Лабораторная работа №14

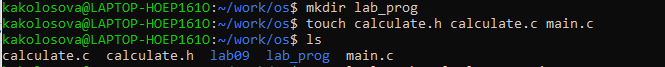
Колосова Кристина Александровна

# 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. 1. Создание подкаталога*

1. Создала 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.

файл 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)

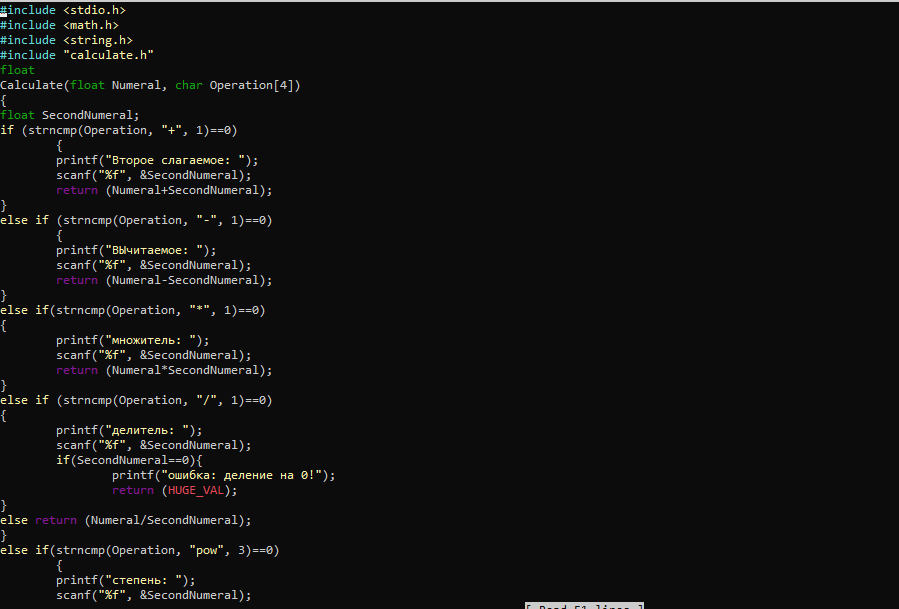
return(sqrt(Numeral));

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); } }



*Рис. 2. файл calculate.c*

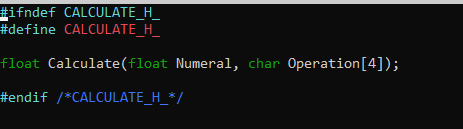
Интерфейсный файл calculate.h, описывающий формат вызова функции калькулятора: calculate.h

#ifndef CALCULATE\_H\_

#define CALCULATE\_H\_

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

#endif /\*CALCULATE\_H\_\*/



*Рис. 3. файл calculate.h*

Основной файл main.c, реализующий интерфейс пользователя к калькулятору: main.c

#include <stdio.h>

#include "calculate.h"

int main (void) {

float Numeral;

char Operation[4];

float Result; printf("Число: ");

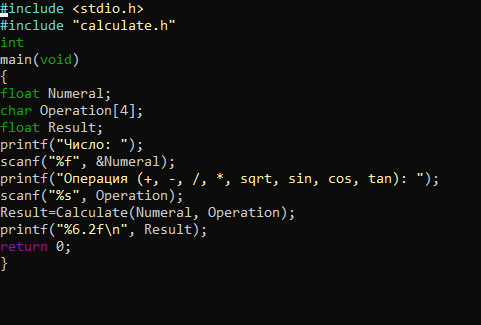
scanf("%f",&Numeral);

printf("Операция (+,-,\*,/,pow,sqrt,sin,cos,tan): ");

scanf("%s",&Operation); Result = Calculate(Numeral, Operation);

printf("%6.2f\n",Result); return 0;

}



*Рис. 4. файл main.c*

1. Выполнила компиляцию программы посредством gcc:

gcc -c calculate.c

gcc -c main.c

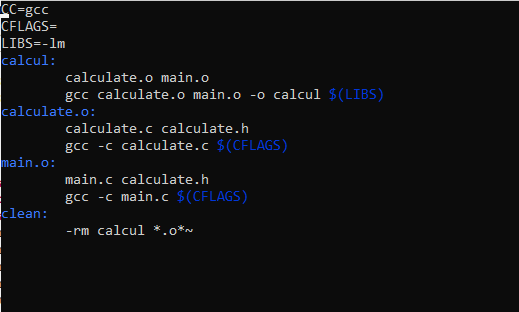
gcc calculate.o main.o -o calcul -lm



*Рис. 5. компиляция*

4. Исправила синтаксические ошибки.

5. Создала Makefile со следующим содержанием:



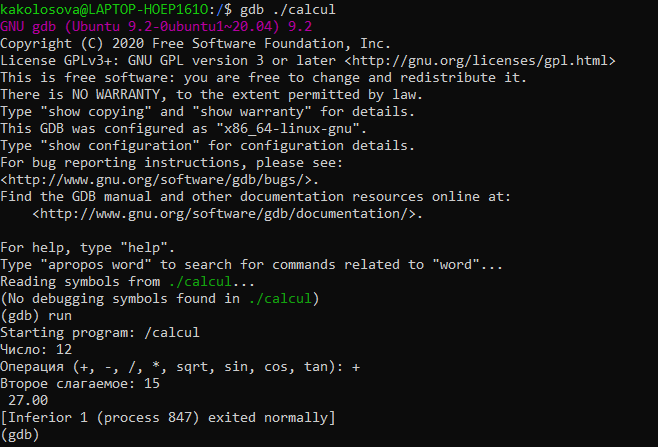
*Рис. 6. Makefile*

Тут прописано какие команды будут выполняться: компиляция и создание файлов.

6. С помощью gdb выполнила отладку программы calcul:

– Запустила отладчик GDB, загрузив в него программу для отладки: gdb ./calcul

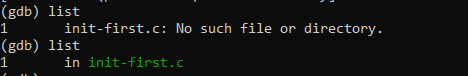
– Для запуска программы внутри отладчика ввела команду run:



*Рис. 7. Загрузка и отладка*

– Для постраничного (по 9 строк) просмотра исходного код используйте команду list: list

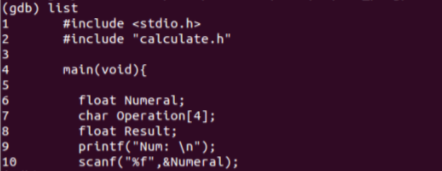
ДЛЯ ВЫПОЛНЕНИЯ ЭТИХ ПУНКТОВ ПРИШЛОСЬ ЗАХОДИТЬ ЧЕРЕЗ ВИРТУАЛЬНУЮ МАШИНУ, ПОЭТОМУ СТИЛЬ СКРИНШОТОВ ОТЛИЧАЕТСЯ! ПРИ ПОПЫТКЕ ВЫПОНИТЬ КОМАНДУ LIST ВСЕГДА ПОЯВЛЯЛОСЬ



*Рис. 8. Просмотр кода когда не получилось*

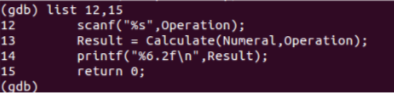
A вот как должно быть:

(С русскими символами выводилось некорректно, исправила на транслит…)



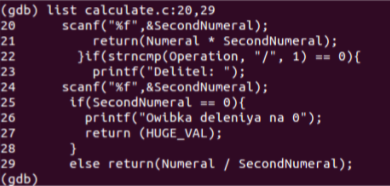
*Рис. 9. 1-9 строки*

– Для просмотра строк с 12 по 15 основного файла используйте list с параметрами: list 12,15



*Рис. 10. 12-15 строки*

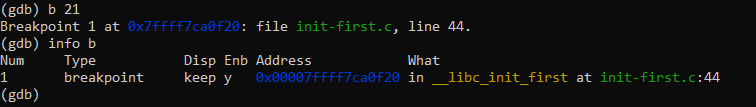
– Для просмотра определённых строк не основного файла используйте list с параметрами: list calculate.c:20,29



*Рис. 11. 20-29 строки*

– Установите точку останова в файле calculate.c на строке номер 21: list calculate.c:20,27 break 21

– Вывела информацию об имеющихся в проекте точка останова: info breakpoints



*Рис. 12. Точка останова*

– Запустила программу внутри отладчика и убедилась, что программа остановится в момент прохождения точки останова: run 5 - backtrace

– Отладчик выдает следующую информацию:

#0 Calculate (Numeral=5, Operation=0x7fffffffd280 "-")

at calculate.c:21

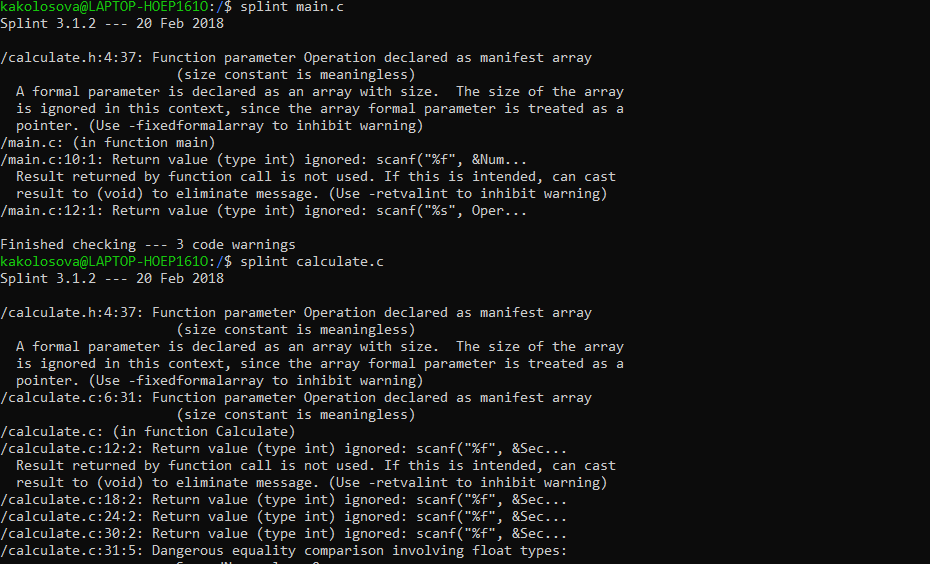
#1 0x0000000000400b2b in main () at main.c:1778

– Уберите точки останова: info breakpoints delete 1



*Рис. 13. Информация о точке останова*

7. С помощью утилиты splint попробовала проанализировать коды файлов calculate.c и main.c.



*Рис. 14. Анализ с помощью* splint

# Выводы

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.

# Библиография

Вот тут посмотрела про init

<https://stackoverflow.com/questions/31379422/why-is-init-from-glibcs-csu-init-first-c-called-before-start-even-if-start-i>

тут про корректную передачу массива символов как аргумент

[Занятие 14. Строки и символьные массивы в языке Си. (youngcoder.net)](http://www.youngcoder.net/2013/08/stroki-simvolnie-massivi.html)