

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

Лабораторная работа №13

Старцева А. С.

5 мая 2023

Российский университет дружбы народов, Москва, Россия

Информация

- Старцева Алина Сергеевна
- 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 C калькулятора с простейшими функциями.

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


Выполнение лабораторной работы №13

```
[astarceva@asstarceva ~]$ mkdir ~/work/os/lab_prog  
[astarceva@asstarceva ~]$ ls ~/work/os  
lab08  lab_prog
```


```
[astarceva@asstarceva os]$ cd lab_prog  
[astarceva@asstarceva lab_prog]$ touch calculate.h calculate.c main.c  
[astarceva@asstarceva lab_prog]$ ls  
calculate.c calculate.h main.c
```

Открыть ▾  • calculate.c
~/work/cn/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/cn/lab_prog

```
////////////////////////////////////  
// calculate.h  
  
#ifndef CALCULATE_H_  
#define CALCULATE_H_  
  
float Calculate(float Numeral, char Operation[4]);  
  
#endif /*CALCULATE_H_*/
```

Открыть ▾  main.c
~/work/cn/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("%%.2f\n", Result);  
    return 0;  
}
```

Компиляция и Makefile

```
(astarceva@asstarceva lab_prog)$ gcc -c calculate.c
(astarceva@asstarceva lab_prog)$ gcc -c main.c
(astarceva@asstarceva lab_prog)$ gcc calculate.o main.o -o calcul -lm
```

```
(astarceva@asstarceva lab_prog)$ touch Makefile
(astarceva@asstarceva lab_prog)$ ls
calcul calculate.c calculate.h calculate.o main.c main.o Makefile
```

```
Открыть ▾  Makefile
~/work/na/na_lab_prog

# Makefile
#

CC = gcc
CFLAGS = -g
LDS = -le

calcul: calculate.o main.o
$(CC) calculate.o main.o -o calcul $(LDS)

calculate.o: calculate.c calculate.h
$(CC) -c calculate.c $(CFLAGS)

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

clean:
rm calcul *.o *.a

# End Makefile
```

Работа с отладчиком

```
astarceva@astarceva lab_prog$ gdb ./calcul
GNU gdb (GDB) Fedora 12.1.2.fc36
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./calcul...

This GDB supports auto-downloading debuginfo from the following URLs:
https://debuginfod.fedoraproject.org/
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
```

```
(gdb) run
Starting program: /home/astarceva/work/os/lab_prog/calcul
^
```

```
(gdb) list
Downloading 0.00 MB source file /usr/src/debug/glibc-2.35-22.fc36.x86_64/elf/w
1 int c
2 /* Terminate the frame unwind info section with a 4byte 0 as a sentinel
3    this would be the 'length' field in a real FDE. */
4
5 typedef unsigned int ui32 __attribute__((mode(SI)));
6 static const ui32 __FRAME_END__[1]
7     __attribute__((used, section(".eh_frame")))
8     = { 0 };
(gdb)
```

```
(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 u32 __attribute__((mode(SI)));
(gdb)
```

```
(gdb) list calculate.c:20,29
```

```
(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.
```

Анализ с помощью утилиты splint

```
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))
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:60:11: Return value type double does not match declared type float
    (HUGE_VAL)
```

```
Finished checking --- 4 code warnings
[astarceva@asstarceva 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 (возмож- но в нескольких вариантах); – анализ разработанного кода; – сборка, компиляция и разработка исполняемого модуля; – тестирование и отладка, сохранение произведённых изменений; • доку-

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 задаётся список целей, разделённых пробелами, за которым идёт двоеточие и список зависимостей. Затем в следующих

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.