right. The terminal content shows four lines of commands and their prompts: [vidugina@fedora lab\_prog]\$ gcc -c calculate.c, [vidugina@fedora lab\_prog]\$ gcc -c main.c, [vidugina@fedora lab\_prog]\$ gcc calculate.o main.o -o calcul -lm, and [vidugina@fedora lab\_prog]\$.

```
vidugina@fedora:~/work/os/lab_prog

[vidugina@fedora lab_prog]$ gcc -c calculate.c
[vidugina@fedora lab_prog]$ gcc -c main.c
[vidugina@fedora lab_prog]$ gcc calculate.o main.o -o calcul -lm
[vidugina@fedora lab_prog]$
```

*Puc 3.1*

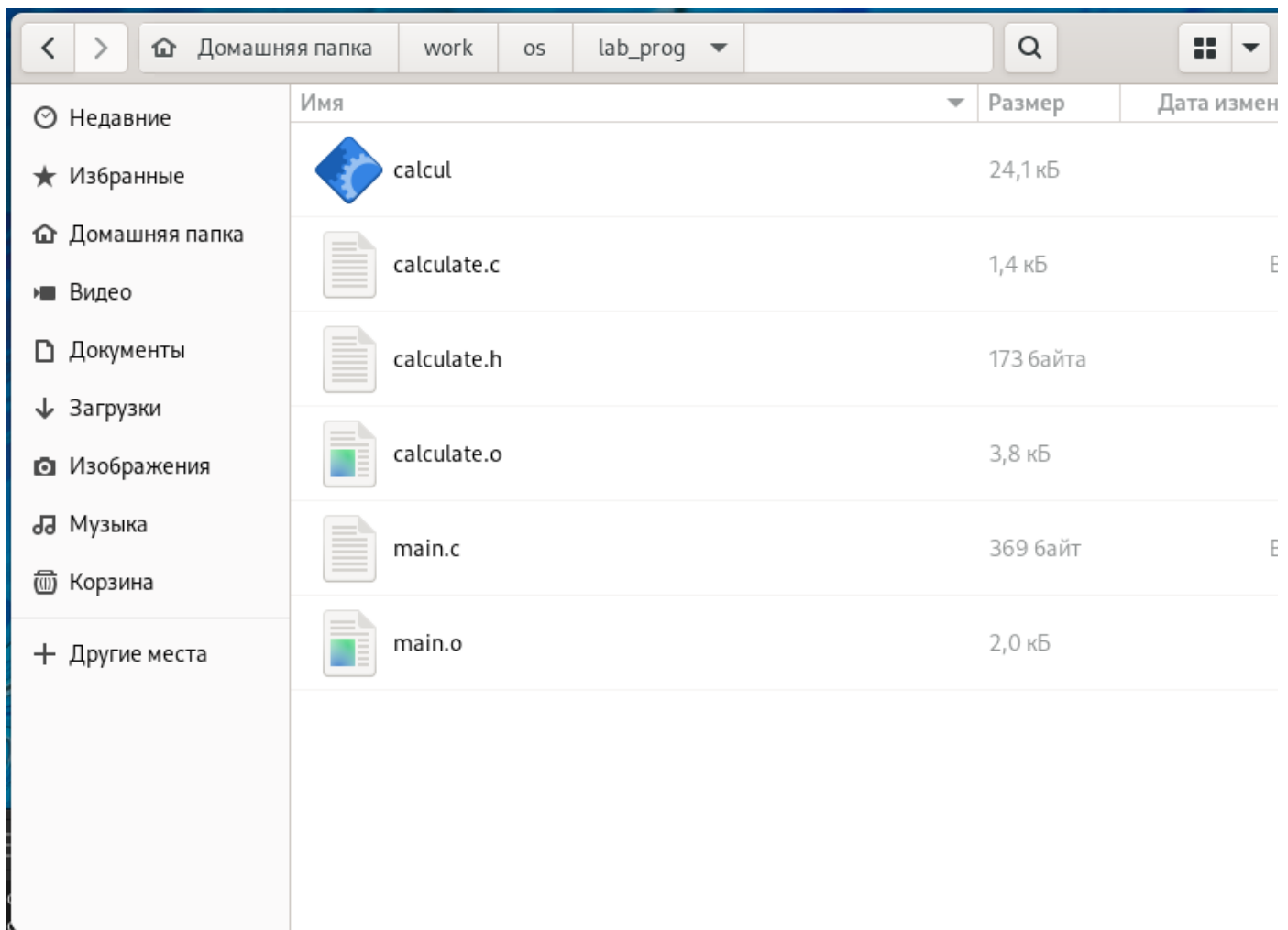


Рис 3.2

**4. При необходимости исправьте синтаксические ошибки.**

Синтаксических ошибок нет.

**5. Создайте Makefile со следующим содержанием...Поясните в отчёте его содержание.**

```
1 #
2 # Makefile
3 #
4
5 CC = gcc
6 CFLAGS =
7 LIBS = -lm
8
9 calcul: calculate.o main.o
10     gcc calculate.o main.o -o calcul $(LIBS)
11
12 calculate.o: calculate.c calculate.h
13     gcc -c calculate.c $(CFLAGS)
14
15 main.o: main.c calculate.h
16     gcc -c main.c $(CFLAGS)
17
18 clean:
19     rm calcul *.o *~
20
21 # End Makefile
```

Рис 5.1

В данном Makefile существует 3 переменные и 4 цели:

Переменные:

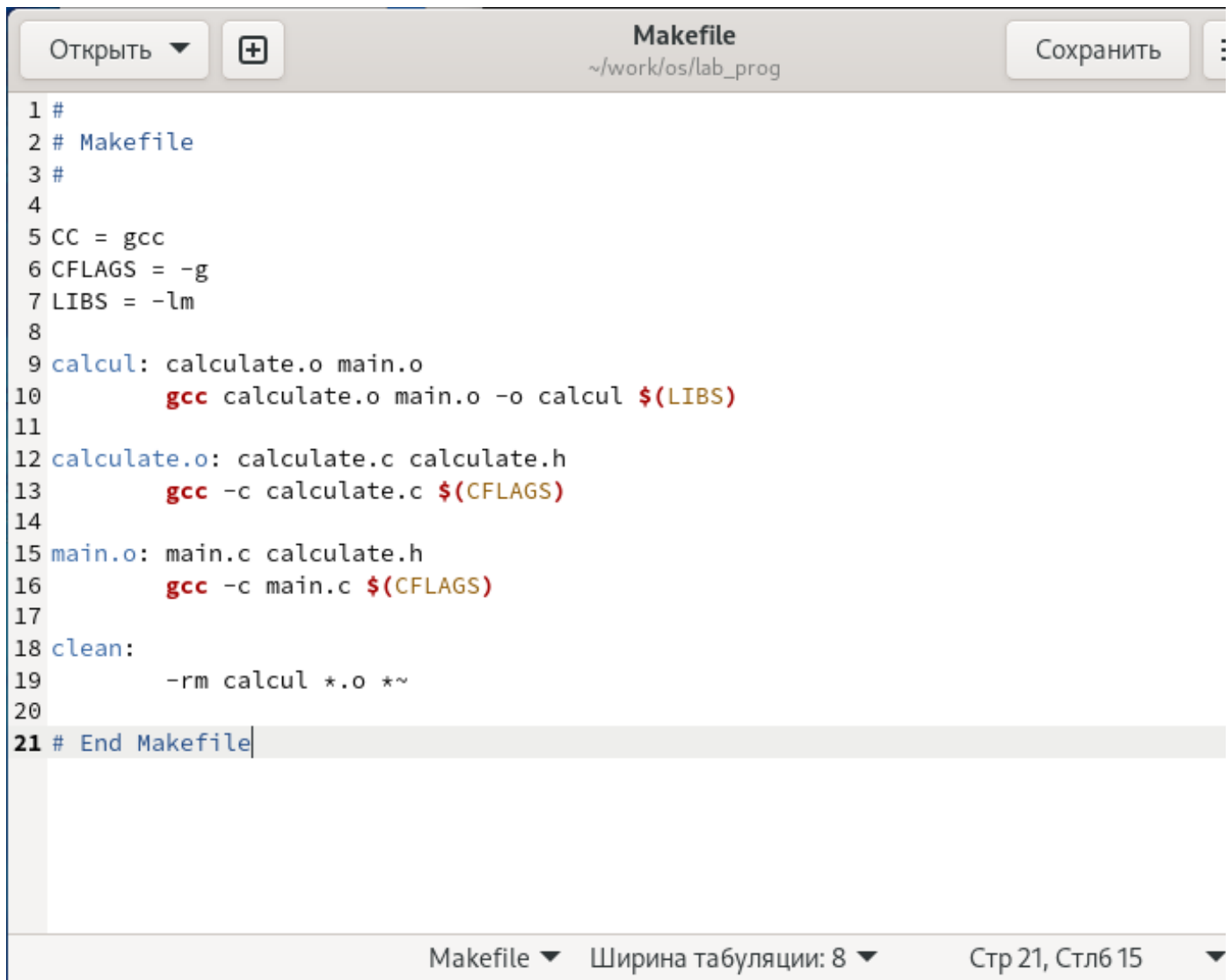
- CC = gcc: далее не используется
- CFLAGS = : не содержит ничего
- LIBS = -lm: флаг gcc, находит и вызывает библиотеку libm, содержащую <math.h>

Цели:

- calcul: Зависит от целей calculate.o и main.o. То есть будет выполнена, только при успешном выполнении этих команд (а значит необходимые файлы будут созданы). Последнее действие для компиляции программы.
- calculate.o: Зависит от файлов calculate.c и calculate.h. Компилирует файл calculate.c.
- main.o: Зависит от файлов main.c и calculate.h. Компилирует файл main.c.
- clean: Удаляет файлы calcul, те которые заканчиваются ".o".

**6. С помощью gdb выполните отладку программы calcul (перед использованием gdb исправьте Makefile):**

Добавляем в переменную CFLAGS -g. Данная переменная стоит в строках 13 и 16 и тем самым при компиляции добавится возможность работать в gdb.



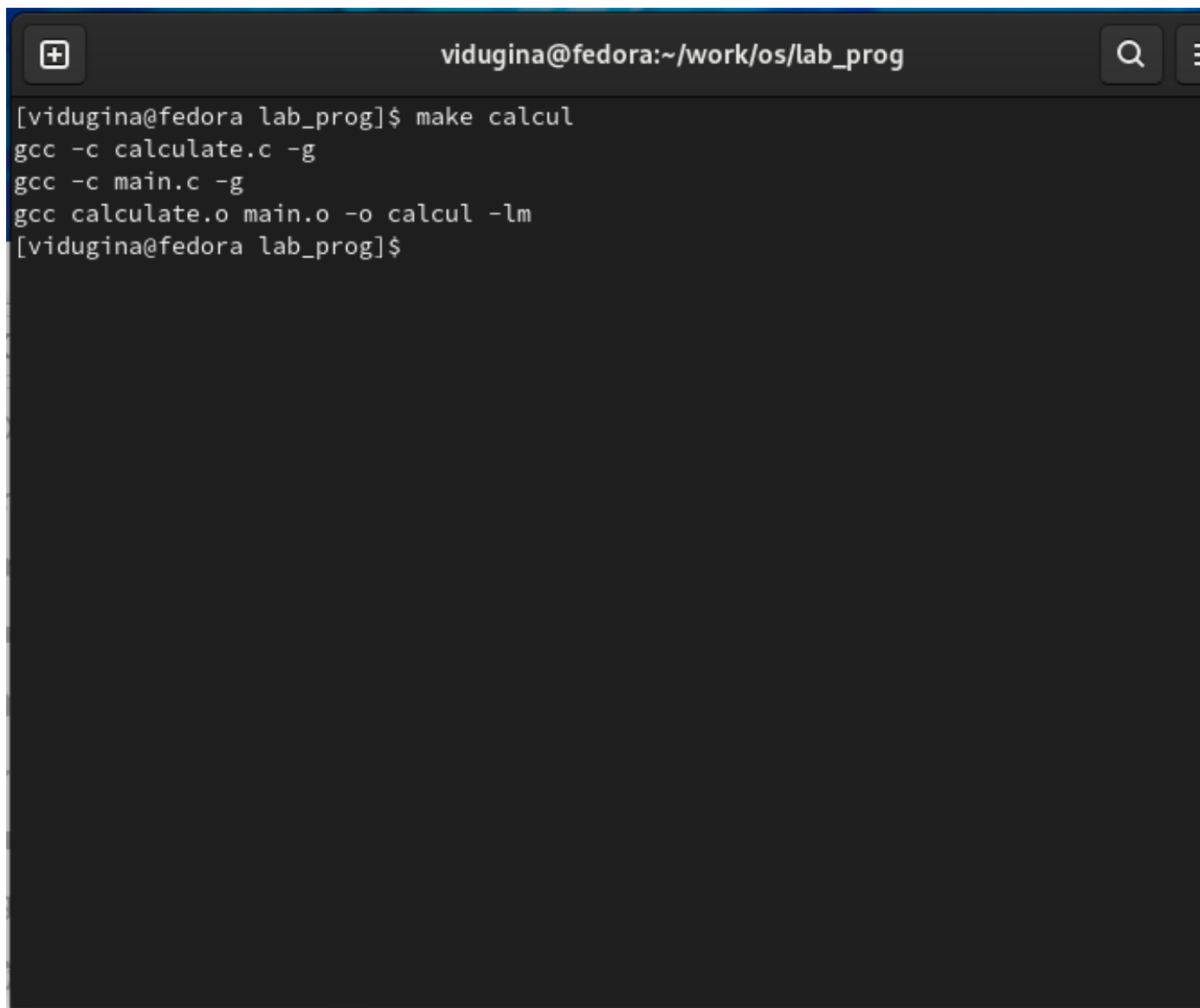
The screenshot shows a text editor window titled "Makefile" with the path "~/work/os/lab\_prog". The editor contains a Makefile with the following content:

```
1 #
2 # Makefile
3 #
4
5 CC = gcc
6 CFLAGS = -g
7 LIBS = -lm
8
9 calcul: calculate.o main.o
10     gcc calculate.o main.o -o calcul $(LIBS)
11
12 calculate.o: calculate.c calculate.h
13     gcc -c calculate.c $(CFLAGS)
14
15 main.o: main.c calculate.h
16     gcc -c main.c $(CFLAGS)
17
18 clean:
19     -rm calcul *.o *~
20
21 # End Makefile
```

The status bar at the bottom indicates "Makefile", "Ширина табуляции: 8", and "Стр 21, Стлб 15".

Рис 6.1 - Изменённый Makefile

Компилируем программу, используя Makefile.

A terminal window with a dark background. The title bar shows the user 'vidugina@fedora' and the directory '~/work/os/lab\_prog'. The terminal contains the following text:

```
[vidugina@fedora lab_prog]$ make calcul
gcc -c calculate.c -g
gcc -c main.c -g
gcc calculate.o main.o -o calcul -lm
[vidugina@fedora lab_prog]$
```

Рис 6.2

Запускаем отладчик GDB, загрузив в него calcul. Просматриваем строки 20-27 в файле calculate.c (list calculate.c:20,27) и добавлем точку останова на строке 21.

```
vidugina@fedora:~/work/os/lab_prog — gdb ./calcul
[vidugina@fedora lab_prog]$ gdb ./calcul
GNU gdb (GDB) Fedora 10.2-9.fc35
Copyright (C) 2021 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...
(gdb) list calculate.c:20,27
20     printf("Вычитаемое: ");
21     scanf("%f",&SecondNumeral);
22     return(Numeral - SecondNumeral);
23 }
24 else if(strncmp(Operation, "*", 1) == 0)
25 {
26     printf("Множитель: ");
27     scanf("%f",&SecondNumeral);
(gdb) break 21
Breakpoint 1 at 0x40121e: file calculate.c, line 21.
(gdb)
```

Рис 6.3

Выводим информацию о точках останова.

```
(gdb) info breakpoints
Num    Type           Disp Enb Address              What
1      breakpoint    keep y   0x000000000040121e in Calculate at calculate.c:21
(gdb)
```

Рис 6.4

Запускаем программу и убеждаемся, что она останавливается в нужном месте.

```
(gdb) run
Starting program: /home/vidugina/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -

Breakpoint 1, Calculate (Numeral=5, Operation=0x7fffffffdf64 "-") at calculate.c:21
21     scanf("%f",&SecondNumeral);
(gdb)
```

Рис 6.5

Смотрим значение переменной, используя сначала print, а потом display. Значения они



выводят одинаковое, но оформляя по разному.

```
(gdb) print Numeral
$1 = 5
(gdb) display Numeral
1: Numeral = 5
(gdb)
```

Рис 6.6

Удаляем точку останова.

```
(gdb) info breakpoints
Num    Type           Disp Enb Address          What
1      breakpoint     keep y   0x0000000000004012 in Calculate at calculate.c:21
       breakpoint already hit 1 time
(gdb) delete 1
(gdb)
```

Рис 6.7

**7. С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c.**

Вводим splint calculate.c.

```
[vidugina@fedora lab_prog]$ splint calculate.c
Splint 3.1.2 --- 23 Jul 2021

calculate.h:6:37: Function parameter Operation declared as manifest array (size
        constant is meaningless)
    A formal parameter is declared as an array with size.  The size of the array
    is ignored in this context, since the array formal parameter is treated as a
    pointer. (Use -fixedformalarray to inhibit warning)
calculate.c:9:38: Function parameter Operation declared as manifest array (size
        constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:15:1: Return value (type int) ignored: scanf("%f", &Sec...
    Result returned by function call is not used. If this is intended, can cast
    result to (void) to eliminate message. (Use -retvalint to inhibit warning)
calculate.c:21:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:27:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:33:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:34:4: Dangerous equality comparison involving float types:
        SecondNumeral == 0
    Two real (float, double, or long double) values are compared directly using
    == or != primitive. This may produce unexpected results since floating point
    representations are inexact. Instead, compare the difference to FLT_EPSILON
    or DBL_EPSILON. (Use -realcompare to inhibit warning)
calculate.c:37:7: Return value type double does not match declared type float:
        (HUGE_VAL)
    To allow all numeric types to match, use +relaxtypes.
calculate.c:45:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:46:7: Return value type double does not match declared type float:
        (pow(Numeral, SecondNumeral))
calculate.c:49:7: Return value type double does not match declared type float:
        (sqrt(Numeral))
calculate.c:51:7: Return value type double does not match declared type float:
        (sin(Numeral))
calculate.c:53:7: Return value type double does not match declared type float:
        (cos(Numeral))
calculate.c:55:7: Return value type double does not match declared type float:
        (tan(Numeral))
calculate.c:59:7: Return value type double does not match declared type float:
        (HUGE_VAL)

Finished checking --- 15 code warnings
[vidugina@fedora lab_prog]$
```

Рис 7.1

Вводим splint main.c.

```
vidugina@fedora:~/work/os/lab_prog

[vidugina@fedora lab_prog]$ splint main.c
Splint 3.1.2 --- 23 Jul 2021

calculate.h:6:37: Function parameter Operation declared as manifest array (s
                constant is meaningless)
  A formal parameter is declared as an array with size. The size of the arr
  is ignored in this context, since the array formal parameter is treated as
  pointer. (Use -fixedformalarray to inhibit warning)
main.c: (in function main)
main.c:11:1: Return value (type int) ignored: scanf("%f", &Num...
  Result returned by function call is not used. If this is intended, can cas
  result to (void) to eliminate message. (Use -retvalint to inhibit warning)
main.c:13:12: Format argument 1 to scanf (%s) expects char * gets char [4] +
                &Operation
  Type of parameter is not consistent with corresponding code in format str
  (Use -formattype to inhibit warning)
  main.c:13:9: Corresponding format code
main.c:13:1: Return value (type int) ignored: scanf("%s", &Ope...

Finished checking --- 4 code warnings
[vidugina@fedora lab_prog]$
```

Рис 7.2

**Вывод:** я приобрела простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования С калькулятора с простейшими функциями.

## Контрольные вопросы

1. Как получить информацию о возможностях программ gcc, make, gdb и др.?
2. Назовите и дайте краткую характеристику основным этапам разработки приложений в UNIX.
3. Что такое суффикс в контексте языка программирования? Приведите примеры использования.
4. Каково основное назначение компилятора языка С в UNIX?
5. Для чего предназначена утилита make?
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?

- 8. Назовите и дайте основную характеристику основным командам отладчика gdb.**
- 9. Опишите по шагам схему отладки программы, которую Вы использовали при выполнении лабораторной работы.**
- 10. Прокомментируйте реакцию компилятора на синтаксические ошибки в программе при его первом запуске.**
- 11. Назовите основные средства, повышающие понимание исходного кода программы.**
- 12. Каковы основные задачи, решаемые программой splint?**