

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

Лабораторная работа №13

---

Толстых М. А.

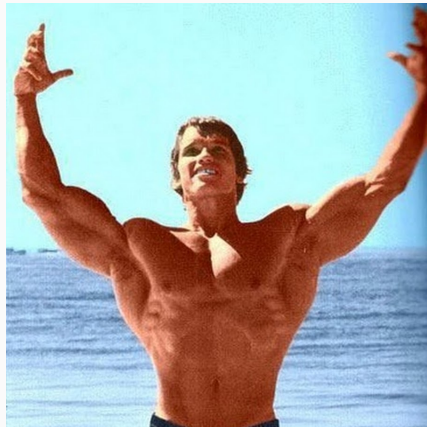
4 мая 2023

Российский университет дружбы народов, Москва, Россия

## Информация

---

- Толстых Максим Алексеевич
- студент 1 курса, группа НММбд-03-22
- Российский университет дружбы народов



## Вводная часть

---

- Командный процессор ОС UNIX
- Командные файлы

- 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.

- Ознакомиться с теоретическим материалом.
- Выполнить упражнения.
- Ответить на контрольные вопросы.

## Выполнение лабораторной работы №13

---



```
[matolstikh@fedora ~]$ mkdir ~/work/os/lab_prog  
[matolstikh@fedora ~]$ ls ~/work/os/  
lab08 lab09 lab_prog
```

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

```
//////////////////////////////////
// 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("Добавление: ");
        scanf("%f", &SecondNumeral);
        return(Numeral + SecondNumeral);
    }
    else if(strncmp(Operation, "-", 1) == 0)
    {
        printf("Вычитание: ");
        scanf("%f", &SecondNumeral);
        return(Numeral - SecondNumeral);
    }
}
```

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

#ifndef CALCULATE_H
#define CALCULATE_H

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

#endif /*CALCULATE_H*/
```

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

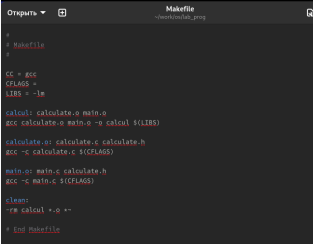
#include <stdio.h>
#include "calculate.h"

int
main (void)
{
    float Numeral;
    char Operation[4];
    float Result;
    printf("Введите: ");
    scanf("%f", &Numeral);
    printf("Операция (+, -, *, /, pow, sqrt, sin, cos, tan): ");
    scanf("%s", &Operation);
    Result = Calculate(Numeral, Operation);
    printf("%8.2f", Result);
    return 0;
}
```

# Компиляция и Makefile

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

```
[matolstikh@fedora lab_prog]$ gcc calculate.o main.o -o calcul -lm
[matolstikh@fedora lab_prog]$ touch Makefile
```



```
Открыть ▾
Makefile
~/.work/ou/lab_prog

#
# 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)

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

clean:
rm calcul *.o *~

# End Makefile
```

```
[matolstikh@fedora lab_prog]$ gdb ./calcul
```

```
(gdb) run
Starting program: /home/matolstikh/work/os/lab_prog/calcul
d *Downloading 0.01 MB separate debug info for system-supplied DSO at 0x7ffff7c000
Downloading 2.25 MB separate debug info for /lib64/libc.so.6
Downloading 7.42 MB separate debug info for /lib64/libc.so.6
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Welcome!
Onepage (+,-,*,/,pow,sqrt,sin,cos,tan): 8topoe charaemoe: +5
4.00
[Inferior 1 (process 31198) exited normally]
```

```
(gdb) list
Downloading 0.00 MB source file /usr/src/debug/glibc-2.35-22.fc36.x86_64/sf/softn.c
1 /* Terminate the frame unwind info section with a 4byte 0 as a sentinel;
2    this would be the 'length' field in a real FDE.  */
3
4 typedef unsigned int u123 __attribute__((mode(ST)))
5 static const u123 __FRAME_END__ = {
6   __attribute__((used section ("eh_frame")))
7   0
8 };
(gdb)
Line number 8 out of range; softn.c has 7 lines.
```

# Работа с отладчиком

```
(gdb) list 1, 4
1      /* Terminate the frame unwind info section with a byte 0 as a sentinel;
2         this would be the 'length' field in a real FDE. */
3
4      __attribute__((unwind_init_u31 __attribute__((mode(SI))))
```

```
(gdb) break 21
No line 21 in the current file.
Make breakpoint pending on future shared library load? (y or [n]) y
Breakpoint 1 (21) pending.
```

```
(gdb) b
Num      Type      Disp Enb Address  What
1        breakpoint keep y   <PENDING> 21
(gdb)
```

# Анализ с помощью утилиты splint

```
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:16:5: 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:22:5: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:28:5: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:34:5: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:35:8: 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:38:13: Return value type double does not match declared type float:
(HUGE_VAL)
To allow all numeric types to match, use *relaxtypes.
calculate.c:46:5: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:47:11: 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))
```

```
[matolstikh@fedora lab_prog]$ splint main.c
Splint 3.1.2 --- 22 Jan 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:13:3: 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:15:14: 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:15:11: Corresponding format code
main.c:15:3: Return value (type int) ignored: scanf("%s", &Ope...

Finished checking src 4 code warnings
```

1. Чтобы получить информацию о возможностях программ gcc, make, gdb и др. нужно воспользоваться командой man или опцией -help (-h) для каждой команды.
2. 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 (возмож- но в нескольких вариантах); – анализ разработанного кода; – сборка, компиляция и разработка исполняемого модуля;

4. Основное назначение компилятора языка Си в UNIX заключается в компиляции всей программы и получении исполняемого файла/модуля.
5. Для сборки разрабатываемого приложения и собственно компиляции полезно воспользоваться утилитой make. Она позволяет автоматизировать процесс преобразования файлов программы из одной формы в другую, отслеживает взаимосвязи между файлами.
6. Для работы с утилитой make необходимо в корне рабочего каталога с Вашим проектом создать файл с названием makefile или Makefile, в котором будут описаны правила обработки файлов Вашего программного комплекса. В самом простом случае Makefile имеет следующий синтаксис: ... : ...  
<команда 1> ... Сначала записывается список целей, разделённых пробелами

## Результаты

---



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