

Лабораторная работа 13

Сафин Андрей Алексеевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	14

Список иллюстраций

4.1	Создание директории и файлов	8
4.2	Компиляция файлов	8
4.3	Исправленный текст Makefile	9
4.4	Makefile	10
4.5	Использование run в GDB	10
4.6	Действия с list	11
4.7	Действия с точками останова	12
4.8	Splint calculate.c	13
4.9	Splint main.c	13

Список таблиц

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

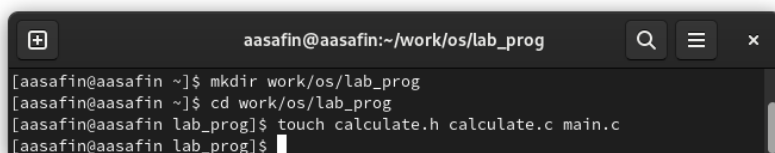
Написать программу-калькулятор и отладить её.

3 Теоретическое введение

В процессе разработки ПО неизбежно возникают различного рода ошибки. Для их нахождения и исправления применяются специальные утилиты - средства отладки. Примерами могут являться: - GDB - Splint

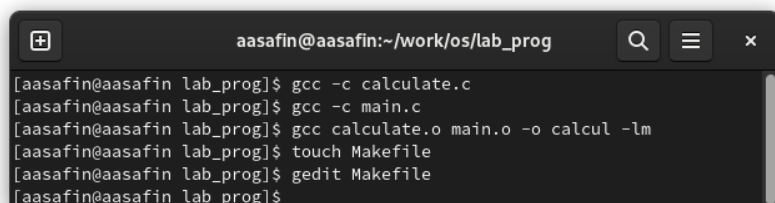
4 Выполнение лабораторной работы

1. Создана подкаталог ~/work/os/lab_prog (рис. 4.1).
2. Созданы файлы calculate.h, calculate.c, main.c (рис. 4.1).
3. Выполнена компиляция файлов (рис. 4.2).
4. Синтаксических ошибок не выявлено.
5. Создан Makefile (рис. 4.2).
6. Makefile исправлен (строка 6) (рис. 4.3). Этот файл позволяет с помощью make автоматически компилировать calculate.c, calculate.h, main.h в исполняемый файл calcul.



```
aasafin@aasafin:~/work/os/lab_prog
[aasafin@aasafin ~]$ mkdir work/os/lab_prog
[aasafin@aasafin ~]$ cd work/os/lab_prog
[aasafin@aasafin lab_prog]$ touch calculate.h calculate.c main.c
[aasafin@aasafin lab_prog]$
```

Рис. 4.1: Создание директории и файлов



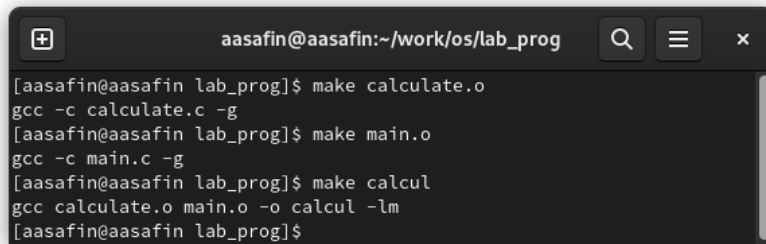
```
aasafin@aasafin:~/work/os/lab_prog
[aasafin@aasafin lab_prog]$ gcc -c calculate.c
[aasafin@aasafin lab_prog]$ gcc -c main.c
[aasafin@aasafin lab_prog]$ gcc calculate.o main.o -o calcul -lm
[aasafin@aasafin lab_prog]$ touch Makefile
[aasafin@aasafin lab_prog]$ gedit Makefile
[aasafin@aasafin lab_prog]$
```

Рис. 4.2: Компиляция файлов


```
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
```

Рис. 4.3: Исправленный текст Makefile

Файлы скомпилированы заново с помощью make (рис. 4.4). Запущен отладчик GDB, использована команда run (рис. 4.5). Произведены различные действия с list (рис. 4.6). Создана точка останова, после выполнения выведена информация о ней и о значении Numeral в момент точки останова, после чего она удалена (рис. 4.7).



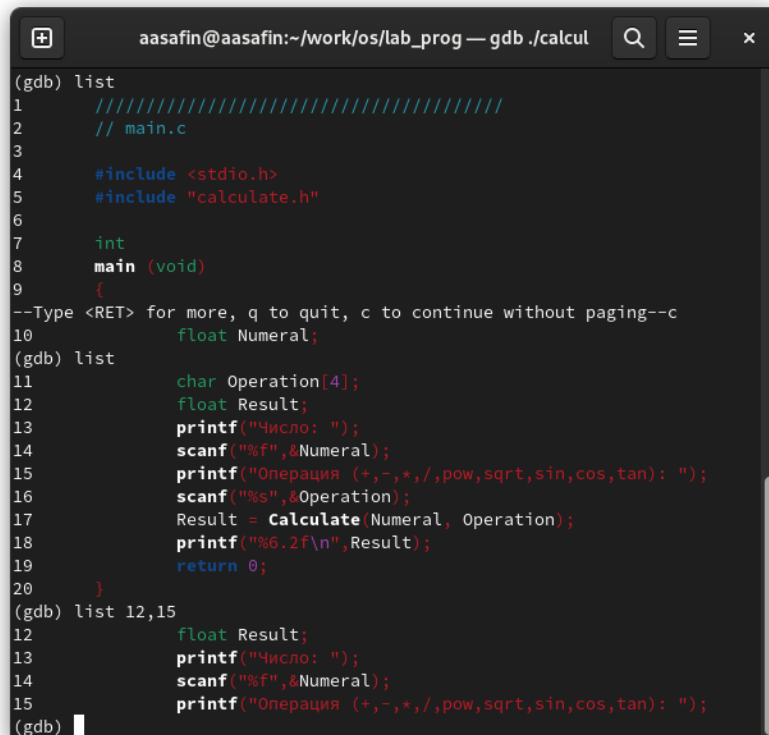
```
aasafin@aasafin:~/work/os/lab_prog
[aasafin@aasafin lab_prog]$ make calculate.o
gcc -c calculate.c -g
[aasafin@aasafin lab_prog]$ make main.o
gcc -c main.c -g
[aasafin@aasafin lab_prog]$ make calcul
gcc calculate.o main.o -o calcul -lm
[aasafin@aasafin lab_prog]$
```

Рис. 4.4: Makefile



```
aasafin@aasafin:~/work/os/lab_prog — gdb ./calcul
(gdb) run
Starting program: /home/aasafin/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 8
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): *
Множитель: 2
16.00
[Inferior 1 (process 4179) exited normally]
(gdb)
```

Рис. 4.5: Использование run в GDB



The screenshot shows a GDB terminal window with the title bar 'aasafin@aasafin:~/work/os/lab_prog — gdb ./calcul'. The terminal displays the following sequence of commands and output:

```
(gdb) list
1  //////////////////////////////////////
2  // main.c
3
4  #include <stdio.h>
5  #include "calculate.h"
6
7  int
8  main (void)
9  {
--Type <RET> for more, q to quit, c to continue without paging--
10     float Numeral;
(gdb) list
11     char Operation[4];
12     float Result;
13     printf("Число: ");
14     scanf("%f",&Numeral);
15     printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
16     scanf("%s",&Operation);
17     Result = calculate(Numeral, Operation);
18     printf("%.2f\n",Result);
19     return 0;
20 }
(gdb) list 12,15
12     float Result;
13     printf("Число: ");
14     scanf("%f",&Numeral);
15     printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
(gdb)
```

Рис. 4.6: Действия с list

```
aasafin@aasafin:~/work/os/lab_prog — gdb ./calcul
(gdb) break 21
Breakpoint 2 at 0x40120f: file calculate.c, line 21.
(gdb) info breakpoints
Num      Type          Disp Enb Address            What
2        breakpoint    keep y  0x000000000040120f in calculate
                                           at calculate.c:21
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/aasafin/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 2
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): tan
-2.19
[Inferior 1 (process 4287) exited normally]
(gdb) run
Starting program: /home/aasafin/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 2, Calculate (Numeral=5, Operation=0x7fffffffdfc4 "-") at ca
lculate.c:21
21                                     printf("Вычитаемое: ");
(gdb) backtrace
#0 Calculate (Numeral=5, Operation=0x7fffffffdfc4 "-")
    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
2        breakpoint    keep y  0x000000000040120f in calculate
                                           at calculate.c:21
        breakpoint already hit 1 time
(gdb) delete 2
(gdb)
```

Рис. 4.7: Действия с точками останова

7. Произведен анализ файлов calculate.c и main.c с помощью splint (рис. 4.8-4.9).

```
aasafin@aasafin:~/work/os/lab_prog
[aaasafin@aaasafin lab_prog]$ splint calculate.c
Splint 3.1.2 --- 22 Jan 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:10:31: Function parameter Operation declared as manifest array
        (size constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:16:4: 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:22:4: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:28:4: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:34:4: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:35:7: 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:38:12: Return value type double does not match declared type float:
        (HUGE_VAL)
    To allow all numeric types to match, use +relaxtypes.
calculate.c:46:4: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:47:10: Return value type double does not match declared type float:
        (pow(Numeral, SecondNumeral))
calculate.c:50:9: Return value type double does not match declared type float:
        (sqrt(Numeral))
calculate.c:52:9: Return value type double does not match declared type float:
        (sin(Numeral))
calculate.c:54:9: Return value type double does not match declared type float:
        (cos(Numeral))
calculate.c:56:9: Return value type double does not match declared type float:
        (tan(Numeral))
calculate.c:60:10: Return value type double does not match declared type float:
        (HUGE_VAL)

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

Рис. 4.8: Splint calculate.c

```
aasafin@aasafin:~/work/os/lab_prog
[aaasafin@aaasafin lab_prog]$ splint main.c
Splint 3.1.2 --- 22 Jan 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:14:2: 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:16:13: 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:16:10: Corresponding format code
main.c:16:2: Return value (type int) ignored: scanf("%s", &Ope...

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

Рис. 4.9: Splint main.c

5 Выводы

Навык работы со средствами отладки в GNU Linux получен.