

# Лабораторная работа № 13

Операционные системы

---

Голованова Мария Константиновна

6 мая 2023

Российский университет дружбы народов, Москва, Россия

## Информация

---

- Голованова Мария Константиновна
  - НММбд-01-22, 1132226478
  - Факультет физико-математических и естественных наук
  - Российский университет дружбы народов

## Цель работы

---

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

## Задание

---

Создать на языке программирования C 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.

## Теоретическое введение

---



Стандартным средством для компиляции программ в ОС типа UNIX является GCC (GNU Compiler Collection). Это набор компиляторов для разного рода языков программирования (C, C++, Java, Фортран и др.). Работа с GCC производится при помощи одноимённой управляющей программы gcc, которая интерпретирует аргументы командной строки, определяет и осуществляет запуск нужного компилятора для входного файла. Файлы с расширением (суффиксом) .c воспринимаются gcc как программы на языке C, файлы с расширением .cc или .C — как файлы на языке C++, а файлы с расширением .o считаются объектными.

Для сборки разрабатываемого приложения и собственно компиляции полезно воспользоваться утилитой `make`. Она позволяет автоматизировать процесс преобразования файлов программы из одной формы в другую, отслеживает взаимосвязи между файлами. Для работы с утилитой `make` необходимо в корне рабочего каталога с Вашим проектом создать файл с названием `makefile` или `Makefile`, в котором будут описаны правила обработки файлов Вашего программного комплекса. Сначала задаётся список целей, разделённых пробелами, за которым идёт двоеточие и список зависимостей. Затем в следующих строках указываются команды. Строки с командами обязательно должны начинаться с табуляции. В качестве цели в `Makefile` может выступать имя файла или название какого-то действия. Зависимость задаёт исходные параметры (условия) для достижения указанной цели. Зависимость также может быть названием какого-то действия. Команды — собственно действия, которые необходимо выполнить для достижения цели.

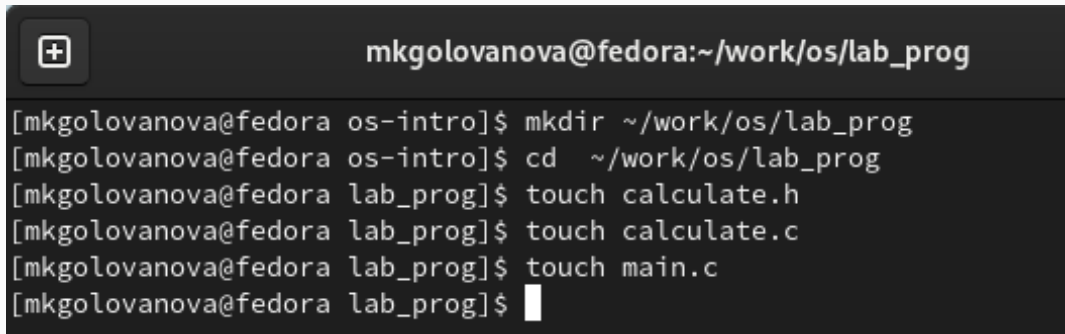
Во время работы над кодом программы программист неизбежно сталкивается с появлением ошибок в ней. Использование отладчика для поиска и устранения ошибок в программе существенно облегчает жизнь программиста. В комплект программ GNU для ОС типа UNIX входит отладчик GDB (GNU Debugger). Для использования GDB необходимо скомпилировать анализируемый код программы таким образом, чтобы отладочная информация содержалась в результирующем бинарном файле. Для этого следует воспользоваться опцией `-g` компилятора `gcc`: `* gcc -c file.c -g` После этого для начала работы с `gdb` необходимо в командной строке ввести одноимённую команду, указав в качестве аргумента анализируемый бинарный файл: `* gdb file.o` Затем можно использовать по мере необходимости различные команды `gdb`. Для выхода из `gdb` можно воспользоваться командой `quit` (или её сокращённым вариантом `q`) или комбинацией клавиш `Ctrl-d`. Более подробную информацию по работе с `gdb` можно получить с помощью команд `gdb -h` и `man gdb`.

## Выполнение лабораторной работы

---

1.

- Я создала в домашнем каталоге подкаталог `~/work/os/lab_prog`, а затем создала в нём файлы: `calculate.h`, `calculate.c`, `main.c`. (рис. 1).



A terminal window with a dark background. The title bar shows a plus icon in a square and the text `mkgolovanova@fedora:~/work/os/lab_prog`. The terminal content shows a series of commands and their outputs:

```
[mkgolovanova@fedora os-intro]$ mkdir ~/work/os/lab_prog
[mkgolovanova@fedora os-intro]$ cd ~/work/os/lab_prog
[mkgolovanova@fedora lab_prog]$ touch calculate.h
[mkgolovanova@fedora lab_prog]$ touch calculate.c
[mkgolovanova@fedora lab_prog]$ touch main.c
[mkgolovanova@fedora lab_prog]$
```

The last line shows a cursor (a white block) at the end of the command prompt.

{.column

- Я ввела в файл `calculate.c` текст для реализации функций калькулятора (рис. 2, рис. 3).

GNU nano 6.4 calculate.c

```
////////// calculate.c

#include <stdio.h>
#include <math.h>
#include <string.h>
#include "calculate.h"


float
Calculate(float Numeral, char Operation[4])
{
    float SecondNumeral;
    if(strncmp(Operation, "+", 1) == 0)
    {
        printf("Второе слагаемое: ");
        scanf("%f", &SecondNumeral);
        return(Numeral + SecondNumeral);
    }
    else if(strncmp(Operation, "-", 1) == 0)
    {
        printf("Вычитаемое: ");
        scanf("%f", &SecondNumeral);
        return(Numeral - SecondNumeral);
    }
    else if(strncmp(Operation, "*", 1) == 0)
    {
        printf("Множитель: ");
        scanf("%f", &SecondNumeral);
        return(Numeral * SecondNumeral);
    }
    else if(strncmp(Operation, "/", 1) == 0)
    {
        printf("Делитель: ");
        scanf("%f", &SecondNumeral);
        if(SecondNumeral == 0)
        {
            printf("Ошибка: деление на ноль! ");
            return(HUGE_VAL);
        }
        else
            return(Numeral / SecondNumeral);
    }
}
```

{ columnwidth=50%}

```
{
printf("Ошибка: деление на ноль! ");
return(HUGE_VAL);
}
else
return(Numeral / SecondNumeral);
}
else if(strncmp(Operation, "pow", 3) == 0)
{
printf("Степень: ");
scanf("%f",&SecondNumeral);
return(pow(Numeral, SecondNumeral));
}
else if(strncmp(Operation, "sqrt", 4) == 0)
return(sqrt(Numeral));
else if(strncmp(Operation, "sin", 3) == 0)
return(sin(Numeral));
else if(strncmp(Operation, "cos", 3) == 0)
return(cos(Numeral));
else if(strncmp(Operation, "tan", 3) == 0)
return(tan(Numeral));
else
{
printf("Неправильно введено действие ");
return(HUGE_VAL);
}
}
```



- Я ввела текст основного файла main.c, реализующий интерфейс пользователя к калькулятору (рис. 5).



```
GNU nano 6.4                               main.c
////////////////////////////////////
// main.c

#include <stdio.h>
#include "calculate.h"

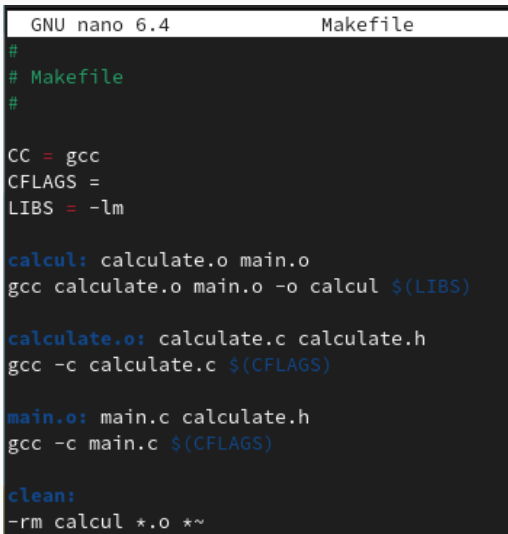
int
main (void)
{
    float Numeral;
    char Operation[4];
    float Result;
    printf("Число: ");
    scanf("%f",&Numeral);
    printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
    scanf("%s",&Operation);
    Result = Calculate(Numeral, Operation);
```

- Я выполнила компиляцию программы посредством gcc (рис. 6).

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

{.column

- Я создала Makefile со следующим содержанием (рис. 7).



```
GNU nano 6.4      Makefile
#
# 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 *~
```

- Я исправила Makefile (рис. 8).

```
GNU nano 6.4                                Makefile
#
# Makefile
#

CC = gcc
CFLAGS =
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 *.ts
```

С помощью gdb я выполнила отладку программы calcul: - Запустила отладчик GDB, загрузив в него программу для отладки (рис.9).

```
[mkgolovanova@fedora lab_prog]$ gdb ./calcul
GNU gdb (GDB) Fedora Linux 13.1-3.fc37
Copyright (C) 2023 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
```

- Для запуска программы внутри отладчика я ввела команду run (рис. 10).

```
(gdb) run
Starting program: /home/mkgolovanova/work/os/lab_prog/calcul
Downloading separate debug info for system-supplied DSO at 0x7ffff7fc6000
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 6
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): +
Второе слагаемое: 2495
2501.00
[Inferior 1 (process 4066) exited normally]
(gdb)
```

{.column

- Для постраничного (по 9 строк) просмотра исходного кода я использовала команду list (рис. 11).

```
(gdb) list
Downloading source file /usr/src/debug/glibc-2.36-9.fc37.x86_64/elf/<built-in>
1      /usr/src/debug/glibc-2.36-9.fc37.x86_64/elf/<built-in>: Каталог не пуст.
(gdb) █
```

{.column

- Для просмотра строк с 12 по 15 основного файла я использовала list с параметрами (рис. 12).

```
(gdb) list 12,15
Downloading source file /usr/src/debug/glibc-2.36-9.fc37.x86_64/elf/<built-in>
Specified first and last lines are in different files.
(gdb)
```

{.column



- Для просмотра определённых строк не основного файла я использовала list с параметрами (рис. 13).

```
specified first and last lines are in different files.
```

```
(gdb) list calculate.c:20,29
```

```
No source file named calculate.c.
```

```
(gdb) █
```

{.column

- Я установила точку останова в файле calculate.c на строке номер 21 и вывела информацию об имеющихся в проекте точка останова (рис. 14).

```
(gdb) list calculate.c:20,27
No source file named calculate.c.
(gdb) break 21
No line 21 in the current file.
Make breakpoint pending on future shared library load? (y or [n])
(gdb) info breakpoints
No breakpoints or watchpoints.
(gdb) Quit
(gdb) break 21
No line 21 in the current file.
Make breakpoint pending on future shared library load? (y or [n]) y
Breakpoint 1 (21) pending.
(gdb) info breakpoints
```

Num	Type	Disp	Enb	Address	What
-----	------	------	-----	---------	------

6.

- Я запустила программу внутри отладчика и убедилась, что программа остановится в момент прохождения точки останова (рис. 15).

```
(gdb) run
Starting program: /home/mkgolovanova/work/os/lab_prog/calcul
Downloading source file /usr/src/debug/glibc-2.36-9.fc37.x86_64/stdio-common/errname.c
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Downloading source file /usr/src/debug/glibc-2.36-9.fc37.x86_64/stdio-common/errname.c
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -
Вычитаемое: backtrace
5.00
[Inferior 1 (process 4419) exited normally]
(gdb) backtrace
```

- Отладчик выдал следующую информацию (рис. 16).

```
#0  Calculate (Numeral=5, Operation=0x7fffffffdd280 "-")
    at calculate.c:21
```

- Я посмотрела, чему равно на этом этапе значение переменной Numeral и сравнила с результатом вывода на экран после использования команды display Numeral (рис. 17).

```
[Inferior 1 (process 4624) exited normally]
(gdb) print Numeral
No symbol "Numeral" in current context.
(gdb) display Numeral
No symbol "Numeral" in current context.
```

{.column

- Я убрала точки останова (рис. 18).

```
(gdb) info breakpoints
Num      Type      Disp Enb Address                What
1        breakpoint keep y   <MULTIPLE>
1.1              y    0x00007ffff7d66e90 in __get_errname at errname.c:55
1.2              y    0x00007ffff7fee8b0 in __get_errname at errname.c:55
(gdb) delete 1
(gdb) █
```

{.column

- С помощью утилиты splint я проанализировала коды файлов calculate.c и main.c (рис. 19, рис. 20).

```
[mkgolovanova@fedora lab_prog]$ splint calculate.c
Splint 3.1.2 --- 23 Jul 2022

calculate.h:7:37: 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:9:31: 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))
```

```
[mkgolovanova@fedora lab_prog]$ splint main.c
Splint 3.1.2 --- 23 Jul 2022

calculate.h:7:37: 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)
main.c: (in function main)
main.c:14:1: Return value (type int) ignored: scanf("%f", &Num...
    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)
main.c:16:12: Format argument 1 to scanf (%s) expects char * gets char [4] *:
                    &Operation
    Type of parameter is not consistent with corresponding code in format string.
    (Use -formattype to inhibit warning)
    main.c:16:9: Corresponding format code
main.c:16:1: Return value (type int) ignored: scanf("%s", &Ope...

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

## Выводы

---



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.