

Лабораторная работа 13 Средства, применяемые при разработке программного обеспечения в ОС типа UNIX/Linux

Хусаинова Динара Айратовна

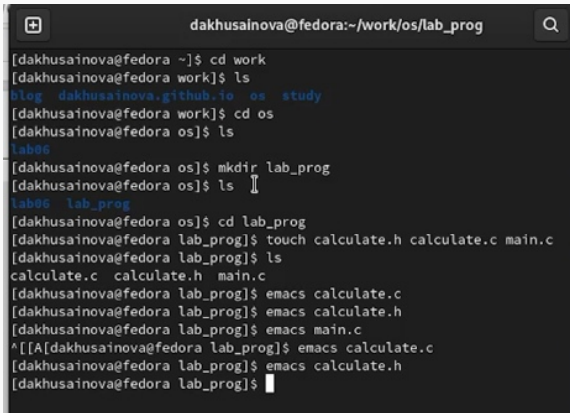
02.06.2022

RUDN

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.

Создание файлов

В домашнем каталоге создаем подкаталог `~/work/os/lab_prog`. А после создаем в нём файлы: `calculate.h`, `calculate.c`, `main.c`. (рис. 1).

A terminal window with a dark background and light text. The title bar shows the user 'dakhusainova' on a 'fedora' machine in the directory '~/work/os/lab_prog'. The terminal shows a sequence of commands: navigating to the 'work' directory, listing contents, navigating to the 'os' directory, listing contents (showing 'lab06'), creating a new directory 'lab_prog', listing contents (showing 'lab06' and 'lab_prog'), navigating into 'lab_prog', and using the 'touch' command to create three files: 'calculate.h', 'calculate.c', and 'main.c'. Finally, the 'ls' command is used to list the files, showing 'calculate.c', 'calculate.h', and 'main.c'. The terminal ends with the 'emacs' command being used to edit each of the three files in sequence, with the cursor visible at the end of the last command.

```
dakhusainova@fedora: ~/work/os/lab_prog
[dakhusainova@fedora ~]$ cd work
[dakhusainova@fedora work]$ ls
blog dakhusainova.github.io os study
[dakhusainova@fedora work]$ cd os
[dakhusainova@fedora os]$ ls
lab06
[dakhusainova@fedora os]$ mkdir lab_prog
[dakhusainova@fedora os]$ ls
lab06 lab_prog
[dakhusainova@fedora os]$ cd lab_prog
[dakhusainova@fedora lab_prog]$ touch calculate.h calculate.c main.c
[dakhusainova@fedora lab_prog]$ ls
calculate.c calculate.h main.c
[dakhusainova@fedora lab_prog]$ emacs calculate.c
[dakhusainova@fedora lab_prog]$ emacs calculate.h
[dakhusainova@fedora lab_prog]$ emacs main.c
^[[A[dakhusainova@fedora lab_prog]$ emacs calculate.c
[dakhusainova@fedora lab_prog]$ emacs calculate.h
[dakhusainova@fedora lab_prog]$
```

Figure 1: Создание файлов и каталога

Реализация функций калькулятора в файле calculate.h(рис. 2).

```
[dakhusainova@fedora lab_prog]$ cat calculate.h
#ifndef CALCULATE_H_
#define CALCULATE_H_

float Calculate(float Numeral, char Operation[4]);

#endif /*CALCULATE_H_*/
```

Figure 2: calculate.h

Реализация функций калькулятора в файле calculate.c(рис. 3).

```
[dakhusainova@fedora lab_prog]$ cat 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);}
```

Figure 3: calculate.c

Реализация функций калькулятора в файле main.c(рис. 4).

```
[dakhusainova@fedora lab_prog]$ cat 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;
}
[dakhusainova@fedora lab_prog]$
```

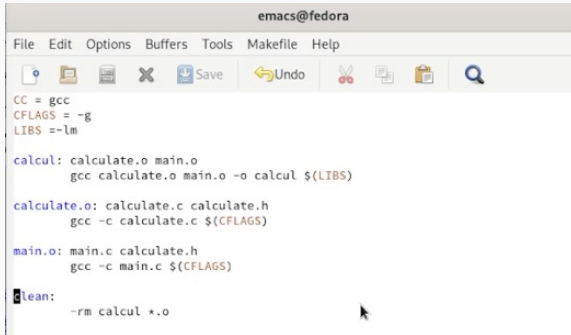
Figure 4: main.c

Выполняем компиляцию программы посредством gcc(рис. 5).

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

Figure 5: Компиляция

Создаем Makefile со следующим содержанием(рис. 6).



```
emacs@fedora
File Edit Options Buffers Tools Makefile Help
[Icons: Open, Save, Undo, Redo, Search]

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
```

Figure 6: Makefile

Запускаем наш калькулятор и проверяем его работу. Просматриваем строки файлов, ставим точки останова, запускаем программу внутри отладчика и убеждаемся, что программа остановится в момент прохождения точки останова(рис. 7,8,9,10).

Квадратный корень

```
Число: 4
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): sqrt
2.00
[Inferior 1 (process 6227) exited normally]
(gdb) list
1      #include <stdio.h>
2      #include "calculate.h"
3      int
4      main (void)
5      {
6          float Numeral;
7          char Operation[4];
8          float Result;
9          printf("Число: ");
10         scanf("%f",&Numeral);
(gdb) █
```

Figure 7: Извлекаем квадратный корень и просматриваем строки файла

```
10      scanf("%d", &Numeral);  
(gdb) list 12,15  
12      scanf("%s",&Operation);  
13      Result = Calculate(Numeral, Operation);  
14      printf("%6.2f\n",Result);  
15      return 0;  
(gdb)
```

Figure 8: Просматриваем строки с 12 по 15

```
21      return NODE_VAL;
(gdb) break 22
Breakpoint 2 at 0x401295: file calculate.c, line 22.
(gdb) run
Starting program: /home/dakhusainova/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 6
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): /

Breakpoint 2, Calculate (Numeral=6, Operation=0x7fffffffdee4 "/" ) at
22      else if(strncmp Operation, "/", 1) == 0){
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/dakhusainova/work/os/lab_prog/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 2, Calculate (Numeral=5, Operation=0x7fffffffdee4 "/" ) at
22      else if(strncmp Operation, "/", 1) == 0){
(gdb)
```

Figure 9: Ставим точки останова и наблюдаем, как останавливается программа при попытке вычислить выражение

```
(gdb) print Numeral
$1 = 5
(gdb) display Numeral
1: Numeral = 5
(gdb) info breakpoints
Num   Type             Disp Enb Address            What
1     breakpoint       keep y   0x0000000000401286 in Calculate at calculate.c:21
2     breakpoint       keep y   0x0000000000401295 in Calculate at calculate.c:22
      breakpoint already hit 1 time
(gdb) delete 1
(gdb) delete 2
Undefined command: "delete". Try "help".
(gdb) info breakpoints
Num   Type             Disp Enb Address            What
2     breakpoint       keep y   0x0000000000401295 in Calculate at calculate.c:22
      breakpoint already hit 1 time
(gdb) delete 2
(gdb) info breakpoints
No breakpoints or watchpoints.
(gdb)
```

Figure 10: Просматриваем значение переменной и удаляем точки останова

С помощью утилиты splint проанализируем коды файлов calculate.c и main.c(рис. 11).

```
[dakhusainova@fedora lab_prog]$ splint calculate.c
Splint 3.1.2 --- 23 Jul 2021

calculate.h:4:37: 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:7:31: Function parameter Operation declared as manifest array (size
        constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:12:5: 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:16:5: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:20:5: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:24:5: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:25:8: 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:27:13: Return value type double does not match declared type float:
```

Figure 11: Просмотр файла calculate.c

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.