

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

Лабораторная работа №13

Нати Ф. Б.

Российский университет дружбы народов, Москва, Россия

Информация

- Нати Франсиску Бунда
- студент 1 курса, группа НММбд-02-22
- 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

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


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

```
Открыть ▾  • calculate.c
~\work\os\lab_prog  

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


```
Открыть ▾  • calculate.h
~\work\os\lab_prog

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

#ifndef CALCULATE_H_
#define CALCULATE_H_

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

#endif /*CALCULATE_H_*/
```

```
Открыть ▾  • main.c
~\work\os\lab_prog


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

Компиляция и Makefile

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

```
[fbnati@fedora lab_prog]$ touch Makefile
```

Открыть ▾  Makefile
~/work/ios/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
```

Работа с отладчиком

```
(gdb) run
Starting program: /home/fbnati/work/os/lab_prog/calcul
Downloading separate debug info for /home/fbnati/work/os/lab_prog/system-s
4 DSO at 0x7ffff7fc400e...

Downloading separate debug info for /lib64/libc.so.6...
Downloading separate debug info for /lib64/libc.so.6...
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 4
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): *
Второе слагаемое: 6
10.00
Inferior 1 (process 7446) exited normally
(gdb)
```

```
(gdb) list
Downloading source file /usr/src/debug/glibc-2.25-4.fc30.x86_64/elf/sofini.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 u132 __attribute__((mode(SI)));
5 static const u132 __FRAME_END__[1]
6   __attribute__((used, section(".eh_frame")))
7   = { 0 };
```

```
(gdb) list 1, 4
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 u132 __attribute__((mode(SI)));
```

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

```
(gdb) print Numeral
```

```
(gdb) i b
Num    Type      Disp Enb Address  What
1      breakpoint keep y  <PENDING> 21
(gdb) delete 1
(gdb) i b
No breakpoints or watchpoints.
```

Анализ с помощью утилиты splint

```
[fbnati@fedora lab_prog]$ splint calculate.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)
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)
```

```
[fbnati@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 --- 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 (возможно в нескольких вариантах); – анализ разрабо-

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: ... : ...

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 C калькулятора с простейшими функциями.