

# Отчет по выполнению лабораторной работы №13

Дисциплина: операционные системы

---

Астраханцева А. А.

3 мая 2023

Российский университет дружбы народов, Москва, Россия

- Астраханцева Анастасия Александровна
- студентка НКАбд-01-22
- Студ. билет: 1132226437
- Российский университет дружбы народов
- <https://anastasiia7205.github.io/>



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.

1. Ознакомиться с теоритиречским материалом
2. Выполнить все задания из “Последовательность выполнения лабораторной работы”
3. Ответить на контрольные вопросы

## Выполнение лабораторной работы

---


Создаем каталог и нужные файлы в нем

```
4-linux»: это недопустимый идентификатор
[aaastrakhantseva@aaastrakhantseva ~]$ mkdir -p ~/work/os/lab_prog
[aaastrakhantseva@aaastrakhantseva ~]$ cd ~/work/os/lab_prog
[aaastrakhantseva@aaastrakhantseva lab_prog]$ touch calculate.h, calculate.c, main.c
[aaastrakhantseva@aaastrakhantseva lab_prog]$ ls
calculate.c, calculate.h, main.c
[aaastrakhantseva@aaastrakhantseva lab_prog]$
```

Рис. 1: Создание каталога и файлов

## Запись текста программы в файлы

В файл calculate.c, calculate.h и main.c записываем текст программы.

```
Открыть ▾  calculate.h,  
~/work/os/lab_prog  
  
////////////////////////////////////  
// 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)
```

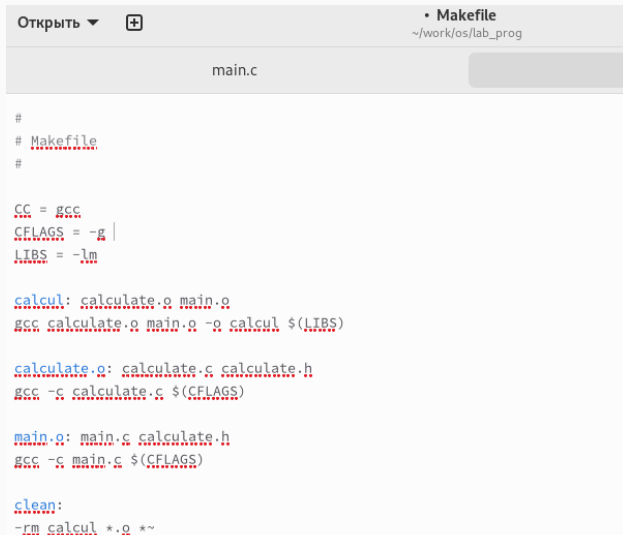
Выполним компиляцию программы посредством gcc

```
[aaastrakhantseva@aaastrakhantseva lab_prog]$ gcc -c calculate.c
[aaastrakhantseva@aaastrakhantseva lab_prog]$ gcc -c main.c
[aaastrakhantseva@aaastrakhantseva lab_prog]$ gcc calculate.o main.o calcul -lm
/usr/bin/ld: невозможно найти calcul: Нет такого файла или каталога
collect2: ошибка: выполнение ld завершилось с кодом возврата 1
[aaastrakhantseva@aaastrakhantseva lab_prog]$ gcc calculate.o main.o -o calcul -lm
[aaastrakhantseva@aaastrakhantseva lab_prog]$
```

Рис. 3: Компиляция программы



Создаем Makefile и записываем в него текст



```
#
# Makefile
#

CC = gcc
CFLAGS = -g
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 *~
```

## Использование make для отладки

Далее использую make для отладки.

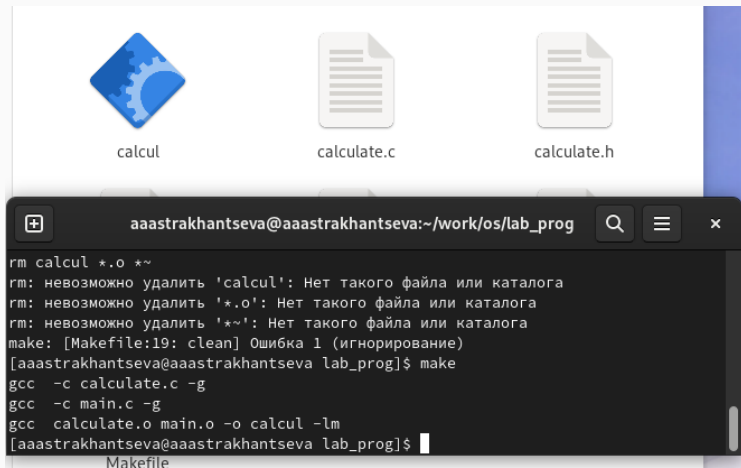


Рис. 5: Make

## Запуск программы

После этого запускаем gdb и вводим run, для того, чтобы запустить нашу программу.

```
make: «calcul» не такой-то особенный.
[aaastrakhantseva@aaastrakhantseva lab_prog]$ gdb ./calcul
GNU gdb (GDB) Fedora Linux 13.1-1.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
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for /home/aaastrakhantseva/work/os/lab_prog/calcul
--Type <RET> for more, q to quit, c to continue without paging--
(No debugging symbols found in ./calcul)
(gdb) run
```

Рис. 6: Запуск нашего файла

Проверяем, что все работает корректно.

```
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): pow
Степень: 3
125.00
[Inferior 1 (process 9009) exited normally]
(gdb) █
```

Рис. 7: Запуск и проверка программы

Для постраничного (по 9 строк) просмотра исходного код использую команду list:

```
(gdb) list
1      //////////////////////////////////////////
2      // main.c
3
4      #include <stdio.h>
5      #include "calculate.h"
6      int
7      main (void)
8      {
9      float Numeral;
10     char Operation[4];
(gdb)
```

Рис. 8: Команда list

Для просмотра строк с 12 по 15 основного файла использую list с параметрами:

```
(gdb) list 12,15
12      printf("Число: ");
13      scanf("%f",&Numeral);
14      printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
15      scanf("%s",&Operation);
(gdb)
```

Рис. 9: Команда list с параметрами

## Команды gdb: list с параметрами из другого файла

Для просмотра определённых строк не основного файла используйте list с параметрами:

```
(gdb) list calculate.c:20,29
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);
28     return(Numeral * SecondNumeral);
29 }
(gdb) █
```

Рис. 10: Просмотр определённых строк не основного файла

Устанавливаю точку останова в файле calculate.c на строке номер 21:

```
(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. 11: Установка точки останова



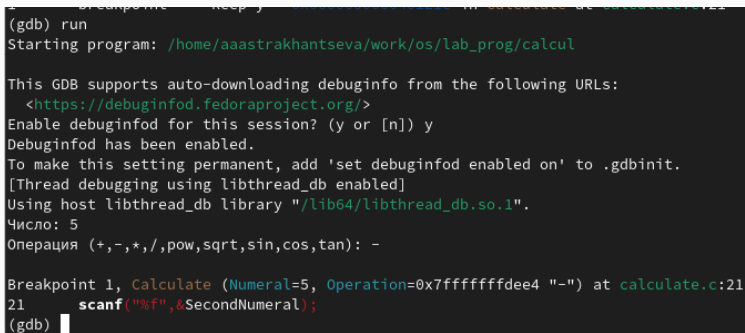
Выведу информацию об имеющихся в проекте точка останова

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

Рис. 12: Информация о точках останова

## Запуск программы с установленной точкой останова

Запускаем программу внутри отладчика и убеждаемся, что программа остановится в момент прохождения точки останова:



```
(gdb) run
Starting program: /home/aaastrakhantseva/work/os/lab_prog/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.
[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=0x7fffffffdee4 "-") at calculate.c:21
21      scanf("%f",&SecondNumeral);
(gdb) |
```

Рис. 13: Запуск программы с установленной точкой останова

Посмотрим, чему равно на этом этапе значение переменной Numeral и сравним с результатом вывода на экран после использования команды `display Numeral`

```
21      scanf ("%i",&secondNumeral);  
(gdb) print Numeral  
$1 = 5  
(gdb) display Numeral  
1: Numeral = 5  
(gdb)
```

Рис. 14: Значение переменной Numeral

Уберем точки останова

```
(gdb) info breakpoints
Num      Type           Disp Enb Address            What
1        breakpoint     keep y   0x0000000000040121e in calculate at calculate.c:21
          breakpoint already hit 1 time
(gdb) delete 1
(gdb) info breakpoints
No breakpoints or watchpoints.
(gdb)
```

Рис. 15: Удаление точки останова

С помощью утилиты splint попробуем проанализировать коды файлов calculate.c и main.c.

```
[aaastrakhantseva@aaastrakhantseva 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)
```

Рис. 16: Вывод splint calculate.c

## Анализ код файлов calculate.c и main.c.

```
Finished checking --- 10 code warnings
[aaastrakhantseva@aaastrakhantseva 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:13:1: 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:15:12: 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:15:9: Corresponding format code
main.c:15:1: Return value (type int) ignored: scanf("%s", &Ope...

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

Рис. 17: Вывод splint main.c

В ходе выполнения лабораторной работы №13 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.