

# **Архитектура компьютеров и операционные системы | Операционные системы**

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

Мугари Абдеррахим - НКАбд-03-22

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы :</b>	<b>6</b>
2.1	Контрольные вопросы: . . . . .	10
2.2	выводы по результатам выполнения заданий: . . . . .	10
<b>3</b>	<b>Выводы, согласованные с целью работы:</b>	<b>11</b>

## Список иллюстраций

2.1	создание подкаталога . . . . .	6
2.2	создание файлов программы . . . . .	6
2.3	компиляция файлов . . . . .	7
2.4	создание Makefile . . . . .	7
2.5	запуск программы и управление кодом . . . . .	8
2.6	добавление точки останова . . . . .	8
2.7	проверка функционирования точки останова . . . . .	9
2.8	проверка значения операнда . . . . .	9
2.9	роанализировать коды файлов calculate.c . . . . .	9
2.10	роанализировать коды файлов calculate.c . . . . .	10

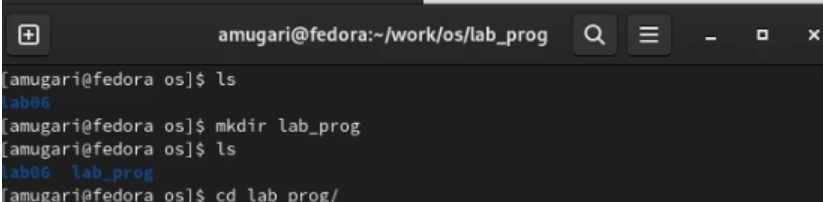
## Список таблиц

# 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 :

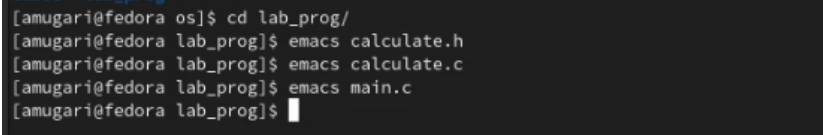
- Сначала в домашнем каталоге я создал подкаталог `~/work/os/lab_prog` (рис. 2.1)



```
amugari@fedora: ~/work/os/lab_prog
[amugari@fedora os]$ ls
lab06
[amugari@fedora os]$ mkdir lab_prog
[amugari@fedora os]$ ls
lab06  lab_prog
[amugari@fedora os]$ cd lab_prog/
```

Рис. 2.1: создание подкаталога

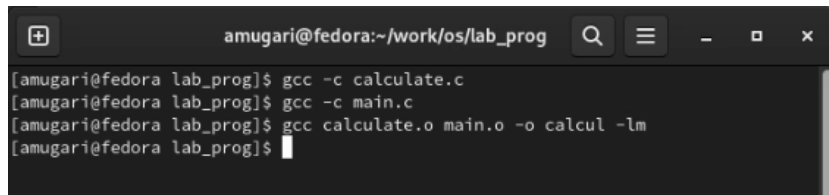
- затем я создал в нем три файла: **calculate.h**, **calculate.c**, **main.c** и поместил в них код (рис. 2.2)



```
[amugari@fedora os]$ cd lab_prog/
[amugari@fedora lab_prog]$ emacs calculate.h
[amugari@fedora lab_prog]$ emacs calculate.c
[amugari@fedora lab_prog]$ emacs main.c
[amugari@fedora lab_prog]$
```

Рис. 2.2: создание файлов программы

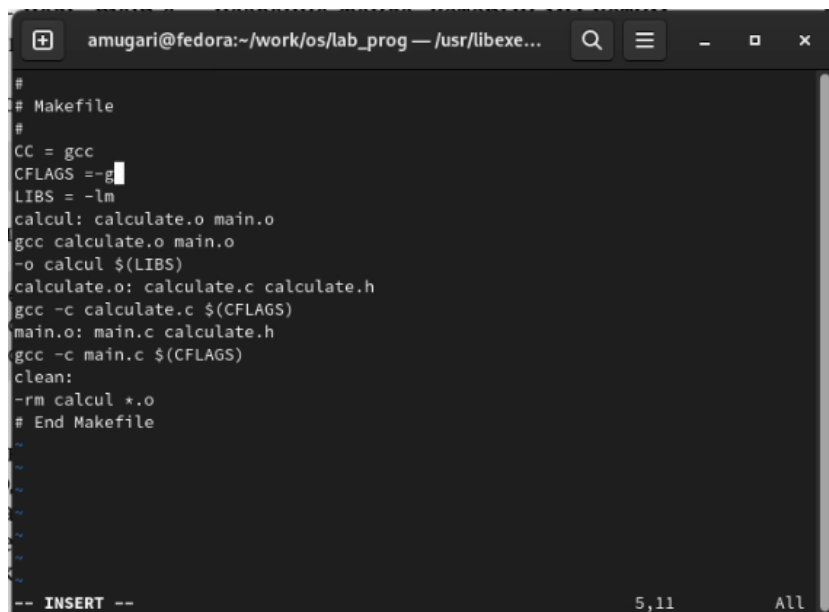
- затем я скомпилировал программу, используя `gcc` (рис. 2.3)

A terminal window with a dark background. The title bar shows 'amugari@fedora:~/work/os/lab\_prog'. The terminal contains four lines of text: '[amugari@fedora lab\_prog]\$ gcc -c calculate.c', '[amugari@fedora lab\_prog]\$ gcc -c main.c', '[amugari@fedora lab\_prog]\$ gcc calculate.o main.o -o calcul -lm', and '[amugari@fedora lab\_prog]\$' followed by a cursor.

```
amugari@fedora:~/work/os/lab_prog
[amugari@fedora lab_prog]$ gcc -c calculate.c
[amugari@fedora lab_prog]$ gcc -c main.c
[amugari@fedora lab_prog]$ gcc calculate.o main.o -o calcul -lm
[amugari@fedora lab_prog]$
```

Рис. 2.3: компиляция файлов

- затем я создал **Makefile** и добавил его скрипт (рис. 2.4)

A terminal window with a dark background. The title bar shows 'amugari@fedora:~/work/os/lab\_prog — /usr/libexe...'. The terminal contains the content of a Makefile. At the bottom, it shows '-- INSERT --', '5,11', and 'All'.

```
amugari@fedora:~/work/os/lab_prog — /usr/libexe...
#
# 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

-- INSERT --
5,11 All
```

Рис. 2.4: создание Makefile

- после этого я запустил программу и отобразил заголовок **main.c**, а также отобразил строки между 12 и 15 из **main.c** (рис. 2.5)

```
amugari@fedora:~/work/os/lab_prog — gdb ./calcul
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): +
Второе слагаемое: 12
24.00
[Inferior 1 (process 15061) exited normally]
(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);
(gdb)
```

Рис. 2.5: запуск программы и управление кодом

- затем я проверил, что в файле calculate.c нет точек останова, а затем добавил точку останова в строку 21 (рис. 2.6)

```
(gdb) info breakpoints
No breakpoints or watchpoints.
(gdb) list calculate.c:20,27
20 scanf("%f",&SecondNumeral);
21 return(Numeral - SecondNumeral);
22 }
23 else if(strcmp(Operation, "+") == 0)
24 {
25 printf("Нумератор: ");
26 scanf("%f",&SecondNumeral);
27 return(Numeral + SecondNumeral);
(gdb) break 21
Breakpoint 6 at 0x401234: file calculate.c, line 21.
(gdb)
В МОМЕНТ ПРОХОЖДЕНИЯ ТОЧКИ ОСТАНОВА:
```

Рис. 2.6: добавление точки останова

- после этого я повторно запустил файл и убедился, что программа остановится на 21 строке, и это сработало (рис. 2.7)



```

(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/amugari/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 21
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -
Вычитаемое: 12

Breakpoint 6, Calculate (Numeral=21, Operation=0x7fffffffda34 "-") at calculate.c:21
21      return Numeral - SecondNumeral;
(gdb)

```

Рис. 2.7: проверка функционирования точки останова

- после этого я проверил правильность значения первого операнда этой операции (рис. 2.8)

```

(gdb) print Numeral
$1 = 21
(gdb) display Numeral
1: Numeral = 21
(gdb) info breakpoints
Num   Type             Disp Enb Address                  What
6     breakpoint       keep y  0x0000000000401234 in Calculate at calculate.c:21
      breakpoint already hit 1 time
(gdb) delete 6
(gdb) info breakpoints
No breakpoints or watchpoints.
(gdb)

```

Рис. 2.8: проверка значения операнда

- наконец, используя утилиту splint, попытался проанализировать коды файлов calculate.c и main.c. (рис. 2.9) (рис. 2.10)

```

Splint 3.1.2 --- 23 Jul 2022

calculate.h:6:37: 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:8:31: Function parameter Operation declared as manifest array (size
constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:14:1: Return value (type int) ignored: scanf("%f", &Sec...
Result returned by function call is not used. If this is intended, can cast

```

Рис. 2.9: роанализировать коды файлов calculate.c

```

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

calculate.h:6:37: 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:13:1: 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:15:12: 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:15:9: Corresponding format code
main.c:15:1: Return value (type int) ignored: scanf("%s", &Ope...
Finished checking --- 4 code warnings
[amugari@fedora lab_prog]$

```

Рис. 2.10: проанализировать коды файлов calculate.c

## 2.1 Контрольные вопросы:

## 2.2 выводы по результатам выполнения заданий:

- 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.

### **3 Выводы, согласованные с целью 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.