Лабораторная работа №13

Операционные системы

Кирилюк С. А.

Содержание

Список иллюстраций

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 Выполнение лабораторной работы

Перед выполнением заданий в домашнем каталоге создала подкаталог ~/work/os/lab_prog., а также файлы: calculate.h, calculate.c, main.c.(рис. 2.1).

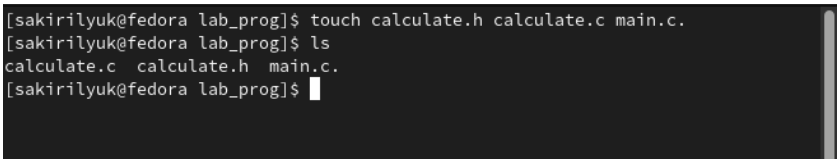

Рис. 2.1: Создание файлов

При помощи редактора для данных файлов написала скрипты (рис. 2.2), (рис. 2.3), (рис. 2.4).

```
////////////////////////////////////
// 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)
```

Рис. 2.2: Скрипт для calculate.c

```
/////////////////////////////////////
// calculate.h

#ifndef CALCULATE_H_
#define CALCULATE_H_

float Calculate(float Numeral, char Operation[4]);

#endif /*CALCULATE_H_*/
```

Рис. 2.3: Скрипт для calculate.h

```
/////////////////////////////////////
// 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("%6.2f\n",Result);
return 0;
}
```

Рис. 2.4: Скрипт для main.c

После чего выполнила компиляцию программы посредством gcc (рис. 2.5).

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

Рис. 2.5: Компиляция

Затем создала Makefile и написала для него скрипт (рис. 2.6). Этот файл используется для автоматической компиляции main.c, calculate.c и создание из них исполняемого файла calcul. Помимо этого, в файле также есть функция ‘clean’, используемая для удаления всех файлов.

Затем я изменила его содержание (рис. 2.7) и проверила его работу (рис. 2.8).

```
#
# 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 *~

# End Makefile
```

Рис. 2.6: Создание файла Makefile

```
#
# Makefile
#

CC = gcc
CFLAGS = -g
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 *~

# End Makefile
```

Рис. 2.7: Изменения в Makefile

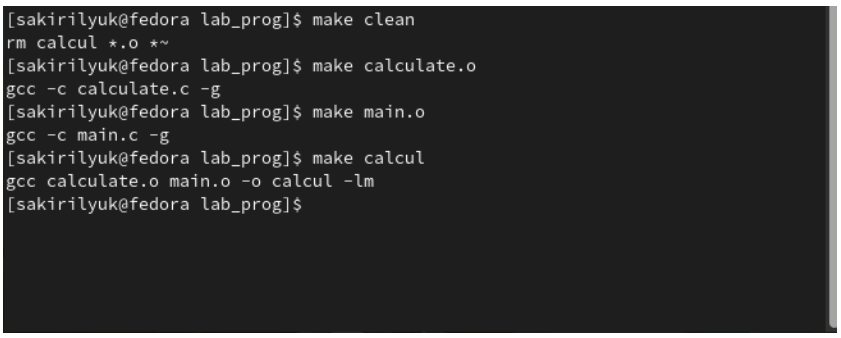

Рис. 2.8: Проверка работы Makefile

С помощью gdb я начала выполнение отладки программы calcul. Сначала я запустила отладчик GDB, загрузив в него программу для отладки (рис. 2.9). Для запуска программы внутри отладчика ввела команду run (рис. 2.10).

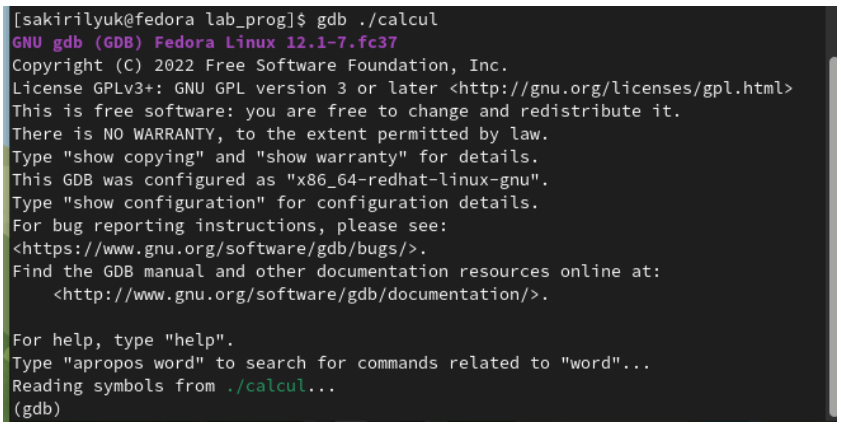

Рис. 2.9: Запуск отладчика

```
(gdb) run
Starting program: /home/sakirilyuk/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]) n
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' 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): +
Второе слагаемое: 9
 14.00
[Inferior 1 (process 4003) exited normally]
Missing separate debuginfos, use: dnf debuginfo-install glibc-2.36-9.fc37.x86_64
(gdb)
```

Рис. 2.10: Запуск программы внутри отладчика

Для постраничного (по 9 строк) просмотра исходного код использовала команду list (рис. 2.11), для просмотра строк с 12 по 15 основного файла - list с параметрами (рис. 2.12). Для просмотра определённых строк не основного файла так же использовала list с параметрами (рис. 2.13).

```
(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. 2.11: Просмотр исходного кода

```
(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. 2.12: Просмотр строк с 12 по 15

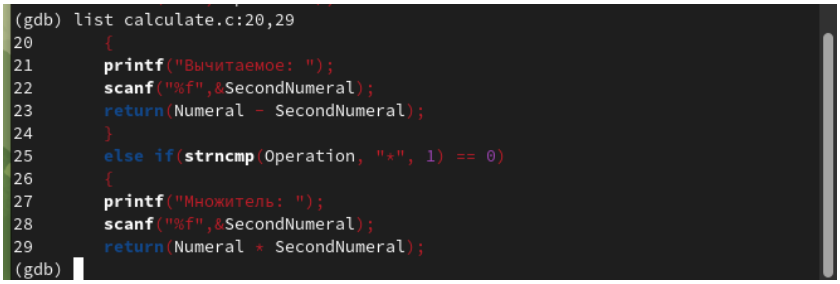

Рис. 2.13: Просмотр определённых строк

Затем установила точку останова в файле calculate.c на строке номер 21 (рис. 2.14), (рис. 2.15). После чего вывела информацию об имеющихся в проекте точек останова (рис. 2.16). Я запустила программу внутри отладчика и убедилась, что программа остановилась в момент прохождения точки останова (рис. 2.17).

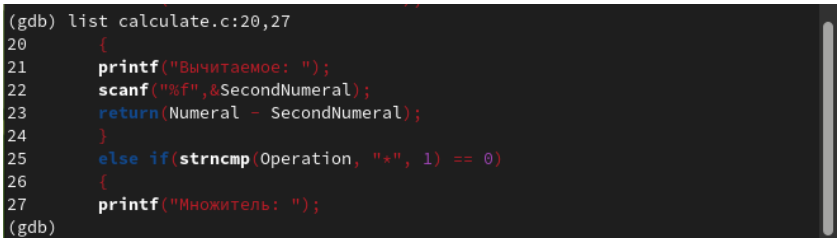

Рис. 2.14: Установка точки останова (1)

Рис. 2.15: Установка точки останова (2)

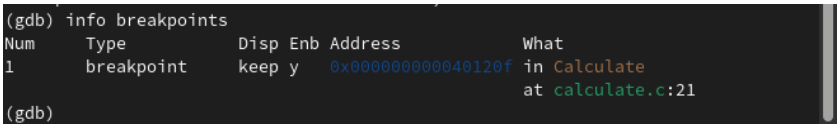

Рис. 2.16: Вывод информации о точках останова

Рис. 2.17: Запуск программы внутри отладчика

Также я посмотрела, чему равно на этом этапе значение переменной Numeral (рис. 2.18). На экран было выведено число 5. Сравнила его с результатом вывода на экран после использования другой команды (рис. 2.19). После чего убрала точки останова (рис. 2.20).

Рис. 2.18: Значение переменной Numeral

Рис. 2.19: Сравнение значений переменной Numeral

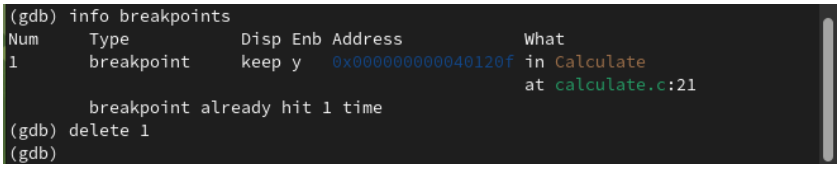

Рис. 2.20: Убираем точки останова

В заключение, с помощью утилиты splint я попробовала проанализировать коды файлов calculate.c (рис. 2.21) и main.c (рис. 2.22).

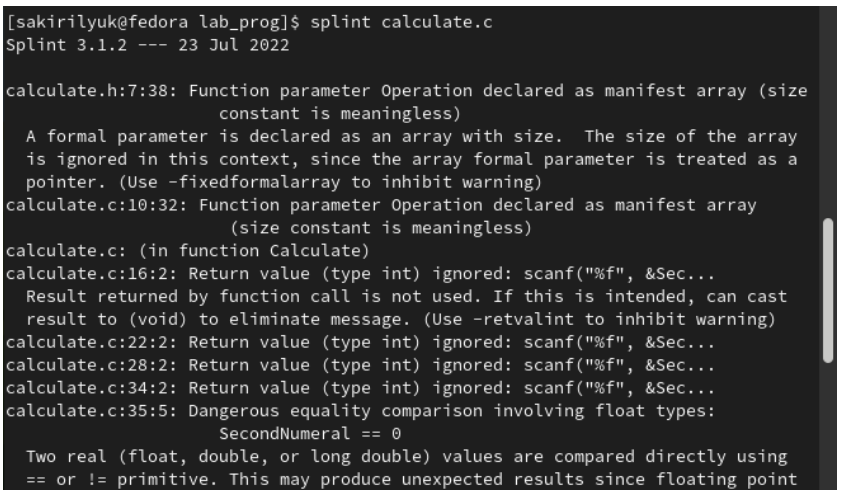

Рис. 2.21: Анализ кода файла calculate.c

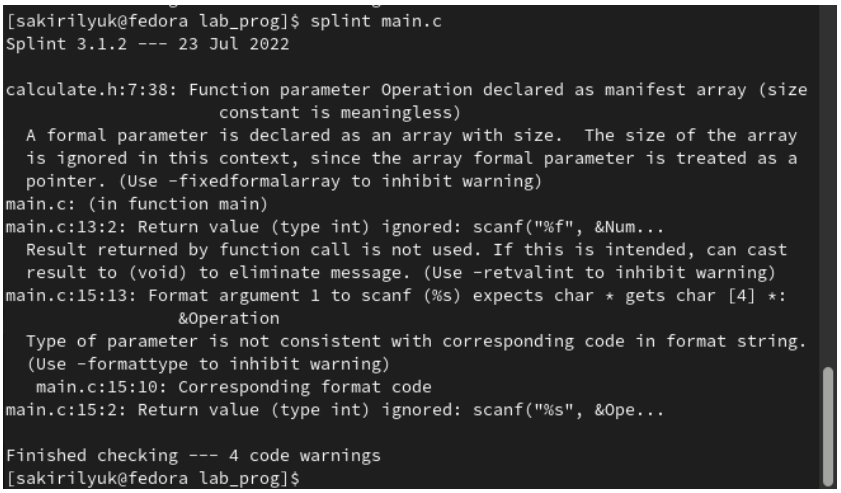

Рис. 2.22: Анализ кода файла main.c

3 Выводы

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