

Отчёт по лабораторной работе №13

Средства для создания приложений в ОС UNIX

Леденев Егор Олегович

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Вывод	13

Список иллюстраций

2.1	Компиляция	8
2.2	Использование make	10
2.3	Использование отладчика	10
2.4	Использование отладчика	11
2.5	Использование отладчика	11
2.6	Использование splint	12

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 Выполнение лабораторной работы

1. Создали подкаталог для файлов лаб работы
2. Создал 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.

Код файла calculate.c (реализует функции калькулятора)

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

```

Код файла calculate.h (описывает формат вызова функции калькулятора)

```

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

```

Код файла main.c (реализует интерфейс пользователя к калькулятору)

```

////////////////////////////////////

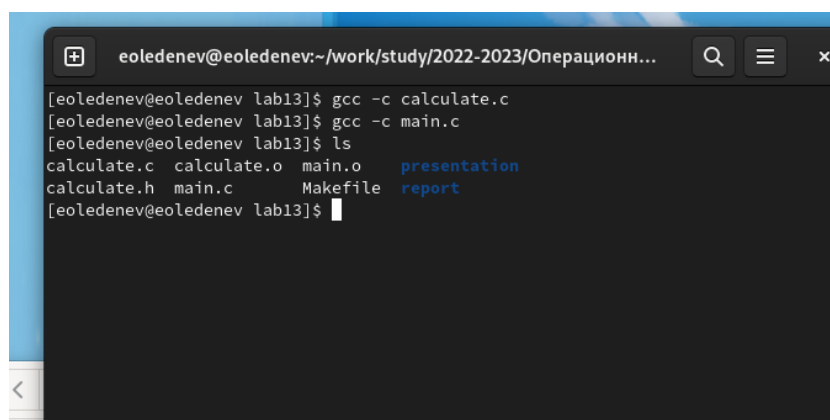
```

```
// 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("%.2f\n",Result);
    return 0;
}
```

3. Выполнили компиляцию программы посредством gcc :



```
eoledenev@eoledenev:~/work/study/2022-2023/Операционн...
[eoledenev@eoledenev lab13]$ gcc -c calculate.c
[eoledenev@eoledenev lab13]$ gcc -c main.c
[eoledenev@eoledenev lab13]$ ls
calculate.c  calculate.o  main.o      presentation
calculate.h  main.c      Makefile    report
[eoledenev@eoledenev lab13]$
```

Рис. 2.1: Компиляция

4. При необходимости исправили синтаксические ошибки.

5. Создали Makefile со следующим содержанием:

```
#  
# 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
```

С помощью программы make получаем различные варианты построения исполняемого модуля.

```
eoledenev@eoledenev:~/work/study/2022-2023/Операционн...
[eoledenev@eoledenev lab13]$ gcc -c calculate.c
[eoledenev@eoledenev lab13]$ gcc -c main.c
[eoledenev@eoledenev lab13]$ ls
calculate.c  calculate.o  main.o      presentation
calculate.h  main.c       Makefile    report
[eoledenev@eoledenev lab13]$ make clean
rm calcul *.o *~
rm: невозможно удалить 'calcul': Нет такого файла или каталога
rm: невозможно удалить '*~': Нет такого файла или каталога
make: [Makefile:14: clean] Ошибка 1 (игнорирование)
[eoledenev@eoledenev lab13]$ ls
calculate.c  calculate.h  main.c  Makefile  presentation  report
[eoledenev@eoledenev lab13]$ make calcul
gcc -c calculate.c -g
gcc -c main.c -g
gcc calculate.o main.o -o calcul -lm
[eoledenev@eoledenev lab13]$ ls
calcul      calculate.h  main.c  Makefile    report
calculate.c  calculate.o  main.o  presentation
[eoledenev@eoledenev lab13]$
```

Рис. 2.2: Использование make

4. С помощью gdb выполнил отладку программы calcul

```
eoledenev@eoledenev:~/work/study/2022-2023/Операционн...
<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) r
Starting program: /home/eoledenev/work/study/2022-2023/Операционные системы/os-интродукция/лабы/lab13/calcul

This GDB supports auto-downloading debuginfo from the following URLs:
https://debuginfod.fedoraproject.org/
Enable debuginfod for this session? (y or [n])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 1
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): +
Второе слагаемое: 3
4.00
[Inferior 1 (process 5897) exited normally]
Missing separate debuginfos, use: dnf debuginfo-install glibc-2.35-20.fc36.x86_64
(gdb)
```

Рис. 2.3: Использование отладчика

```
eoledenev@eoledenev:~/work/study/2022-2023/Операционн...
20 {
21     printf("Вычитаемое: ");
22     scanf("%f",&SecondNumeral);
23     return(Numeral - SecondNumeral);
24 }
25 else if(strncmp(Operation, "+", 1) == 0)
26 {
27     printf("Множитель: ");
28     scanf("%f",&SecondNumeral);
29     return(Numeral * SecondNumeral);
(gdb) break 21
Breakpoint 1 at 0x40120f: file calculate.c, line 21.
(gdb) run
Starting program: /home/eoledenev/work/study/2022-2023/Операционные системы/os-интродукция/labs/lab13/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -
Breakpoint 1, Calculate (Numeral=5, Operation=0x7fffffffdee4 "-") at calculate.c:
:21
21     printf("Вычитаемое: ");
(gdb)
```

Рис. 2.4: Использование отладчика

```
eoledenev@eoledenev:~/work/study/2022-2023/Операционн...
Breakpoint 1, Calculate (Numeral=5, Operation=0x7fffffffdee4 "-") at calculate.c:
:21
21     printf("Вычитаемое: ");
(gdb) backtrace
#0 Calculate (Numeral=5, Operation=0x7fffffffdee4 "-") at calculate.c:21
#1 0x00000000004014eb in main () at main.c:17
(gdb) print Numeral
$1 = 5
(gdb) display Numeral
1: Numeral = 5
(gdb) i b
Num Type Disp Enb Address What
1 breakpoint keep y 0x000000000040120f in Calculate at calculate.c:21
breakpoint already hit 1 time
(gdb) dele 1
(gdb) q
A debugging session is active.

Inferior 1 [process 5918] will be killed.

Quit anyway? (y or n) y
[eoledenev@eoledenev lab13]$
```

Рис. 2.5: Использование отладчика

5. С помощью утилиты splint попробовали проанализировать коды файлов

3 Вывод

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