ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ №13

*дисциплина: Операционные системы*

Шилоносов Данил Вячеславович

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 Задачи

1. В домашнем каталоге создайте подкаталог ~/work/os/lab\_prog.
2. Создайте 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. Реализация функций калькулятора в файле 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);  
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);  
}  
}

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\_\*/

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;  
}

1. Выполните компиляцию программы посредством gcc:

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

1. При необходимости исправьте синтаксические ошибки.
2. Создайте Makefile со следующим содержанием:

#  
# 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

Поясните в отчёте его содержание.

1. С помощью gdb выполните отладку программы calcul (перед использованием gdb исправьте Makefile):

* Запустите отладчик GDB, загрузив в него программу для отладки:

gdb ./calcul

– Для запуска программы внутри отладчика введите команду run:

run

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

list

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

list 12,15

* Для просмотра определённых строк не основного файла используйте list с параметрами:

list calculate.c:20,29

* Установите точку останова в файле calculate.c на строке номер 21:

list calculate.c:20,27  
break 21

* Выведите информацию об имеющихся в проекте точка останова:

info breakpoints

* Запустите программу внутри отладчика и убедитесь, что программа остановится в момент прохождения точки останова:

run  
5  
-  
backtrace

* Отладчик выдаст следующую информацию:

#0 Calculate (Numeral=5, Operation=0x7fffffffd280 "-")  
at calculate.c:21  
#1 0x0000000000400b2b in main () at main.c:17

а команда backtrace покажет весь стек вызываемых функций от начала программы до текущего места.

* Посмотрите, чему равно на этом этапе значение переменной Numeral, введя:

print Numeral

На экран должно быть выведено число 5. - Сравните с результатом вывода на экран после использования команды:

display Numeral

* Уберите точки останова

info breakpoints  
delete 1

1. С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c.

# 3 Теоретическое введение

## 3.1 Этапы разработки приложений

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.

## 3.2 Компиляция исходного текста и построение исполняемого файла

Стандартным средством для компиляции программ в ОС типа UNIX является GCC (GNU Compiler Collection). Это набор компиляторов для разного рода языков программирования (С, C++, Java, Фортран и др.). Работа с GCC производится при помощи одноимённой управляющей программы gcc, которая интерпретирует аргументы командной строки, определяет и осуществляет запуск нужного компилятора для входного файла. Файлы с расширением (суффиксом) .c воспринимаются gcc как программы на языке С, файлы с расширением .cc или .C — как файлы на языке C++, а файлы c расширением .o считаются объектными. Для компиляции файла main.c, содержащего написанную на языке С простейшую программу:

/\*  
\* main.c  
\*/  
#include <stdio.h>  
int main()  
{  
printf("Hello World!\n");  
return 0;  
}

достаточно в командной строке ввести:

gcc -c main.c

# 4 Выполнение лабораторной работы

## 4.1 Редактирование, компиляция и линковка исходных файлов

Создадим файлы calculate.c, calculate.h, main.c и Makefile: (рис. [[1](#fig:001)])

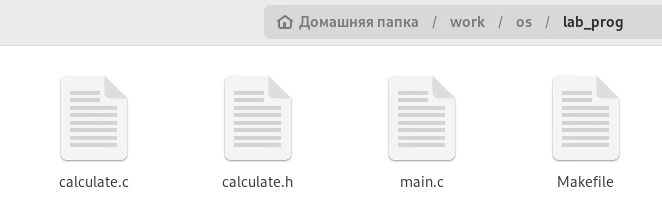


Figure 1: Исходные файлы

Далее, вставим в эти файлы соответствующий код из условия задания.

Исправим описание Makefile (добавим опцию -g для того, чтобы сохранялась отладочная информация): (рис. [[2](#fig:002)])

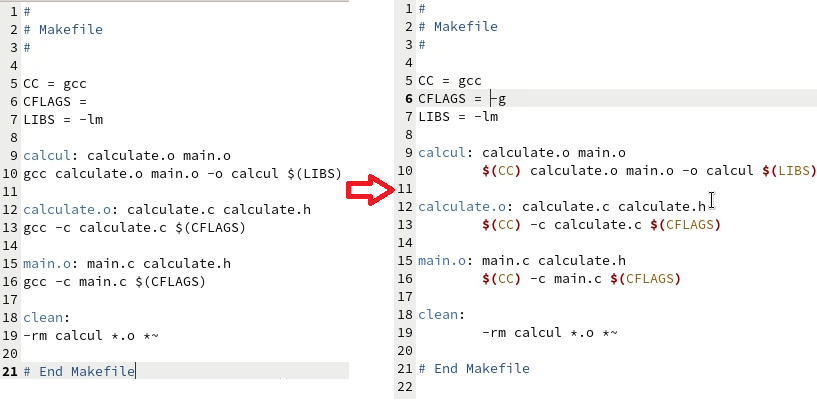


Figure 2: Изменение Makefile

С помощью компилятора GCC скомпилируем объектных файлы, затем слинкуем их в исполняемую программу: (рис. [[3](#fig:003)])

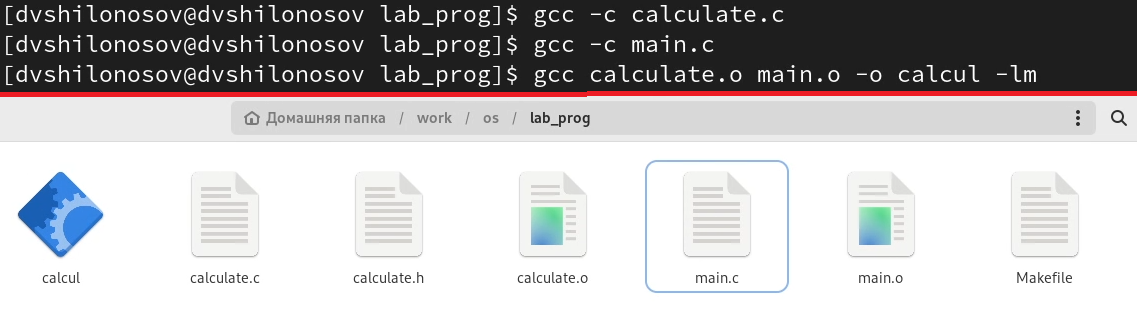


Figure 3: Компиляция и линковка

## 4.2 Работа с отладчиком gdb

Запустим отладчик gdb: (рис. [[4](#fig:004)])

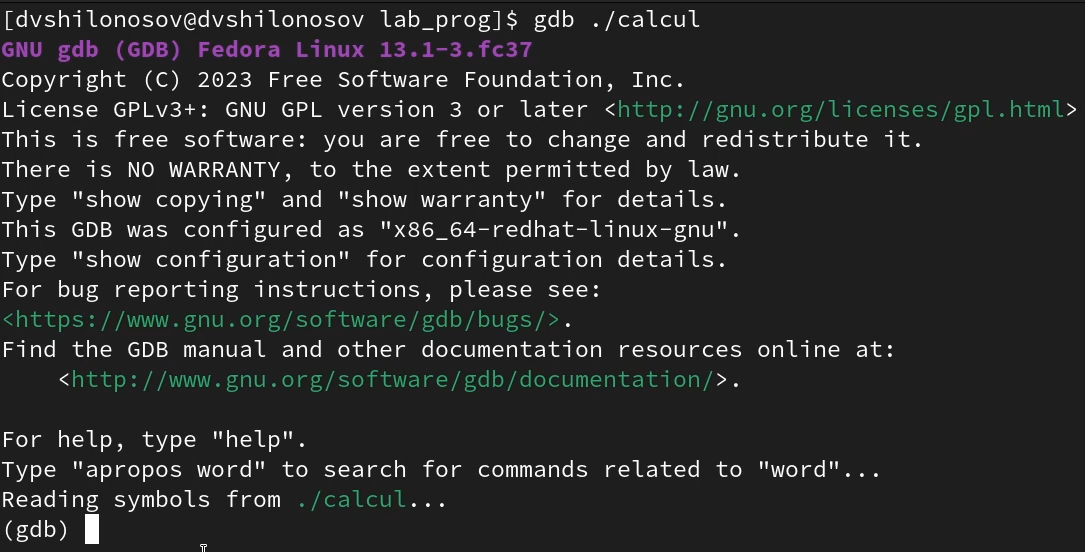


Figure 4: Запуск gdb

В отладчик запустим программу и введем требуемые значения (рис. [[5](#fig:005)])

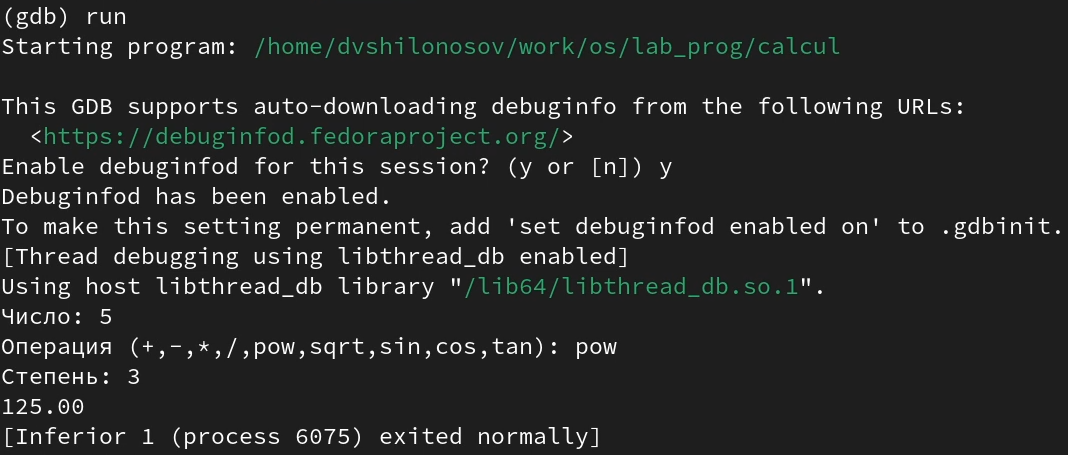


Figure 5: Запуск программы в отладчике gdb

С помощью команды

list

отобразим нужную для просмотра часть кода, а затем, с помощью команды break поставим точку остановы: (рис. [[6](#fig:006)])

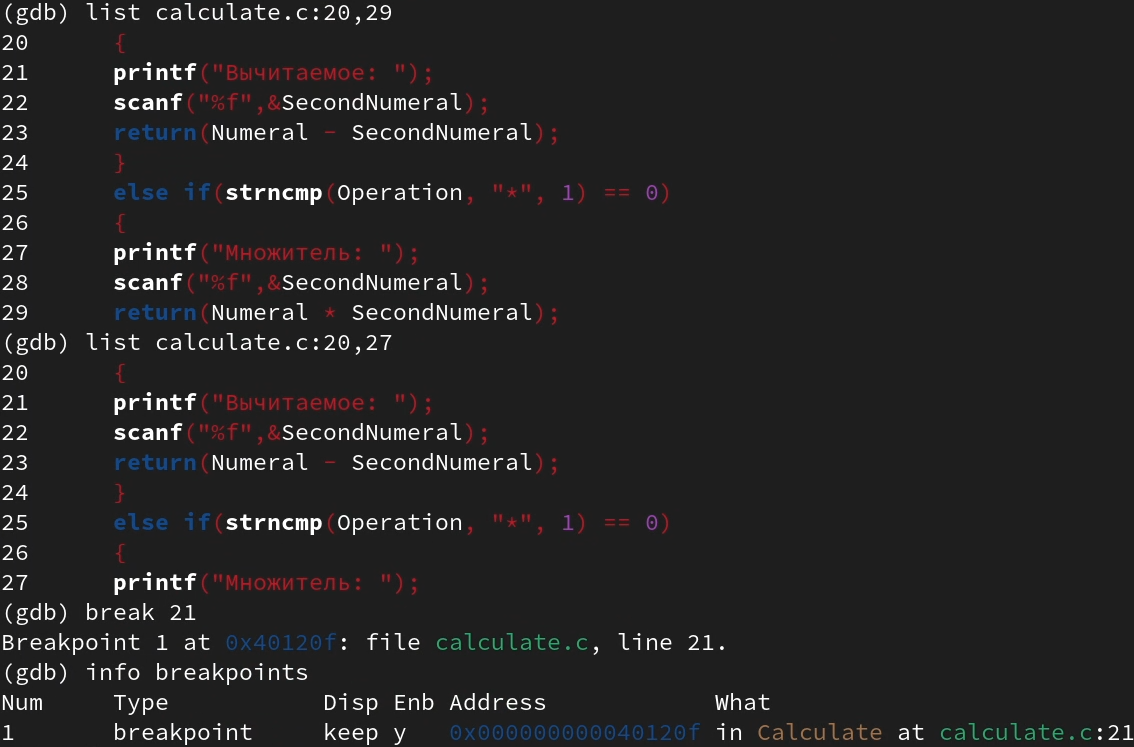


Figure 6: Отображение calculate.c с 20 по 29 строку и установка точки остановы на 21 строке

Вновь запустив программу, видим дополнительную информацию, касающуюся ее выполнения: (рис. [[7](#fig:007)])

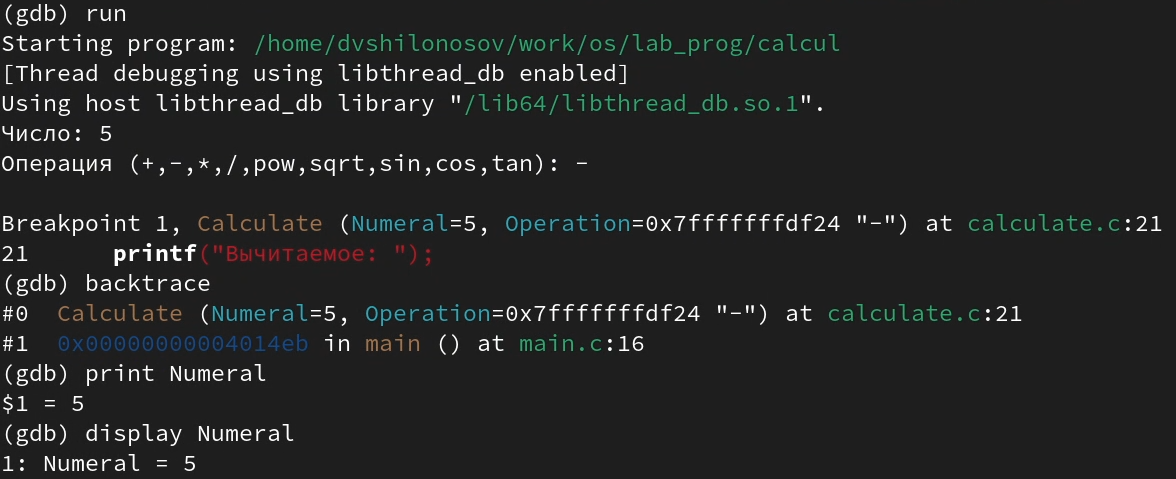


Figure 7: Отображение отладочной информации

## 4.3 Утилита splint

Запустим с помощью утилиты splint файл calculate.c и увидим информацию, которые содержит предупреждения о программе: (рис. [[8](#fig:008)])

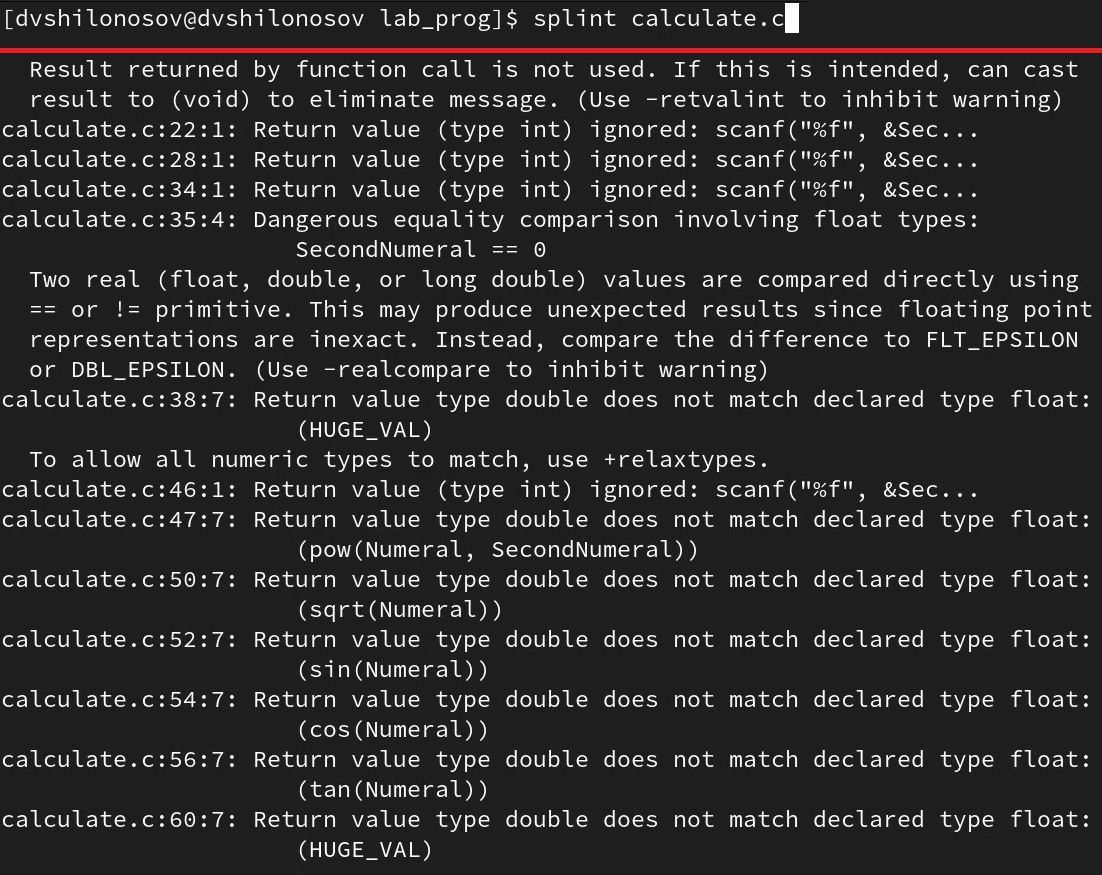


Figure 8: splint calculate.c

# 5 Выводы

В процессе выполнения лабораторной работы были получены простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования С калькулятора с простейшими функциями.