Лабораторная работа №13. Средства, применяемые при разработке программного обеспечения в ОС типа UNIX/Linux.

Парфенова Елизавета Евгеньевна

RUDN University, Moscow, Russian Federation

Цель работы

Приобрести простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования C 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.
3. Выполните компиляцию программы посредством gcc

Задание

4. При необходимости исправьте синтаксические ошибки.
5. Создайте Makefile со следующим содержанием.
6. С помощью gdb выполните отладку программы calcul (перед использованием gdb исправьте Makefile)
7. С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c.

Создание каталогов и файлов

Первым шагом создаем каталог ~/work/os/lab_prog с помощью ***makdir -p*** и файлы calculate.h, calculate.c, main.c. с помощью ***touch***. (рис. 1)

```
[eeparfenova@fedora ~]$ mkdir -p ~/work/os/lab_prog
[eeparfenova@fedora ~]$ cd ~/work/os/lab_prog
[eeparfenova@fedora lab_prog]$ touch calculate.h
[eeparfenova@fedora lab_prog]$ touch calculate.c
[eeparfenova@fedora lab_prog]$ touch main.c
[eeparfenova@fedora lab_prog]$ ls
calculate.c  calculate.h  main.c
```

Figure 1: Создание каталогов и файлов

Компиляция программы

Далее записываем программы в эти файлы. Затем выполняем компиляцию программы посредством gcc, используя команды: *gcc -c calculate.c*, *gcc -c -g main.c*, gcc calculate.o main.o -o calcul -lm* (рис. 2)

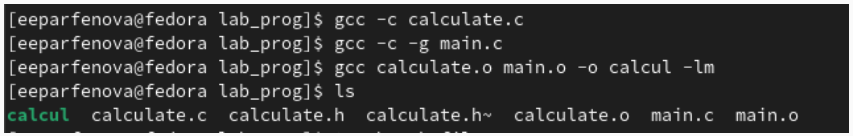

Figure 2: Компиляция программы

Makefile

Далее создаем Makefile с помощью ***touch*** и записываем туда код из файла Лабораторной работы, внеся некторые изменения в 6 и 19 строках. (рис. 3)

```
#
# Makefile
#

CC = gcc
CFLAGS = g
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
```

Figure 3: Makefile

Запуск отладчика

После запускаем отладчик GDB командой *gdb ./calcul*, загрузив в него программу для отладки.(рис. 4)

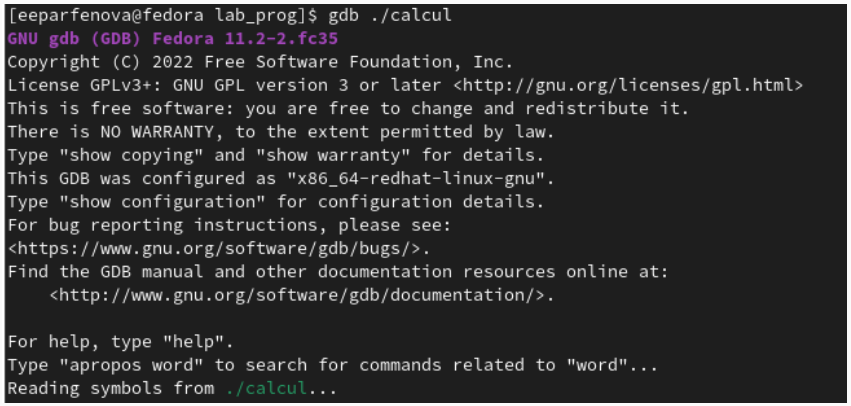

Figure 4: Запуск отладчика

Запуск программы

Запускаем программу внутри отладчика, используя команду ***run***.(рис. 5)

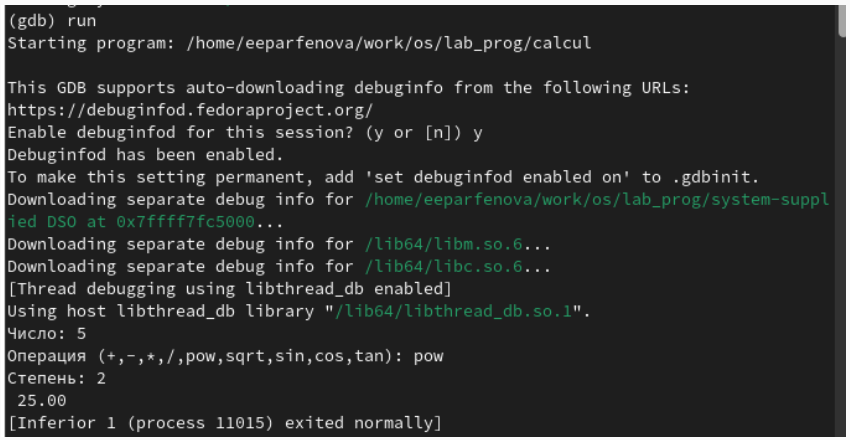

Figure 5: Запуск программы

Просмотр исходного кода

Постранично смотрим код, используя *list*, а после смотрим исходный код с 12 строки по 15 строку командой *list 12,15* (рис. 6)

```
(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) list 12,15
12              printf("Число: ");
13              scanf("%f",&Numeral);
14              printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
15              scanf("%s",&Operation);
```

Figure 6: Просмотр исходного кода

Точки останова

Затем установливаем точку останова в файле calculate.c на строке номер 14, сразу после ввода числа. Делаем это, используя команду ***break 14***. Смотрим информацию об имеющихся в проекте точка останова командой ***info breakpoints***. Запускаем программу и проверяем, когда она остановится. Она остановилась правильно, перед вводом операции. (рис. 7)

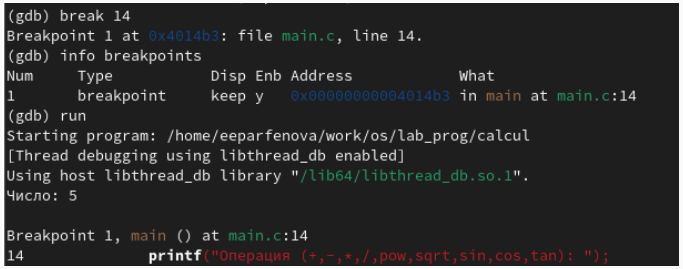

Figure 7: Точки останова

Значение переменной

Смотрим, чему равно на этом этапе значение переменной Numeral, введя ***print Numeral***. Оно равно пяти, так как ввели мы именно это число. Затем сравнваем с результатом вывода на экран, который делаем командой ***display Numeral***. Значения совпали. (рис. 8)

```
(gdb) print Numeral
$1 = 5
(gdb) display Numeral
1: Numeral = 5
```

Figure 8: Значение переменной

Удаление точки останова

После просто убираем точки останова, вызвав вначале информацию о них через ***info breakpoints***, а после удалив, используя ***delete 1*** (точка остановы номер 1) (рис. 9)

```
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x00000000004014b3 in main at main.c:14
        breakpoint already hit 1 time
(gdb) delete 1
```

Figure 9: Удаление точки останова

Утилита splint

Последним шагом с помощью утилиты splint анализируем коды файлов calculate.c и main.c. (рис. 10) (рис. 11)

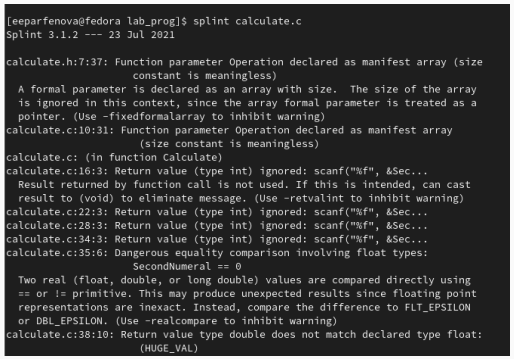

Figure 10: Анализ кода файла calculate.c

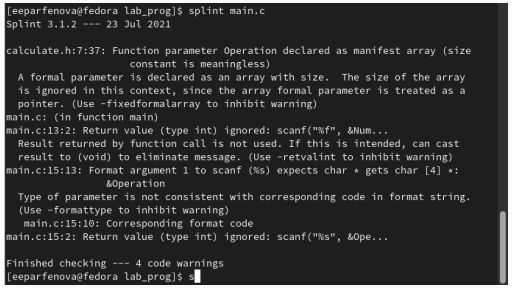

Figure 11: Анализ кода файла main.c

Вывод

Мы приобрели простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования C калькулятора с простейшими функциями.