

Отчет по лабораторной работе №13

**Средства, применяемые при разработке программного обеспечения в
ОС типа UNIX/Linux**

Максим Сергеевич Белов

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	13

Список иллюстраций

4.1	Создание каталога и файлов	8
4.2	Создание и редактирование Makefile	8
4.3	gdb calcul	9
4.4	list	10
4.5	Точка останова	10
4.6	splint calculate.c	11
4.7	splint calculate.c	12

List of Tables

3.1 Описание команд gdb 7

1 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.

2 Задание

1. В домашнем каталоге создайте подкаталог `~/work/os/lab_prog`.
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.
3. Выполните компиляцию программы посредством `gcc`
4. При необходимости исправьте синтаксические ошибки.
5. Создайте `Makefile`
6. С помощью `gdb` выполните отладку программы `calcul` исправьте `Makefile`)
7. С помощью утилиты `splint` попробуйте проанализировать коды файлов `calculate.c` и `main.c`.

3 Теоретическое введение

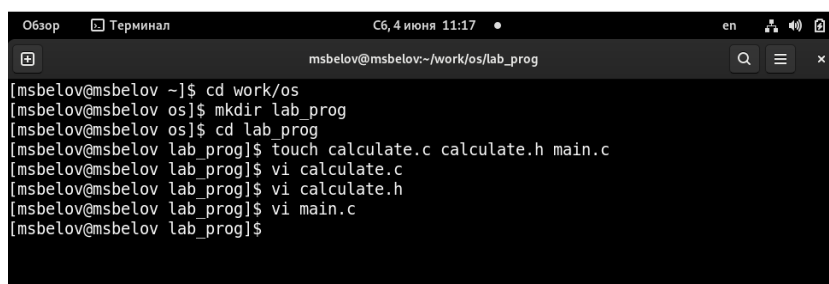
В табл. 3.1 приведено краткое описание команд gdb.

Таблица 3.1: Описание команд gdb

Команда	Описание
<code>backtrace</code>	вывод на экран пути к текущей точке останова (по сути вывод названий всех функций)
<code>break</code>	установить точку останова (в качестве параметра может быть указан номер строки или название функции)
<code>info breakpoints</code>	вывести на экран список используемых точек останова
<code>list</code>	вывести на экран исходный код (в качестве параметра может быть указано название файла и через двоеточие номера начальной и конечной строк)

4 Выполнение лабораторной работы

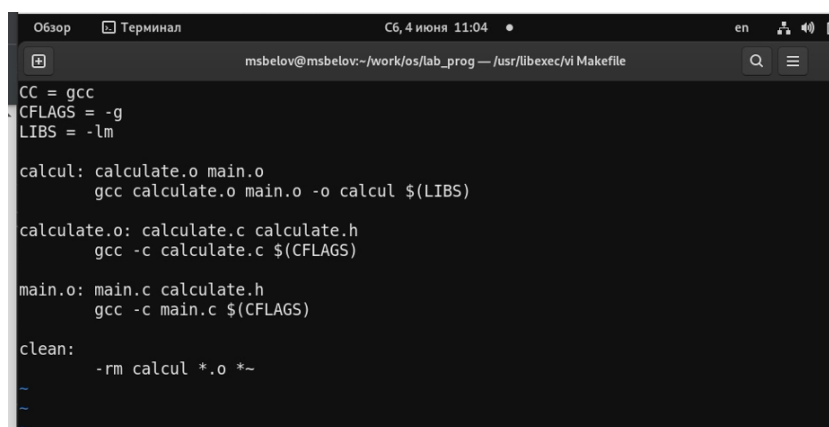
1. В домашнем каталоге создайте подкаталог ~/work/os/lab_prog. Создайте в нём файлы: calculate.h, calculate.c, main.c. (Рис. 4.1)



```
msbelov@msbelov:~/work/os/lab_prog$ cd work/os
msbelov@msbelov:~/work/os$ mkdir lab_prog
msbelov@msbelov:~/work/os$ cd lab_prog
msbelov@msbelov:~/work/os/lab_prog$ touch calculate.c calculate.h main.c
msbelov@msbelov:~/work/os/lab_prog$ vi calculate.c
msbelov@msbelov:~/work/os/lab_prog$ vi calculate.h
msbelov@msbelov:~/work/os/lab_prog$ vi main.c
msbelov@msbelov:~/work/os/lab_prog$
```

Рис. 4.1: Создание каталога и файлов

2. Создадим Makefile и отредактируем его. В этом Makefile в начале задаются параметры в виде названия компилятора и флагов конфигурации к нему(библиотеки). В целом, с помощью этого Makefile мы можем компилировать программу автоматически. (Рис. 4.2)



```
msbelov@msbelov:~/work/os/lab_prog$ vi Makefile
CC = gcc
CFLAGS = -g
LIBS = -lm

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

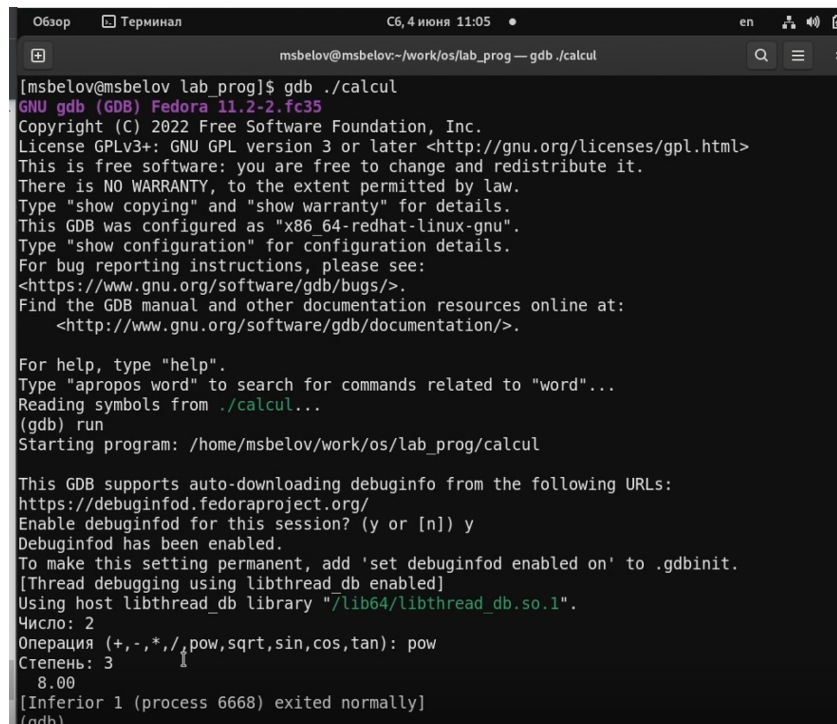
calculate.o: calculate.c calculate.h
gcc -c calculate.c $(CFLAGS)

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

clean:
-rm calcul *.o *~
```

Рис. 4.2: Создание и редактирование Makefile

3. С помощью gdb выполните отладку программы calcul. Также запустим программу внутри отладчика с помощью run. (Рис. 4.3)



```
[msbelov@msbelov 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...
(gdb) run
Starting program: /home/msbelov/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: 2
Операция (+, -, *, /): pow
Степень: 3
8.00
[Inferior 1 (process 6668) exited normally]
(gdb)
```

Рис. 4.3: gdb calcul

Протестируем команды list для вывода строк из файлов. (Рис. 4.4)

```
Обзор Терминал C6, 4 июня 11:07 en
msbelov@msbelov-~/work/os/lab_prog — gdb ./calcul

(gdb) list
1  //////////////////////////////////////////////////
2  // main.c
3
4  #include <stdio.h>
5  #include "calculate.h"
6
7  int
8  main (void)
9  {
10     float Numeral;
(gdb) list 12,15
12     float Result;
13     printf("Число: ");
14     scanf("%f",&Numeral);
15     printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
(gdb) list calculate.c:20,29
20     {
21     printf("Вычитаемое: ");
22     scanf("%f",&SecondNumeral);
23     return(Numeral - SecondNumeral);
24     }
25     else if(strncmp(Operation, "**", 1) == 0)
26     {
27     printf("Множитель: ");
28     scanf("%f",&SecondNumeral);
29     return(Numeral * SecondNumeral);
(gdb) list calculate.c:20,27
20     {
21     printf("Вычитаемое: ");
22     scanf("%f",&SecondNumeral);
23     return(Numeral - SecondNumeral);
24     }
25     else if(strncmp(Operation, "**", 1) == 0)
26     {
```

Рис. 4.4: list

Установим точку останова в calculate.c на 21 строке и затем уберем ее. (Рис. 4.5)

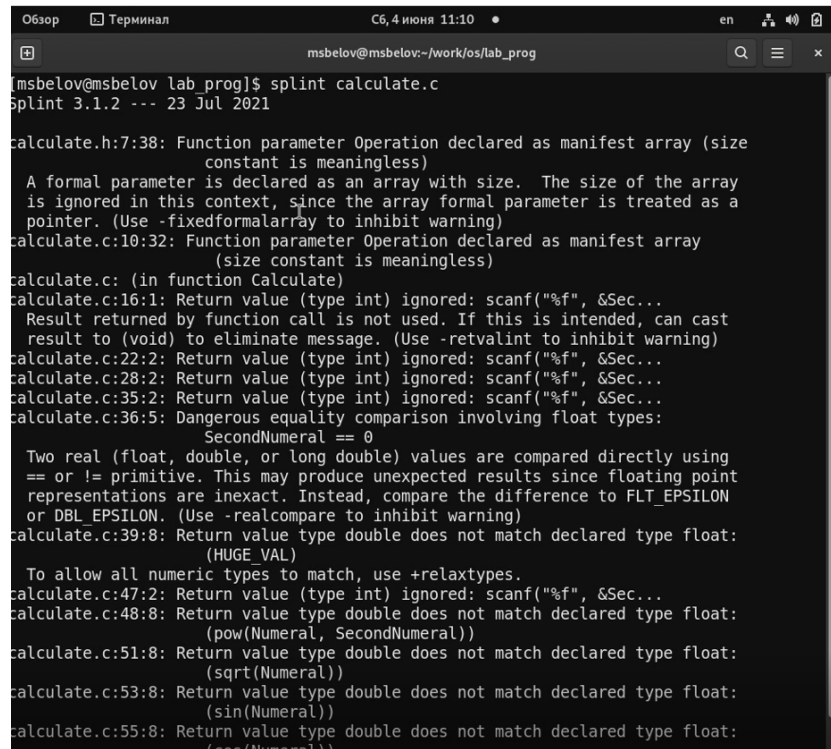
```
(gdb) break 21
Breakpoint 1 at 0x40120f: file calculate.c, line 21.
(gdb) info breakpoints
Num Type Disp Enb Address What
1 breakpoint keep y 0x000000000040120f in Calculate at calculate.c:21
(gdb) run
Starting program: /home/msbelov/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -

Breakpoint 1, Calculate (Numeral=5, Operation=0x7fffffffddc4 "-") at calculate.c:21
21 printf("Вычитаемое: ");
(gdb) backtrace
#0 Calculate (Numeral=5, Operation=0x7fffffffddc4 "-") at calculate.c:21
#1 0x00000000004014eb in main () at main.c:17
(gdb) print Numeral
$1 = 5
(gdb) display Numeral
1: Numeral = 5
(gdb) info breakpoints
Num Type Disp Enb Address What
1 breakpoint keep y 0x000000000040120f in Calculate at calculate.c:21
breakpoint already hit 1 time
(gdb) delete 1
(gdb) info breakpoints
No breakpoints or watchpoints.
```

Рис. 4.5: Точка останова

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

calculate.c (Рис. 4.6)



```
Обзор Терминал C6,4 июня 11:10 en
msbelov@msbelov:~/work/os/lab_prog
[msbelov@msbelov lab_prog]$ splint calculate.c
Splint 3.1.2 --- 23 Jul 2021

calculate.h:7:38: 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:10:32: Function parameter Operation declared as manifest array
(size constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:16:1: 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:22:2: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:28:2: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:35:2: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:36:5: 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:39:8: Return value type double does not match declared type float:
(HUGE_VAL)
To allow all numeric types to match, use +relaxtypes.
calculate.c:47:2: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:48:8: Return value type double does not match declared type float:
(pow(Numeral, SecondNumeral))
calculate.c:51:8: Return value type double does not match declared type float:
(sqrt(Numeral))
calculate.c:53:8: Return value type double does not match declared type float:
(sin(Numeral))
calculate.c:55:8: Return value type double does not match declared type float:
(cos(Numeral))
```

Рис. 4.6: splint calculate.c

main.c (Рис. 4.7)

```
Обзор Терминал C6, 4 июня 11:10 en
msbelov@msbelov:~/work/os/lab_prog

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

calculate.h:7:38: 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
[msbelov@msbelov lab_prog]$
```

Рис. 4.7: splint calculate.c

5 Выводы

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