

Презентация по лабораторной работе №13

Средства, применяемые при разработке программного обеспечения в ОС типа UNIX/Linux

Сагдеров Камал

05.05.2023

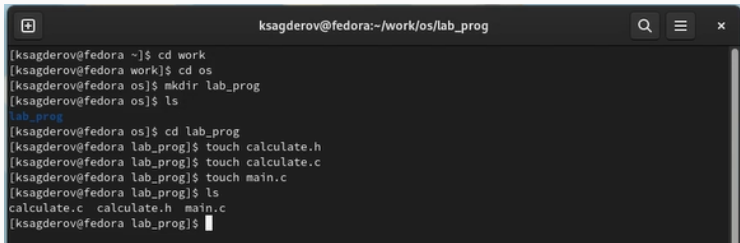
Российский университет дружбы народов, Москва, Россия

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.

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`:
3. Выполните компиляцию программы посредством `gcc`:

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): – Запустите отладчик GDB, загрузив в него программу для отладки
7. С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c

1. В домашнем каталоге создайте подкаталог `~/work/os/lab_prog`.
2. Создайте в нём файлы: `calculate.h`, `calculate.c`, `main.c`

A screenshot of a terminal window with a dark background. The window title is "ksagderov@fedora:~/work/os/lab_prog". The terminal shows a sequence of commands and their outputs:

```
[ksagderov@fedora ~]$ cd work
[ksagderov@fedora work]$ cd os
[ksagderov@fedora os]$ mkdir lab_prog
[ksagderov@fedora os]$ ls
lab_prog
[ksagderov@fedora os]$ cd lab_prog
[ksagderov@fedora lab_prog]$ touch calculate.h
[ksagderov@fedora lab_prog]$ touch calculate.c
[ksagderov@fedora lab_prog]$ touch main.c
[ksagderov@fedora lab_prog]$ ls
calculate.c calculate.h main.c
[ksagderov@fedora lab_prog]$
```

Рис. 1: Терминал

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`:

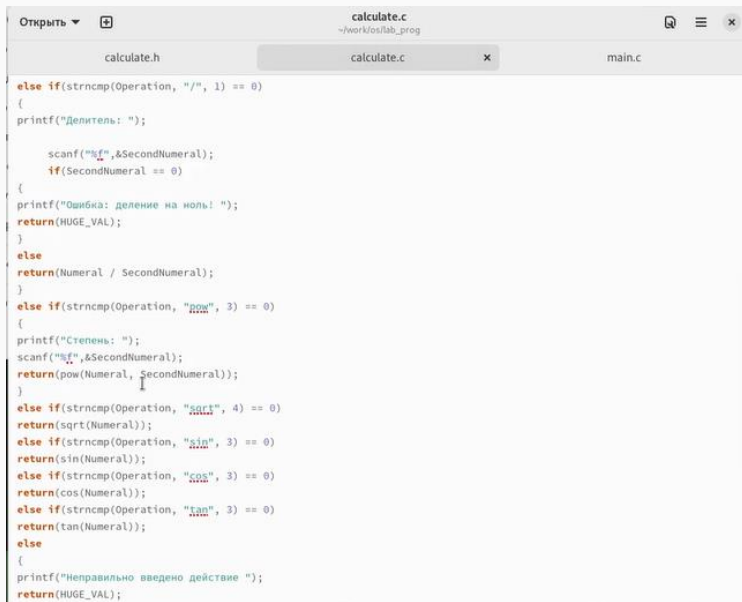


The image shows a code editor window with the title bar 'calculate.h' and a path '~\work\os\lab_prog'. The editor has three tabs: 'calculate.h' (active), 'calculate.c', and 'main.c'. The code in the active tab is as follows:

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

Рис. 2: Текст программы

Процесс выполнения

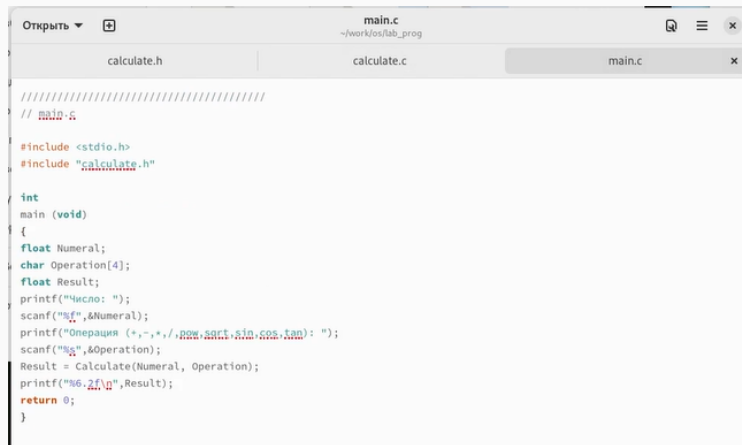


```
calculate.c
~/work/os/lab_prog

calculate.h calculate.c x main.c

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);
}
```

```
main.c
~/work/os/lab_prog

calculate.h | calculate.c | main.c x

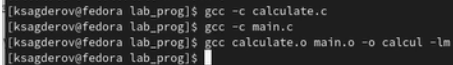
////////////////////////////////////
// main.c

#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;
}
```

Рис. 4: Текст программы

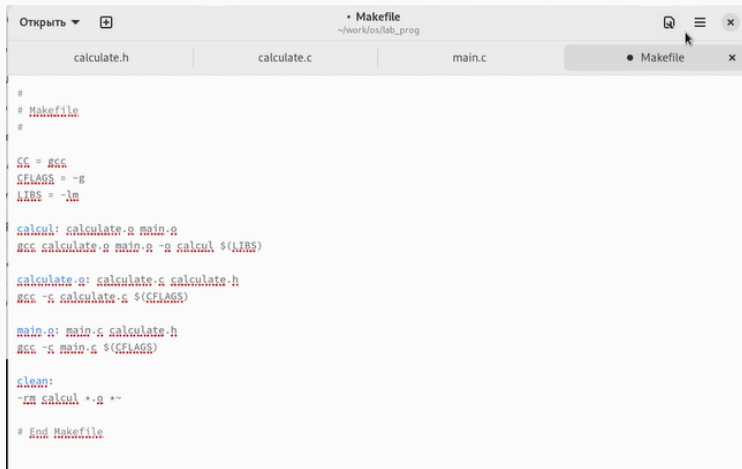
3. Выполните компиляцию программы посредством gcc:

A terminal window with a dark background and light-colored text. It shows four lines of commands entered at a prompt. The first two lines compile individual source files. The third line links the object files into an executable named 'calcul' with the '-lm' flag. The fourth line shows the prompt with a cursor, indicating the process is complete.

```
[ksagderov@fedora lab_prog]$ gcc -c calculate.c  
[ksagderov@fedora lab_prog]$ gcc -c main.c  
[ksagderov@fedora lab_prog]$ gcc calculate.o main.o -o calcul -lm  
[ksagderov@fedora lab_prog]$
```

Рис. 5: Компиляция программ

4. При необходимости исправьте синтаксические ошибки.
5. Создайте 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
```

6. С помощью gdb выполните отладку программы calcul (перед использованием gdb исправьте Makefile): – Запустите отладчик GDB, загрузив в него программу для отладки: – Для запуска программы внутри отладчика введите команду run


```
(gdb) run
Starting program: /home/ksagderov/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 470
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): *
Множитель: 50
23500.00
[Inferior 1 (process 3858) exited normally]
(gdb) 
```

Рис. 7: GDB

Процесс выполнения

- Для постраничного (по 9 строк) просмотра исходного код используйте команду list: – Для просмотра строк с 12 по 15 основного файла используйте list с параметрами:

```
Число: 90
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -
Вычитаемое: 67
23.00
[Inferior 1 (process 2732) exited normally]
(gdb) list
1  //////////////////////////////////////
2  // main.c
3
4  #include <stdio.h>
5  #include "calculate.h"
6
7  int
8  main (void)
9  {
10 float Numeral;
(gdb) list
11 char Operation[4];
12 float Result;
13 printf("Число: ");
14 scanf("%f",&Numeral);
15 printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
16 scanf("%s",&Operation);
17 Result = Calculate(Numeral, Operation);
18 printf("%.2f\n",Result);
19 return 0;
20 }
```

– Для просмотра определённых строк не основного файла используйте list с параметрами: – Установите точку останова в файле calculate.c на строке номер 21: – Выведите информацию об имеющихся в проекте точка останова: – Запустите программу внутри отладчика и убедитесь, что программа остановится в момент прохождения точки останова: – Отладчик выдаст следующую информацию – Сравните с результатом вывода на экран после использования команды - Уберите точки останова:

```
1 //////////////////////////////////////////////////
2 // main.c
3
4 #include <stdio.h>
5 #include "calculate.h"
6
7 int
8 main (void)
9 {
10 float Numeral;
(gdb) list calculate.c:20,29
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) list calculate.c:20,29
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)
```



```
[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=0x7fffffffdea4 "-") at calculate.c:21
21     printf("Вычитаемое: ");
(gdb) backtrace
#0 Calculate (Numeral=5, Operation=0x7fffffffdea4 "-") at calculate.c:21
#1 0x00000000004014eb in main () at main.c:17
(gdb) print Numeral
$1 = 5
(gdb) display Numeral
1: Numeral = 5
(gdb) info breakpoints
Num   Type             Disp Enb Address                  What
1     breakpoint        keep y   0x000000000040120f in Calculate at calculate.c:21
      breakpoint already hit 1 time
(gdb) delete 1
(gdb)
```

Рис. 10: GDB

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

```
ksagderov@fedora:~/work/os/lab_prog
+ Запрос данных...
+ Проверка изменений...
+ Установка пакетов...
Splint 3.1.2 --- 23 Jul 2022

calculate.h:7:37: Function parameter Operation declared as manifest array (size
constant is meaningless)
A formal parameter is declared as an array with size. The size of the array
is ignored in this context, since the array formal parameter is treated as a
pointer. (Use -fixedformalarray to inhibit warning)
calculate.c:10:31: Function parameter Operation declared as manifest array
(size constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:16:1: Return value (type int) ignored: scanf("%f", &Sec...
Result returned by function call is not used. If this is intended, can cast
result to (void) to eliminate message. (Use -retvalint to inhibit warning)
calculate.c:22:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:28:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:35:6: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:36:9: Dangerous equality comparison involving float types:
SecondNumeral == 0
Two real (float, double, or long double) values are compared directly using
== or != primitive. This may produce unexpected results since floating point
representations are inexact. Instead, compare the difference to FLT_EPSILON
or DBL_EPSILON. (Use -realcompare to inhibit warning)
calculate.c:39:7: Return value type double does not match declared type float:
(HUGE_VAL)
To allow all numeric types to match, use +relaxtypes.
calculate.c:47:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:48:7: Return value type double does not match declared type float:
(pow(Numeral, SecondNumeral))
calculate.c:51:7: Return value type double does not match declared type float:
(sqrt(Numeral))
calculate.c:53:7: Return value type double does not match declared type float:
(sin(Numeral))
calculate.c:55:7: Return value type double does not match declared type float:
```

```
[ksagderov@fedora lab_prog]$ splint main.c
Splint 3.1.2 --- 23 Jul 2022

calculate.h:7:37: Function parameter Operation declared as manifest array (size
                    constant is meaningless)
    A formal parameter is declared as an array with size. The size of the array
    is ignored in this context, since the array formal parameter is treated as a
    pointer. (Use -fixedformalarray to inhibit warning)
main.c: (in function main)
main.c:14:1: Return value (type int) ignored: scanf("%f", &Num...
    Result returned by function call is not used. If this is intended, can cast
    result to (void) to eliminate message. (Use -retvalint to inhibit warning)
main.c:16:12: Format argument 1 to scanf (%s) expects char * gets char [4] *:
    &Operation
    Type of parameter is not consistent with corresponding code in format string.
    (Use -formattype to inhibit warning)
    main.c:16:9: Corresponding format code
main.c:16:1: Return value (type int) ignored: scanf("%s", &Ope...

Finished checking -- 4 code warnings
[ksagderov@fedora lab_prog]$
```

Рис. 12: Splint

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

Спасибо за внимание!
