

# Отчет по лабораторной работе №13

---

Ничипорова Елена

25-05-22

РУДН, Москва

## Отчет

---

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

1. Создаю новый подкаталог ~/work/os/lab\_prog(рис. ??)
- Создала в каталоге файлы: calculate.h, calculate.c, main.c.(рис. 1)

```
ednichi@parova@ok3n57 ~$ mkdir -p ~/work/os/lab_prog  
ednichi@parova@ok3n57 ~$
```

```
ednichi@parova@ok3n57 ~/work/os/lab_prog$ touch calculate.h calculate.c main.c  
ednichi@parova@ok3n57 ~/work/os/lab_prog$ ls  
calculate.c calculate.h main.c
```

Figure 1: Создание файлов

- 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.c`: (рис. 2)(рис. 3)

```
////////////////////////////////////  
// calculate.c  
  
#include<stdio.h>  
#include<math.h>  
#include<string.h>  
#include"calculate.h"  
  
float  
Calculate ( float Numeral, char Operation[4])  
{float SecondNumeral ;  
  if(strcmp(Operation,"+",1) == 0)  
  {printf("Сложить: ");  
   scanf("%f",&SecondNumeral);  
   return(Numeral+ SecondNumeral);  
  }  
  else if(strcmp(Operation,"-",1) == 0)  
  { printf("Вычитать: ");  
   scanf("%f",&SecondNumeral);  
   return(Numeral- SecondNumeral);  
  }  
  else if(strcmp(Operation,"*",1) == 0)  
  {printf("Умножить: ");  
   scanf("%f",&SecondNumeral);  
   return(Numeral* SecondNumeral);  
  }  
  else if(strcmp(Operation,"/",1) == 0)  
  {printf("Делить: ");  
   scanf("%f",&SecondNumeral);  
   if(SecondNumeral== 0)  
   {printf("Ошибка: деление на ноль ");  
    return(HUGE_VAL);  
   }  
   else  
   return(Numeral/ SecondNumeral);  
  }  
  else if(strcmp(Operation,"pow",3) == 0)  
  {printf("Степень: ");  
   scanf("%f",&SecondNumeral);  
   return(pow(Numeral,SecondNumeral));  
  }  
  else if(strcmp(Operation,"sqrt",4) == 0)  
  {printf("Квадратный корень: ");  
   scanf("%f",&SecondNumeral);  
   return(sqrt(SecondNumeral));  
  }  
  else  
  {printf("Неизвестная операция ");  
   return(0);  
  }  
}
```

- Интерфейсный файл `calculate.h`, описывающий формат вызова функции-калькулятора:(рис. 4)

```
////////////////////////////////////  
// calculate.h  
  
#ifndef CALCULATE_H_  
#define CALCULATE_H_  
  
float Calculate(float Numeral, char Operation[4]);  
  
#endif /*CALCULATE_H_*/
```

Figure 4: текст программы

- Основной файл main.c, реализующий интерфейс пользователя к калькулятору: (рис. 5)

```
////////////////////////////////////  
// 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;  
}
```

Figure 5: текст программы

- выполнила компиляцию программы посредством gcc(рис. 6). Обнаружила предупреждение в файле main.c и исправила программу(рис. 7). Заново скомпилировала, после этого ошибок не возникло.(рис. 8)



Figure 6: компиляция программы



Figure 7: исправленный текст программы



Figure 8: компиляция программы



- Создала Makefile с необходимым содержанием и исправила его(рис. 9) Данный файл необходим для автоматической компиляции файлов calculate.c (цель calculate.o), main.c (цель main.o), а также их объединения в один исполняемый файл calcul (цель calcul). Цель clean нужна для автоматического удаления файлов. Переменная CC отвечает за утилиту для компиляции. Переменная CFLAGS отвечает за опции в данной утилите. Переменная LIBS отвечает за опции для объединения объектных файлов в один исполняемый файл.

```
#!/  
## Makefile  
#!/  
CC = gcc  
CFLAGS = -g  
LIBS = -lm  
calcul: calculate.o main.o  
    $(CC) calculate.o main.o -o calcul $(LIBS)  
calculate.o: calculate.c calculate.h  
    $(CC) -c calculate.c $(CFLAGS)  
main.o: main.c calculate.h  
    $(CC) -c main.c $(CFLAGS)  
clean:  
    -rm calcul *.o *~  
## End Makefile
```

Figure 9: текст программы Makefile

- Удалила исполняемые и объектные файлы из каталога(рис. 10).  
Выполнила компиляцию файлов (рис. 11)

```
ednichiporova@k3n57 ~/work/os/lab_prog $ make clean  
rm calcul *.o *
```

Figure 10: удаление файлов

```
ednichiporova@k3n57 ~/work/os/lab_prog $ make calculate.o  
gcc -c calculate.c -o  
ednichiporova@k3n57 ~/work/os/lab_prog $ make main.o  
gcc -c main.c -o  
ednichiporova@k3n57 ~/work/os/lab_prog $ make calcul  
gcc calculate.o main.o -o calcul -lm
```

Figure 11: компиляция файлов

- С помощью gdb выполнила отладку программы calcul. Запустила отладчик GDB(рис. 12)

```
dmichiporova@hnd7 ~/work/os/lab_prog $ gdb ./calcul
GNU gdb (Gentoo 10.2 vanilla) 10.2
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./calcul...
(gdb)
```

Figure 12: Работа с gdb

- Запустила программу внутри отладчика(рис. ??). Постранично просмотрела исходный код с помощью команды “list”(рис. 13)



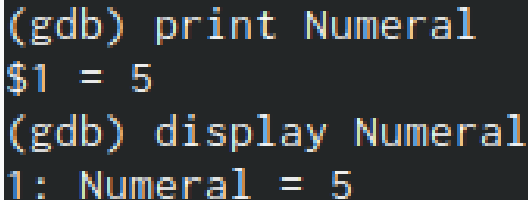
- Поставила точку останова в файле calculate.c на 21 строке(рис. ??).Вывела информацию о имеющихся точках останова (рис. 15)

```
(gdb) break 21  
Breakpoint 1 at 0x55555555527d: file calculate.c, line 22.
```

```
(gdb) info breakpoint  
Num      Type             Disp Enb Address              What  
1        breakpoint      keep y   0x000055555555527d in calculate at calculate.c:22
```

Figure 15: info breakpoint

- Запустила программу внутри отладчика и убедилась, что программа остановилась в момент прохождения точки останова. Посмотрела, чему равно на этом этапе значение переменной Numeral, сравнила с результатом вывода на экран после использования команды “display Numeral”. Значения совпадают(рис. 16)

A screenshot of a GDB terminal window with a black background and yellow text. The text shows the execution of two commands: 'print Numeral' and 'display Numeral', both resulting in the value 5.

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

Figure 16: проверка работы программы

- Убрала точки останова(рис. 17)

```
(gdb) info breakpoint
Num      Type           Disp Enb Address            What
1        breakpoint     keep y   0x000055555555527d in calculate at calculate.c:22
(gdb) delete 1
(gdb) info breakpoint
Undefined info command: "breakpoint". Try "help info".
```

Figure 17: delet 1



- С помощью утилиты splint проанализируем коды файлов `calculate.c` и `main.c`. Воспользуемся командами `splint calculate.c` и `splint main.c`. С помощью утилиты splint выяснилось, что в файлах `calculate.c` и `main.c` присутствует функция чтения `scanf`, возвращающая целое число (тип `int`), но эти числа не используются и нигде не сохраняются. Утилита вывела предупреждение о том, что в файле `calculate.c` происходит сравнение вещественного числа с нулем. Также возвращаемые значения (тип `double`) в функциях `pow`, `sqrt`, `sin`, `cos` и `tan` записываются в переменную типа `float`, что свидетельствует о потере данных.

Приобрела простейшие навыки разработки, анализа, тестирования и отладки при-ложений в ОС типа UNIX/Linux на примере создания на языке программирования C калькулятора с простейшими функциями.