Лабораторная работа №13

Дисцилина: Операционные системы

Маслова Анастасия Сергеевна

Содержание

# 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.

# Задание

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  
1Calculate(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

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 ``bash list

- Для просмотра строк с 12 по 15 основного файла используйте list с параметрами:  
```bash  
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.

# Теоретическое введение

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.

Более подробно - [здесь.](https://esystem.rudn.ru/pluginfile.php/1383191/mod_resource/content/5/013-lab_prog.pdf)

# Выполнение лабораторной работы

1. В домашнем каталоге я создала подкаталог ~/work/os/lab\_prog.
2. Создала в нём файлы: calculate.h, calculate.c, main.c и в них написала код по примеру лабораторной работы (рис.1).

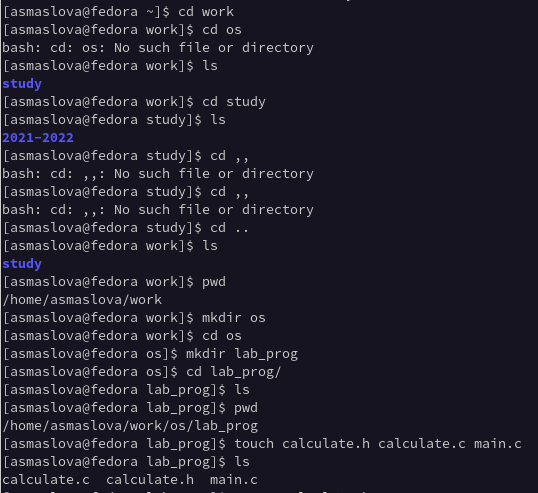


рис.1 Создание подкаталога и необходимых файлов

1. Далее я выполнила компиляцию программы посредством gcc (рис.2).

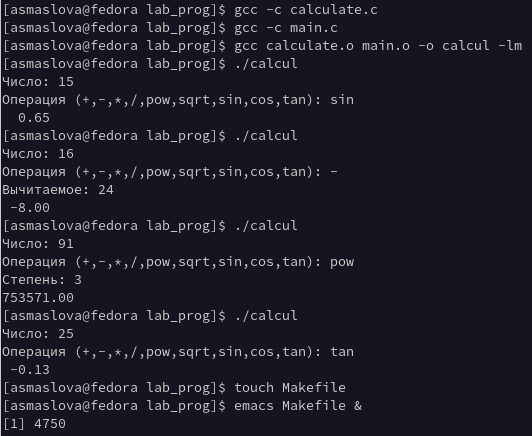
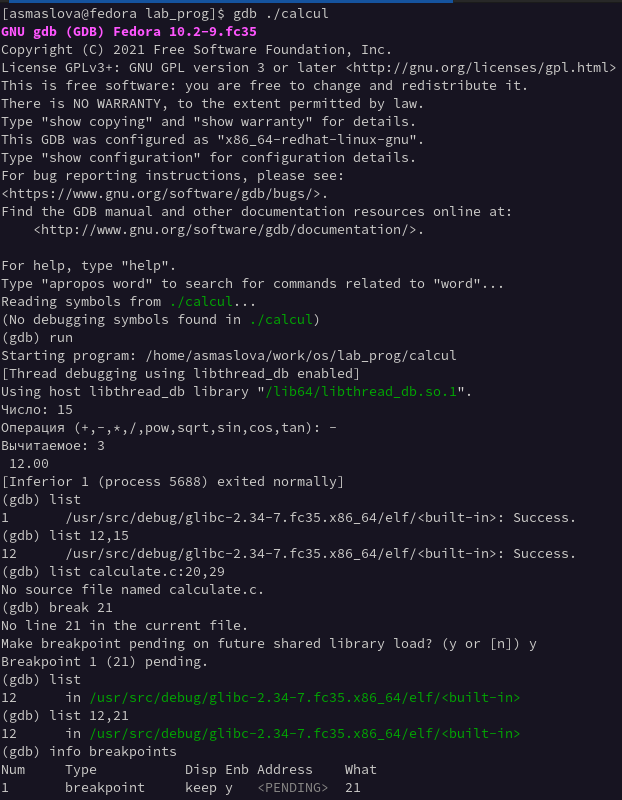
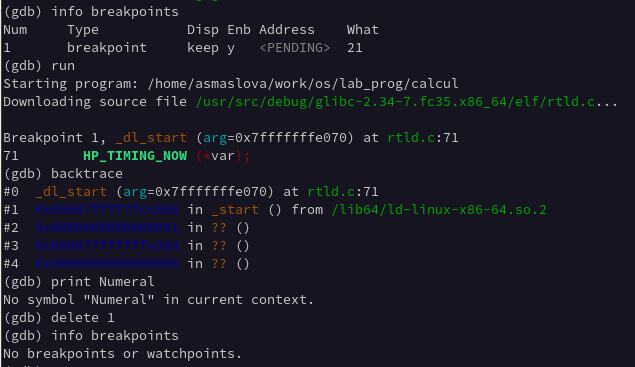


рис.2 Компиляция программы посредством gcc

1. Синтаксических ошибок компилятор мне не выдал, поэтому я ничего не исправляла, вместо этого я проверила работу программы и убедилась, что она работает.
2. После этого я создала Makefile с необходимым содержанием (рис.2).
3. С помощью gdb выполните отладку программы calcul (рис.3-4)

1. С помощью утилиты splint я попробовала проанализировать коды файлов calculate.c и main.c и увидела множество синтаксических ошибок, которые не выдал компилятор (рис.5).

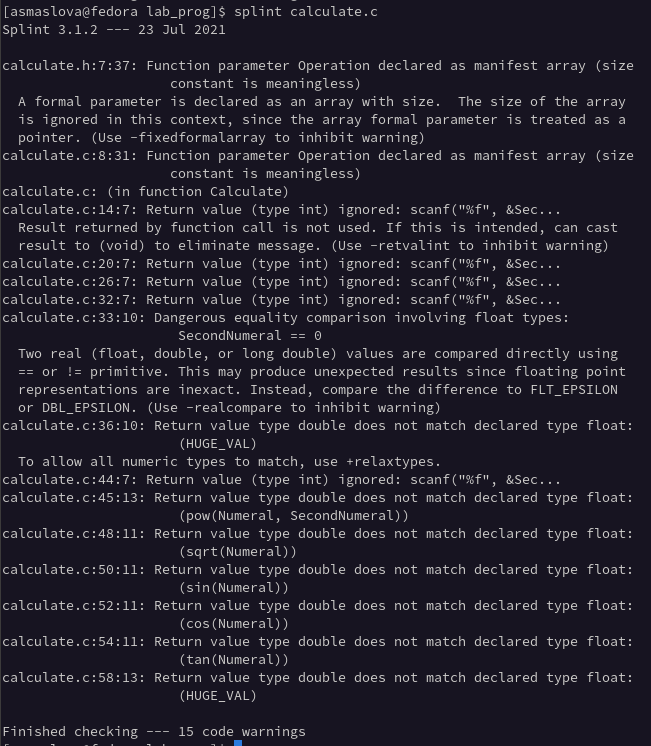


рис.5 Результат работы утилиты splint

# Выводы

В результате лабораторной работы я 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.

# Список литературы