Отчет по лабораторной работе №13.

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

Данила Андреевич Стариков

Содержание

# 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

1. В домашнем каталоге создали подкаталог ~/work/os/lab\_prog, в котором будут храниться файлы с кодом программ.
2. Создали в нём файлы: calculate.h (Листинг 2), calculate.c (Листинг 3), main.c (Листинг 4). Это 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.

Листинг 1: Текст файла calculate.h

#ifndef CALCULATE\_H\_  
#define CALCULATE\_H\_  
  
float Calculate(float Numeral, char Operation[4]);  
  
#endif /\*CALCULATE\_H\_\*/

Листинг 2: Текст файла 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);  
 if(SecondNumeral == 0)  
 {  
 printf("Ошибка: деление на ноль! ");  
 return(HUGE\_VAL);  
 }  
 else return(Numeral / SecondNumeral);  
 }  
 else if(strncmp(Operation, "pow", 3) == 0) {   
 printf("Степень: ");  
 scanf("%f",&SecondNumeral);  
 return(pow(Numeral, SecondNumeral));  
 }  
 else if(strncmp(Operation, "sqrt", 4) == 0) return(sqrt(Numeral));  
 else if(strncmp(Operation, "sin", 3) == 0) return(sin(Numeral));  
 else if(strncmp(Operation, "cos", 3) == 0) return(cos(Numeral));  
 else if(strncmp(Operation, "tan", 3) == 0) return(tan(Numeral));  
 else {  
 printf("Неправильно введено действие ");  
 return(HUGE\_VAL);  
 }  
}

Листинг 3: Текст файла 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;  
}

1. Выполнили компиляцию программы посредством gcc (Рисунок 1) и проверили ее работу (Рисунок 2).

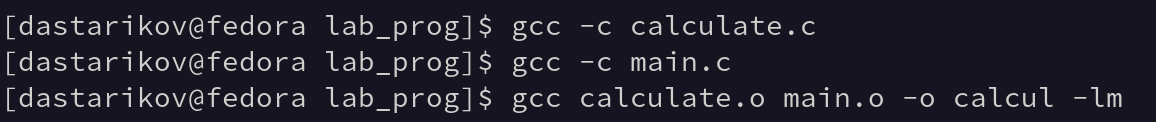


Рис. 1: Компиляция программы.

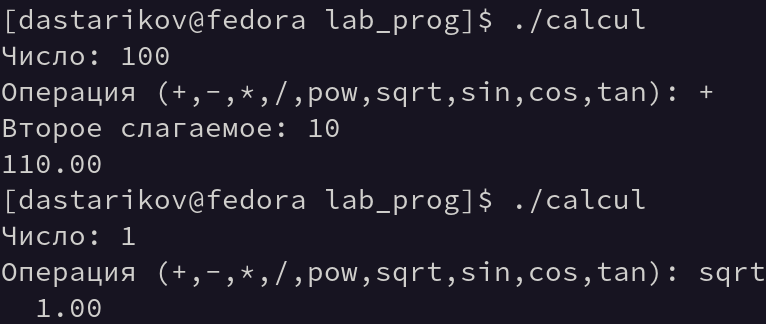


Рис. 2: Работа программы.

1. Для упрощения компиляции создали Makefile (Листинг 4) и запустили его, чтобы убедиться в работе (Рисунок 3)

Листинг 4: Текст Makefile-файла

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 -g -c calculate.c $(CFLAGS)  
  
main.o: main.c calculate.h  
 gcc -g -c main.c $(CFLAGS)  
clean:  
 -rm calcul \*.o \*~

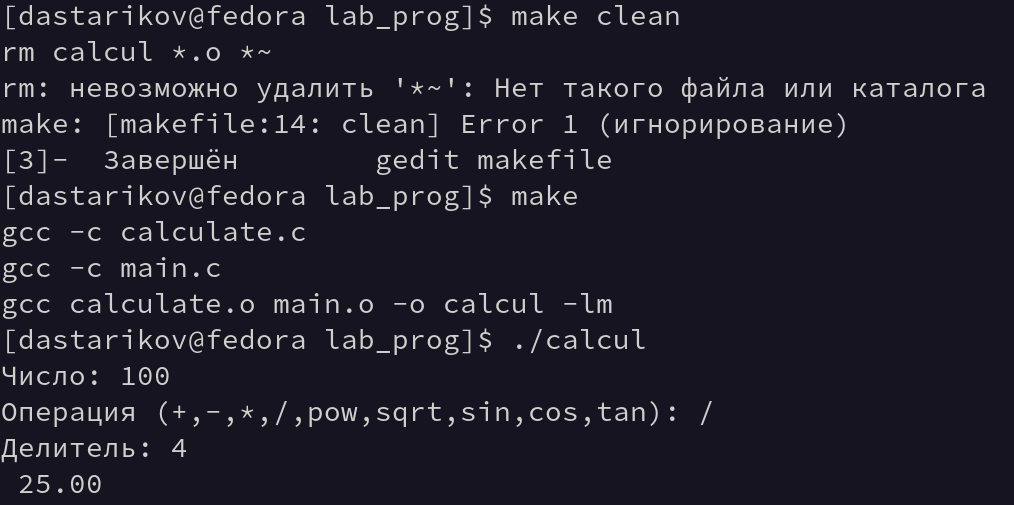


Рис. 3: Проверка работоспособности Makefile.

1. Далее выполнили отладку программы calcul с помощью gdb (Рис. 4, 12):

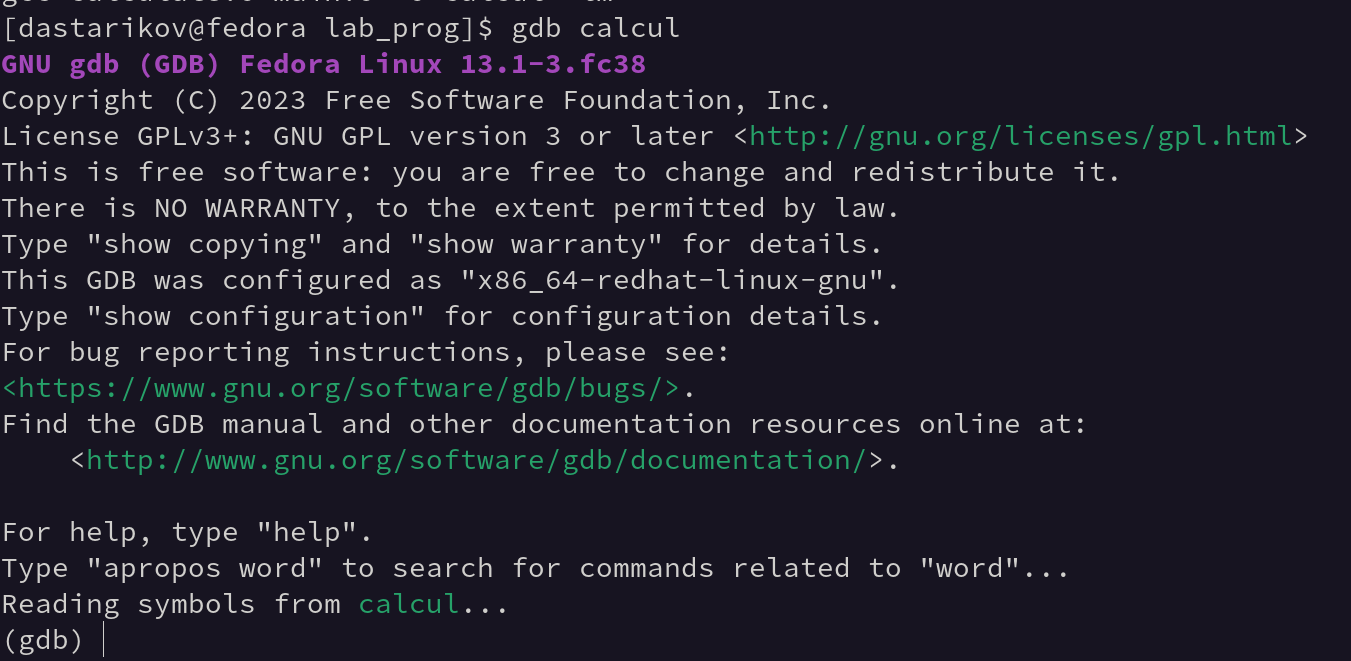


Рис. 4: Запуск отладчика.

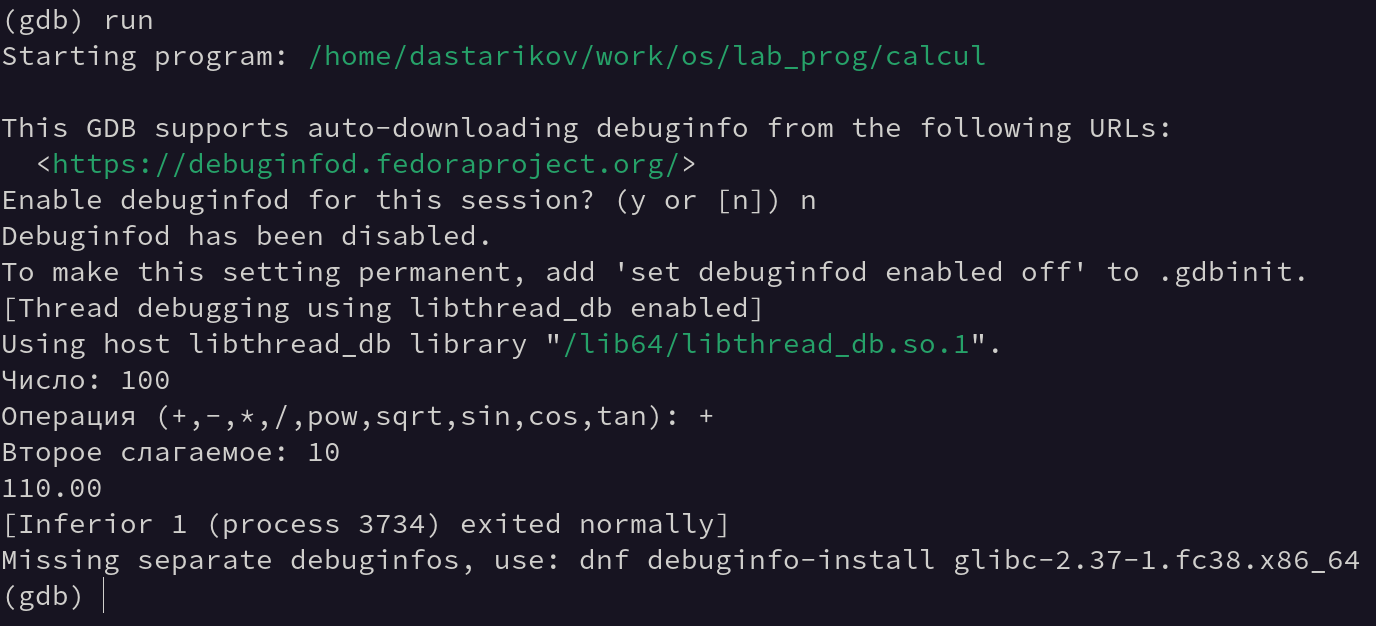


Рис. 5: Запуск программе в среде отладки.

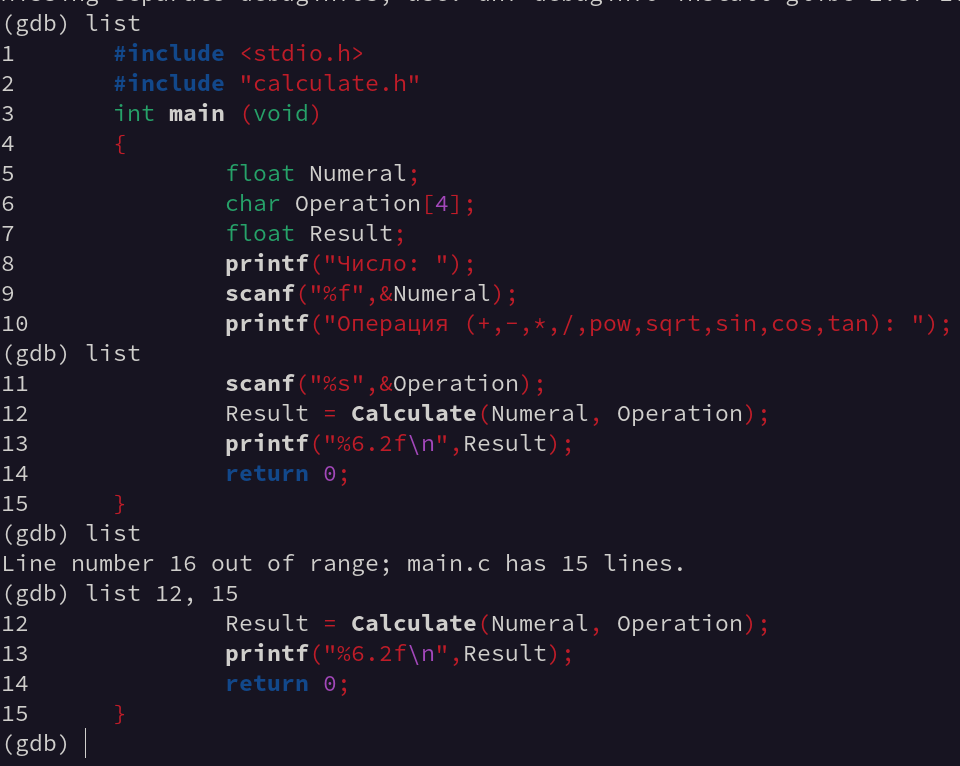


Рис. 6: Просмотр кода основного файла программы.

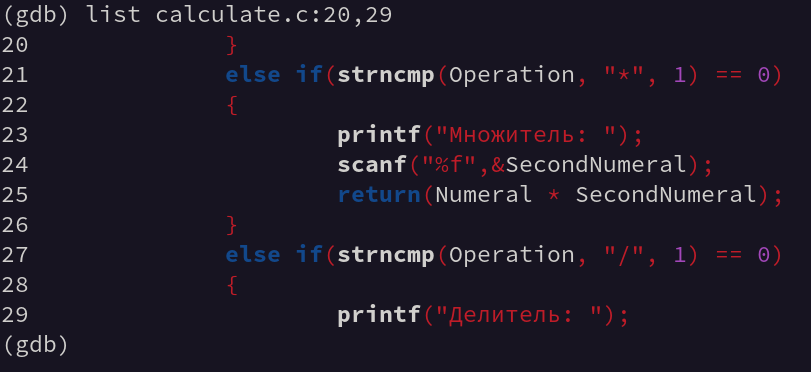


Рис. 7: Просмотр кода файла calculate.c

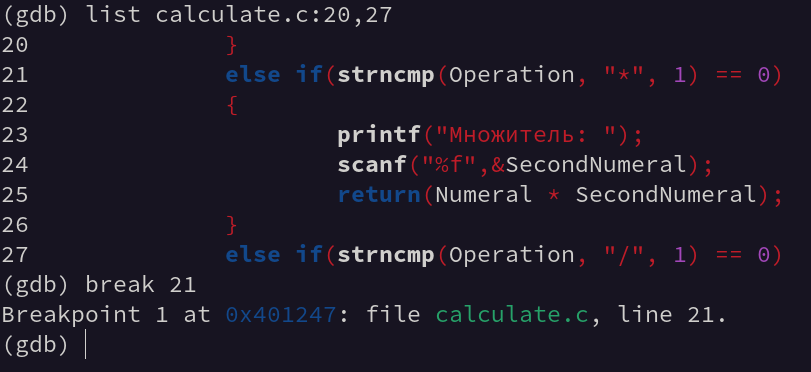


Рис. 8: Просмотр кода файла calculate.c и установка точки останова.

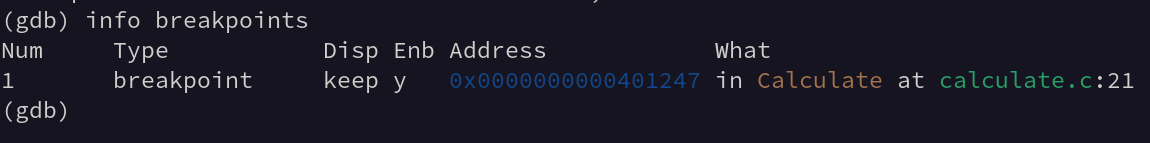


Рис. 9: Просмотр всех точек останова

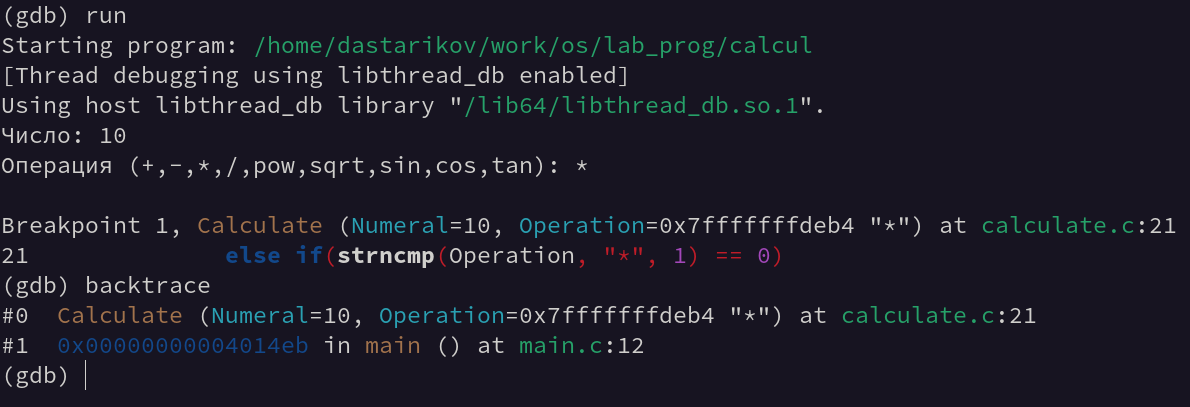


Рис. 10: Запуск программы до точки останова.

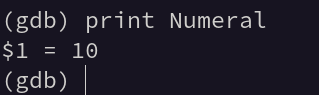


Рис. 11: Просмотр промежуточного значения переменной.

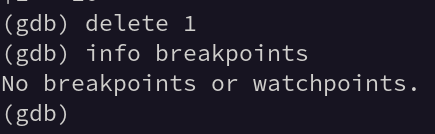


Рис. 12: Удаление точки останова.

1. С помощью утилиты splint проанализировали коды файлов calculate.c и main.c (Рис. 13, 14).

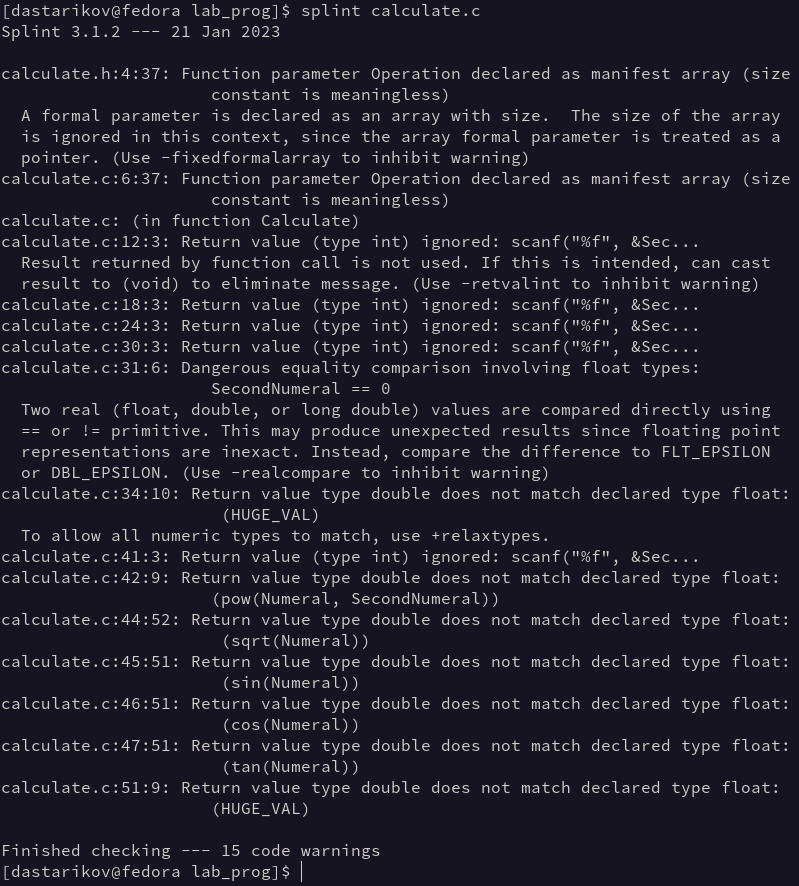


Рис. 13: Анализ calculate.c

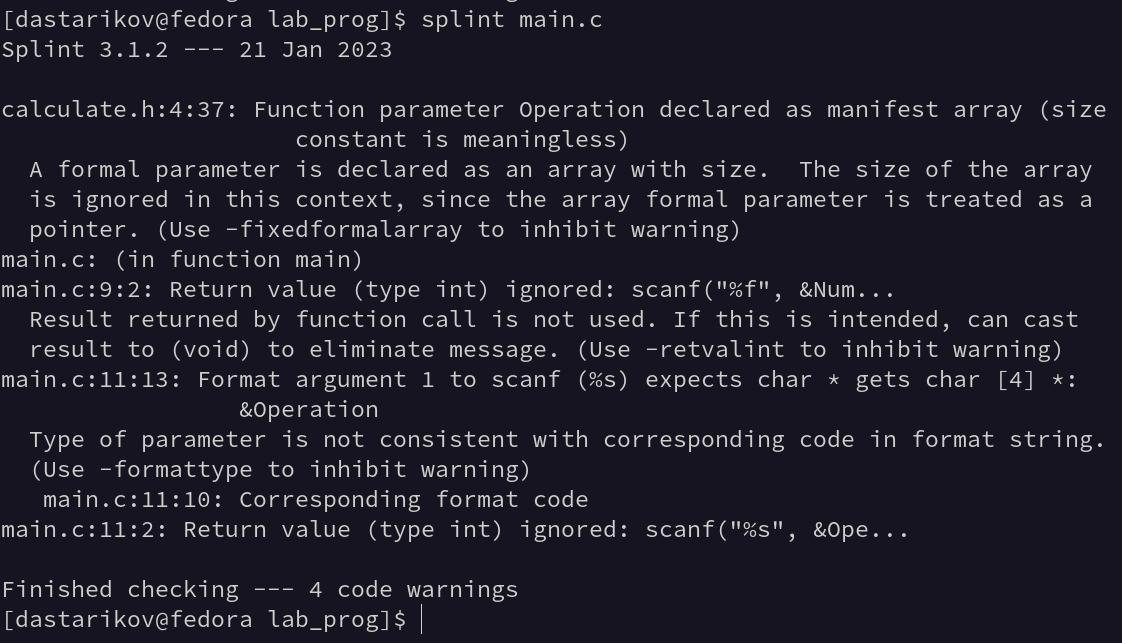


Рис. 14: Анализ main.c.

# 3 Выводы

В рамках лабораторной работы получили практические навыки разработик, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования С калькулятора с простейшими функциями с помощью утилит dgb и splint.