

# Отчёт по лабораторной работе №13

Операционные системы

Балханова Алтана Юрьевна

# Содержание

Цель работы	5
Выполнение лабораторной работы	6
Контрольные вопросы	22
Выводы	25

# Список иллюстраций

0.1	Создание подкаталога . . . . .	6
0.2	Создание файлов . . . . .	7
0.3	calculate.h . . . . .	8
0.4	calculate.c . . . . .	9
0.5	main.c . . . . .	10
0.6	Компиляция . . . . .	10
0.7	Makefile . . . . .	11
0.8	Отладка . . . . .	12
0.9	run . . . . .	12
0.10	Просмотр . . . . .	13
0.11	Просмотр . . . . .	14
0.12	Просмотр . . . . .	15
0.13	Точка останова . . . . .	16
0.14	Информация о точках останова . . . . .	16
0.15	Проверка точки останова . . . . .	17
0.16	Значение переменной . . . . .	18
0.17	Сравнение . . . . .	19
0.18	Delete 1 . . . . .	19
0.19	Calculte.c . . . . .	20
0.20	Main.c . . . . .	21

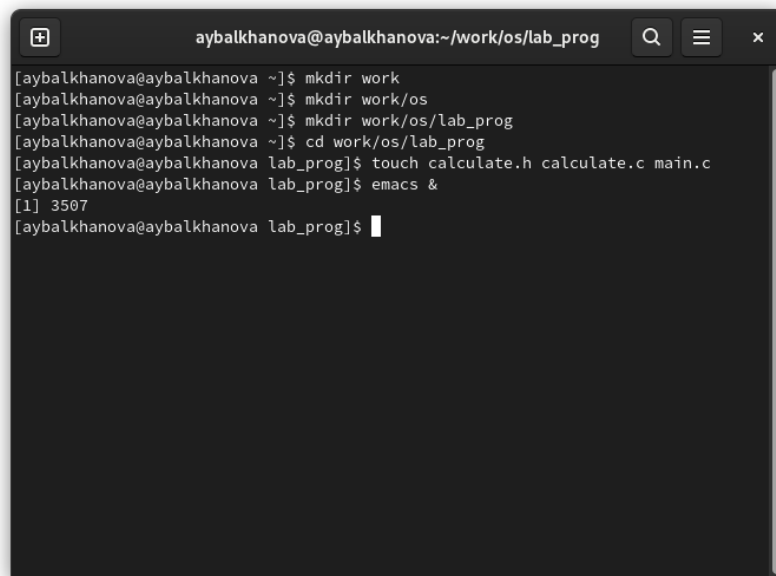
## Список таблиц

## 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.

# Выполнение лабораторной работы

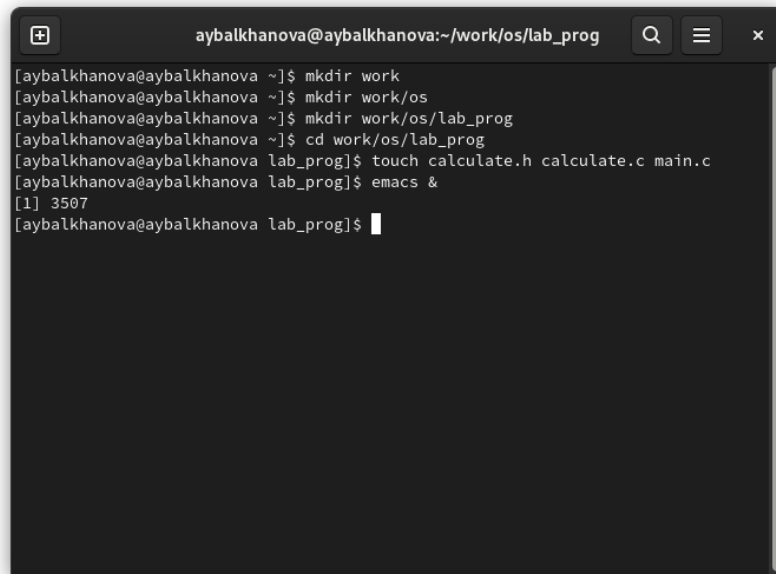
1. В домашнем каталоге создала подкаталог `~/work/os/lab_prog`. (рис. 0.1).



```
aybalkhanova@aybalkhanova:~/work/os/lab_prog
[aybalkhanova@aybalkhanova ~]$ mkdir work
[aybalkhanova@aybalkhanova ~]$ mkdir work/os
[aybalkhanova@aybalkhanova ~]$ mkdir work/os/lab_prog
[aybalkhanova@aybalkhanova ~]$ cd work/os/lab_prog
[aybalkhanova@aybalkhanova lab_prog]$ touch calculate.h calculate.c main.c
[aybalkhanova@aybalkhanova lab_prog]$ emacs &
[1] 3507
[aybalkhanova@aybalkhanova lab_prog]$
```

Рис. 0.1: Создание подкаталога

2. Создала в нём файлы: `calculate.h`, `calculate.c`, `main.c`. (рис. 0.2, 0.3, 0.4, 0.5).



```
aybalkhanova@aybalkhanova:~/work/os/lab_prog
[aybalkhanova@aybalkhanova ~]$ mkdir work
[aybalkhanova@aybalkhanova ~]$ mkdir work/os
[aybalkhanova@aybalkhanova ~]$ mkdir work/os/lab_prog
[aybalkhanova@aybalkhanova ~]$ cd work/os/lab_prog
[aybalkhanova@aybalkhanova lab_prog]$ touch calculate.h calculate.c main.c
[aybalkhanova@aybalkhanova lab_prog]$ emacs &
[1] 3507
[aybalkhanova@aybalkhanova lab_prog]$
```

Рис. 0.2: Создание файлов

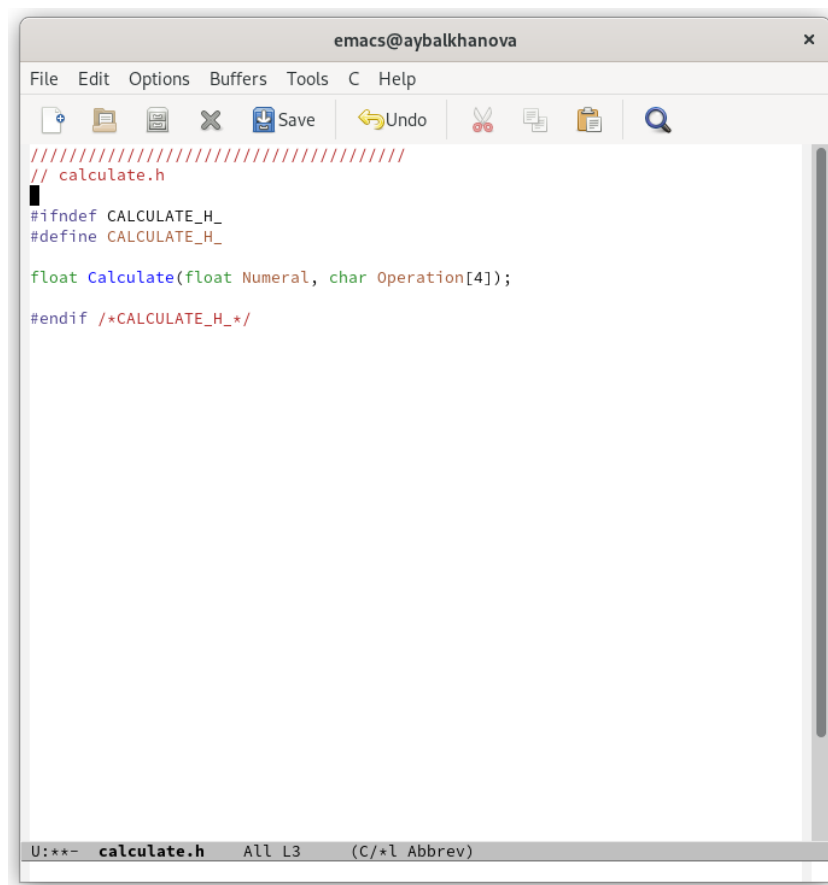
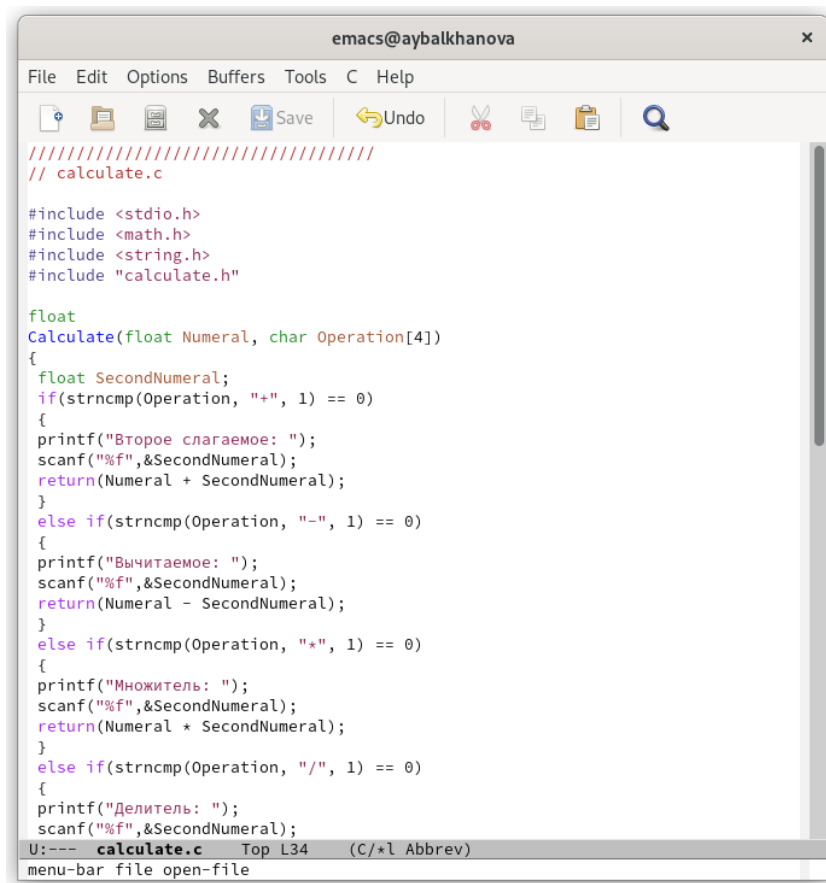


Рис. 0.3: calculate.h





```
////////////////////  
// 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);  
    }  
}
```

U:--- **calculate.c** Top L34 (C/\*l Abbrev)  
menu-bar file open-file

Рис. 0.4: calculate.c

```
emacs@aybalkhanova
File Edit Options Buffers Tools C Isearch Help
Repeat Forward Abort Undo Replace Show Hits Help

////////////////////
// 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);
    printf("%.2f\n",Result);
    return 0;
}

U:*- main.c All L19 (C/*l Abbrev Isearch)
I-search:
```

Рис. 0.5: main.c

3. Выполнила компиляцию программы посредством gcc (рис. 0.6).

```
aybalkhanova@aybalkhanova:~/work/os/lab_prog — gdb ./calcul
[aybalkhanova@aybalkhanova ~]$ mkdir work
[aybalkhanova@aybalkhanova ~]$ mkdir work/os
[aybalkhanova@aybalkhanova ~]$ mkdir work/os/lab_prog
[aybalkhanova@aybalkhanova ~]$ cd work/os/lab_prog
[aybalkhanova@aybalkhanova lab_prog]$ touch calculate.h calculate.c main.c
[aybalkhanova@aybalkhanova lab_prog]$ emacs &
[1] 3507
[aybalkhanova@aybalkhanova lab_prog]$ gcc -c calculate.c
[1]+  Done                  emacs
[aybalkhanova@aybalkhanova lab_prog]$ gcc -c main.c
[aybalkhanova@aybalkhanova lab_prog]$ gcc calculate.o main.o -o calcul -lm
[aybalkhanova@aybalkhanova lab_prog]$ touch Makefile
```

Рис. 0.6: Компиляция

4. Создала Makefile со следующим содержанием (рис. 0.7):

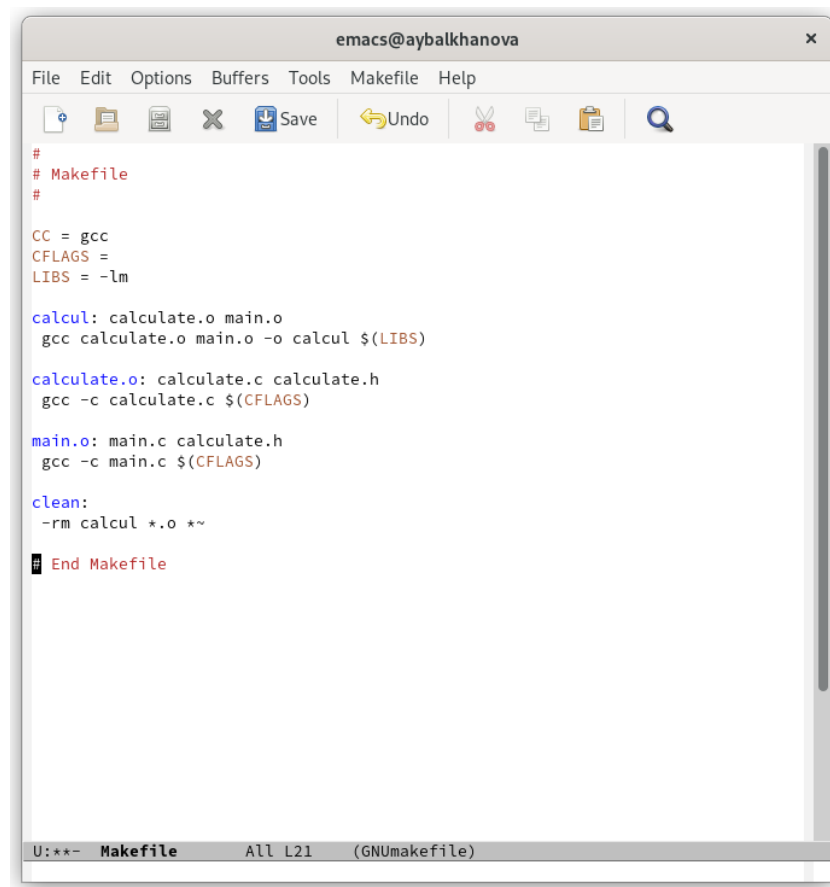


Рис. 0.7: Makefile

Здесь в первой строке `calcul` — цель, `calculate.o main.o` — название файла, который мы хотим скомпилировать; во второй строке, начиная с табуляции, задана команда компиляции `gcc` с опциями.

1. С помощью `gdb` выполнила отладку программы `calcul` (перед использованием `gdb` исправила Makefile):

1. Запустила отладчик GDB, загрузив в него программу для отладки (рис. 0.8):

```
aybalkhanova@aybalkhanova:~/work/os/lab_prog — gdb ./calcul
[aybalkhanova@aybalkhanova ~]$ mkdir work
[aybalkhanova@aybalkhanova ~]$ mkdir work/os
[aybalkhanova@aybalkhanova ~]$ mkdir work/os/lab_prog
[aybalkhanova@aybalkhanova ~]$ cd work/os/lab_prog
[aybalkhanova@aybalkhanova lab_prog]$ touch calculate.h calculate.c main.c
[aybalkhanova@aybalkhanova lab_prog]$ emacs &
[1] 3507
[aybalkhanova@aybalkhanova lab_prog]$ gcc -c calculate.c
[1]+  Done                  emacs
[aybalkhanova@aybalkhanova lab_prog]$ gcc -c main.c
[aybalkhanova@aybalkhanova lab_prog]$ gcc calculate.o main.o -o calcul -lm
[aybalkhanova@aybalkhanova lab_prog]$ touch Makefile
[aybalkhanova@aybalkhanova lab_prog]$ emacs &
[1] 4139
[aybalkhanova@aybalkhanova 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:
```

Рис. 0.8: Отладка

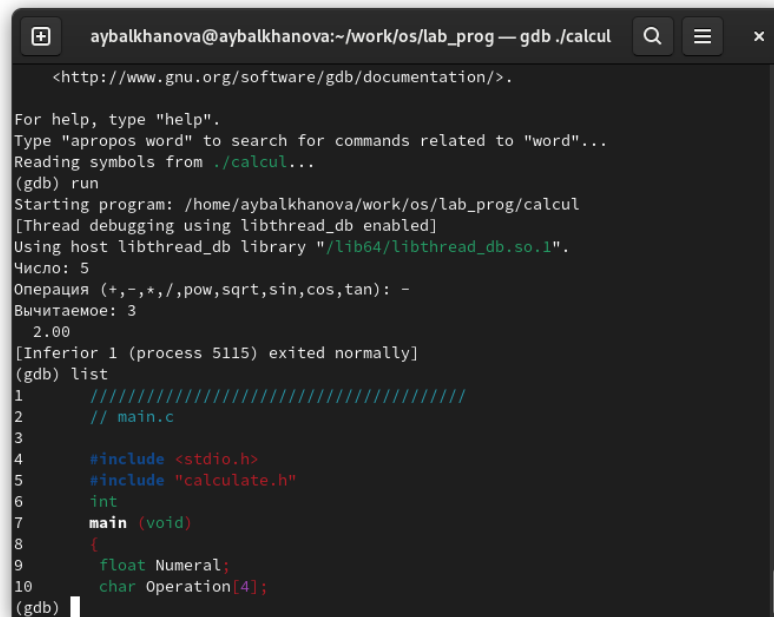
2. Для запуска программы внутри отладчика ввела команду run (рис. 0.9):

```
aybalkhanova@aybalkhanova:~/work/os/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) run
Starting program: /home/aybalkhanova/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): -
Вычитаемое: 3
2.00
[Inferior 1 (process 5115) exited normally]
(gdb)
```

Рис. 0.9: run

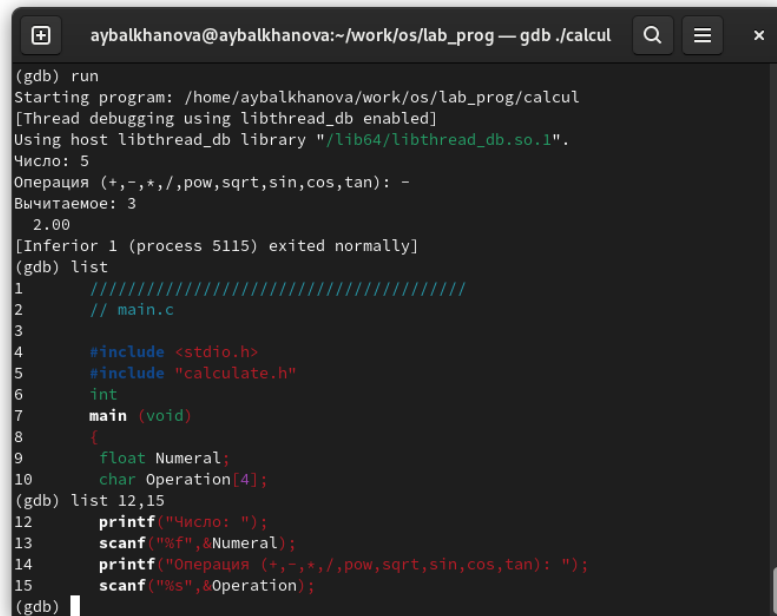
3. Для постраничного (по 9 строк) просмотра исходного код использовала команду list (рис. 0.10):



```
aybalkhanova@aybalkhanova:~/work/os/lab_prog — gdb ./calcul
<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/aybalkhanova/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): -
Вычитаемое: 3
2.00
[Inferior 1 (process 5115) exited normally]
(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. 0.10: Просмотр

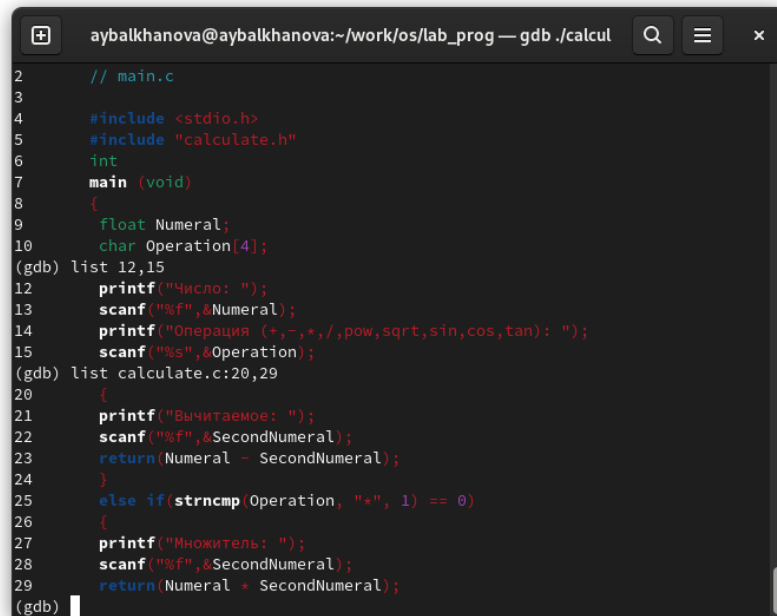
4. Для просмотра строк с 12 по 15 основного файла используйте list с параметрами (рис. 0.11):



```
aybalkhanova@aybalkhanova:~/work/os/lab_prog — gdb ./calcul
(gdb) run
Starting program: /home/aybalkhanova/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): -
Вычитаемое: 3
2.00
[Inferior 1 (process 5115) exited normally]
(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) list 12,15
12     printf("Число: ");
13     scanf("%f",&Numeral);
14     printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
15     scanf("%s",&Operation);
(gdb)
```

Рис. 0.11: Просмотр

5. Для просмотра определённых строк не основного файла используйте `list` с параметрами (рис. 0.12):



```
aybalkhanova@aybalkhanova:~/work/os/lab_prog — gdb ./calcul
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) list 12,15
12    printf("Число: ");
13    scanf("%f",&Numeral);
14    printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
15    scanf("%s",&Operation);
(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)
```

Рис. 0.12: Просмотр

6. Установила точку останова в файле calculate.c на строке номер 21 (рис. 0.13):

```
aybalkhanova@aybalkhanova:~/work/os/lab_prog — gdb ./calcul
13     scanf("%f",&Numeral);
14     printf("Операция (+,-,+,/,pow,sqrt,sin,cos,tan): ");
15     scanf("%s",&Operation);
(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     {
27     printf("Множитель: ");
(gdb) break 21
Breakpoint 1 at 0x40120f: file calculate.c, line 21.
(gdb)
```

Рис. 0.13: Точка останова

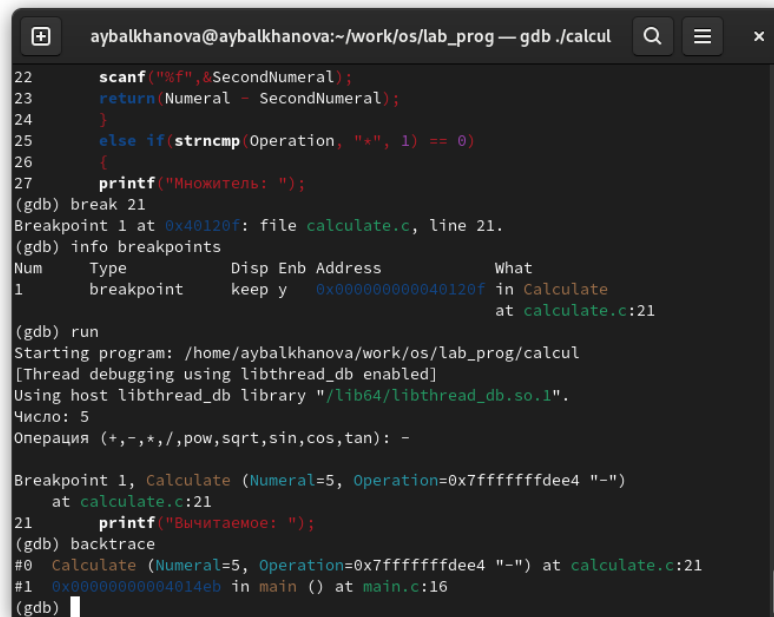
7. Вывела информацию об имеющихся в проекте точка останова (рис. 0.14):

```
aybalkhanova@aybalkhanova:~/work/os/lab_prog — gdb ./calcul
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     {
27     printf("Множитель: ");
(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)
```

Рис. 0.14: Информация о точках останова



8. Запустила программу внутри отладчика и убедилась, что программа остановится в момент прохождения точки останова (рис. 0.15):



```
aybalkhanova@aybalkhanova:~/work/os/lab_prog — gdb ./calcul
22     scanf("%f",&SecondNumeral);
23     return(Numeral - SecondNumeral);
24 }
25 else if(strcmp(Operation, "+", 1) == 0)
26 {
27     printf("Множитель: ");
(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/aybalkhanova/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=0x7fffffffdee4 "-")
  at calculate.c:21
21     printf("Вычитаемое: ");
(gdb) backtrace
#0 Calculate (Numeral=5, Operation=0x7fffffffdee4 "-") at calculate.c:21
#1 0x000000004014eb in main () at main.c:16
(gdb)
```

Рис. 0.15: Проверка точки останова

9. Посмотрела, чему равно на этом этапе значение переменной Numeral, введя (рис. 0.16):

```
aybalkhanova@aybalkhanova:~/work/os/lab_prog — gdb ./calcul
24     }
25     else if(strncmp(Operation, "+", 1) == 0)
26     {
27         printf("Множитель: ");
(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/aybalkhanova/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=0x7fffffffdee4 "-")
    at calculate.c:21
21     printf("Вычитаемое: ");
(gdb) backtrace
#0  Calculate (Numeral=5, Operation=0x7fffffffdee4 "-") at calculate.c:21
#1  0x00000000004014eb in main () at main.c:16
(gdb) print Numeral
$1 = 5
(gdb)
```

Рис. 0.16: Значение переменной

10. Сравнила с результатом вывода на экран после использования команды (рис. 0.17):

```
aybalkhanova@aybalkhanova:~/work/os/lab_prog — gdb ./calcul
26 {
27     printf("Множитель: ");
(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/aybalkhanova/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=0x7fffffffdee4 "-")
at calculate.c:21
21     printf("Вычитаемое: ");
(gdb) backtrace
#0 Calculate (Numeral=5, Operation=0x7fffffffdee4 "-") at calculate.c:21
#1 0x00000000004014eb in main () at main.c:16
(gdb) print Numeral
$1 = 5
(gdb) display Numeral
1: Numeral = 5
(gdb)
```

Рис. 0.17: Сравнение

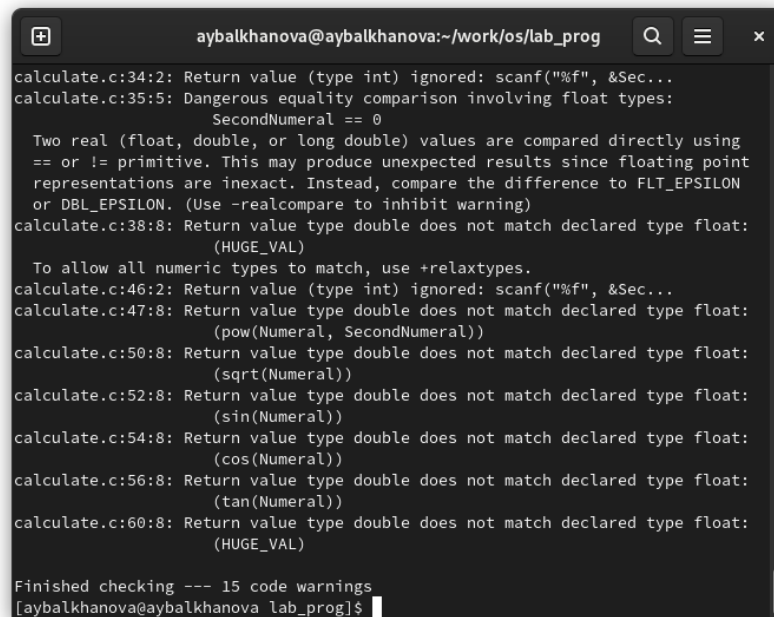
11. Убрала точки останова (рис. 0.18):

```
aybalkhanova@aybalkhanova:~/work/os/lab_prog — gdb ./calcul
(gdb) run
Starting program: /home/aybalkhanova/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=0x7fffffffdee4 "-")
at calculate.c:21
21     printf("Вычитаемое: ");
(gdb) backtrace
#0 Calculate (Numeral=5, Operation=0x7fffffffdee4 "-") at calculate.c:21
#1 0x00000000004014eb in main () at main.c:16
(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.
(gdb)
```

Рис. 0.18: Delete 1

2. С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c (рис. 0.19, 0.20):



```
aybalkhanova@aybalkhanova:~/work/os/lab_prog
calculate.c:34:2: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:35:5: 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:38:8: Return value type double does not match declared type float:
    (HUGE_VAL)
    To allow all numeric types to match, use +relaxtypes.
calculate.c:46:2: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:47:8: Return value type double does not match declared type float:
    (pow(Numeral, SecondNumeral))
calculate.c:50:8: Return value type double does not match declared type float:
    (sqrt(Numeral))
calculate.c:52:8: Return value type double does not match declared type float:
    (sin(Numeral))
calculate.c:54:8: Return value type double does not match declared type float:
    (cos(Numeral))
calculate.c:56:8: Return value type double does not match declared type float:
    (tan(Numeral))
calculate.c:60:8: Return value type double does not match declared type float:
    (HUGE_VAL)

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

Рис. 0.19: Calculate.c

```
aybalkhanova@aybalkhanova:~/work/os/lab_prog
(tan(Numeral))
calculate.c:60:8: Return value type double does not match declared type float:
(HUGE_VAL)

Finished checking --- 15 code warnings
[aybalkhanova@aybalkhanova lab_prog]$ splint main.c
Splint 3.1.2 --- 23 Jul 2021

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:13:2: 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:15:13: 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:15:10: Corresponding format code
main.c:15:2: Return value (type int) ignored: scanf("%s", &Ope...

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

Рис. 0.20: Main.c

# Контрольные вопросы

1. Используя команду `man`, можно узнать о возможностях программ `gcc`, `make`, `gdb` и др.
2. 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.
3. 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. Таким образом, `gcc` по расширению (суффиксу) `.c` распознает тип файла для компиляции и формирует объектный модуль — файл с расширением `.o`.
4. 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.

5. `hello: main.c gcc -o hello main.c` Здесь в первой строке `hello` — цель, `main.c` — название файла, который мы хотим скомпилировать; во второй строке, начиная с табуляции, задана команда компиляции `gcc` с опциями.
6. `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
7. Для использования GDB необходимо скомпилировать анализируемый код программы таким образом, чтобы отладочная информация содержалась в результирующем бинарном файле. После этого для начала работы с `gdb` необходимо в командной строке ввести одноимённую команду, указав в качестве аргумента анализируемый бинарный файл. Затем можно использовать по мере необходимости различные команды `gdb`: `run`, `backtrace`, `delete`, `display`, `list`, `info`

breakpoints.

8. Для облегчения понимания исходного кода используются комментарии.
9. анализатор `splint` генерирует комментарии с описанием разбора кода программы и осуществляет общий контроль, обнаруживая такие ошибки, как одинаковые объекты, определённые в разных файлах, или объекты, чьи значения не используются в работе программы, переменные с некорректно заданными значениями и типами и многое другое.



## 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.