

# **Лабораторная работа № 13**

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

Шулуужук Айраана Вячеславовна НПИбд-02-22

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>8</b>
3.1	Этапы разработки приложений . . . . .	8
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
<b>5</b>	<b>Выводы</b>	<b>13</b>

## Список иллюстраций

4.1	создание файлов и каталога . . . . .	9
4.2	calculate.c . . . . .	9
4.3	calculate.h . . . . .	10
4.4	main.c . . . . .	10
4.5	компиляция программы . . . . .	10
4.6	Makefile . . . . .	11
4.7	скрипт 3 . . . . .	11
4.8	анализ файла calculate.c . . . . .	12
4.9	анализ файла main.c . . . . .	12

## **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 Задание

1. Написать командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени  $t_1$  дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени  $t_2 < t_1$ , также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом). Запустить командный файл в одном виртуальном терминале в фоновом режиме, перенаправив его вывод в другой (`> /dev/tty#`, где `#` — номер терминала куда перенаправляется вывод), в котором также запущен этот файл, но не фоновом, а в привилегированном режиме. Доработать программу так, чтобы имела возможность взаимодействия трёх и более процессов.
2. Реализовать команду `man` с помощью командного файла. Изучите содержимое каталога `/usr/share/man/man1`. В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в системе программ и команд. Каждый архив можно открыть командой `less` сразу же просмотрев содержимое справки. Командный файл должен получать в виде аргумента командной строки название команды и в виде результата выдавать справку об этой команде или сообщение об отсутствии справки, если соответствующего файла нет в каталоге `man1`.
3. Используя встроенную переменную `$RANDOM`, напишите командный файл, генерирующий случайную последовательность букв латинского алфавита.

Учтите, что \$RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767.

## 3 Теоретическое введение

### 3.1 Этапы разработки приложений

Процесс разработки программного обеспечения обычно разделяется на следующие этапы:

- планирование, включающее сбор и анализ требований к функционалу и другим характеристикам разрабатываемого приложения;
- проектирование, включающее в себя разработку базовых алгоритмов и спецификаций, определение языка программирования;
- непосредственная разработка приложения:
- кодирование — по сути создание исходного текста программы (возможно в нескольких вариантах);
- анализ разработанного кода;
- сборка, компиляция и разработка исполняемого модуля;
- тестирование и отладка, сохранение произведённых изменений;
- документирование.

Для создания исходного текста программы разработчик может воспользоваться любым удобным для него редактором текста: vi, vim, mceditor, emacs, geany и др. После завершения написания исходного кода программы (возможно состоящей из нескольких файлов), необходимо её скомпилировать и получить исполняемый модуль.



## 4 Выполнение лабораторной работы

Создадим нужный каталог и необходимые файлы (рис. 4.1)

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

Рис. 4.1: создание файлов и каталога

Внесем тексты программ в файлы (рис. 4.2) (рис. 4.3) (рис. 4.4)

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

Рис. 4.2: calculate.c

```

////////////////////////////////////
// calculate.h

#ifndef CALCULATE_H_
#define CALCULATE_H_

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

#endif /*CALCULATE_H_*/
~

```

Рис. 4.3: calculate.h

```

////////////////////////////////////
// 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;
}
~

```

Рис. 4.4: main.c

Выполним компиляцию программы посредством gcc (рис. 4.5)

```

[avshuluuzhuk@fedora lab_prog1]$ gcc -c calculate.c
[avshuluuzhuk@fedora lab_prog1]$ gcc -c main.c
[avshuluuzhuk@fedora lab_prog1]$ gcc calculate.o main.o -o calcul -lm
[avshuluuzhuk@fedora lab_prog1]$ ls
calcul      calculate.c~ calculate.h~  main.c      main.o
calculate.c calculate.h  calculate.o  main.c~
[avshuluuzhuk@fedora lab_prog1]$

```

Рис. 4.5: компиляция программы

Создадим файл Makefile со следующим содержанием (рис. 4.6)

```

#
# Makefile
#

CC = gcc
CFLAGS =
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)

15 main.o: main.c calculate.h
    gcc -c main.c $(CFLAGS)

clean:
    -rm calcul *.o *~

# End Makefile

```

Рис. 4.6: Makefile

С помощью `gdb` выполним отладку программы `calcul`. Запустим отладчик GDB, загрузив в него программу для отладки и введем команду `run` (рис. 4.7)

```

(gdb) run
Starting program: /home/avshuluuzhuk/work/os/lab_prog1/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): +
Второе слагаемое: 3
      8.00
[Inferior 1 (process 4867) exited normally]
(gdb)

```

Рис. 4.7: скрипт 3

С помощью утилиты `splint` проанализируем коды файлов `calculate.c` и `main.c` (рис. 4.8) (рис. 4.9)

```

[avshuluuzhuk@fedora lab_prog]$ splint calculate.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)
calculate.c:10:31: Function parameter Operation declared as manifest array
        (size constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:16:7: 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:22:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:28:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:34:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:35:10: 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:38:10: Return value type double does not match declared type float:
        (HUGE_VAL)
    To allow all numeric types to match, use +relaxtypes.
calculate.c:46:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:47:13: Return value type double does not match declared type float:
        (pow(Numeral, SecondNumeral))
calculate.c:50:11: Return value type double does not match declared type float:
        (sqrt(Numeral))
calculate.c:52:11: Return value type double does not match declared type float:
        (sin(Numeral))
calculate.c:54:11: Return value type double does not match declared type float:
        (cos(Numeral))
calculate.c:56:11: Return value type double does not match declared type float:
        (tan(Numeral))
calculate.c:60:13: Return value type double does not match declared type float:
        (HUGE_VAL)

Finished checking --- 15 code warnings
[avshuluuzhuk@fedora lab_prog]$

```

Рис. 4.8: анализ файла calculate.c

```

Finished checking --- 4 code warnings
[avshuluuzhuk@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:14:3: 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:16:14: 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:16:11: Corresponding format code
main.c:16:3: Return value (type int) ignored: scanf("%s", &Ope...

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

```

Рис. 4.9: анализ файла main.c

## 5 Выводы

В ходе выполнения работы мы 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.