

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

Лабораторная работа №13

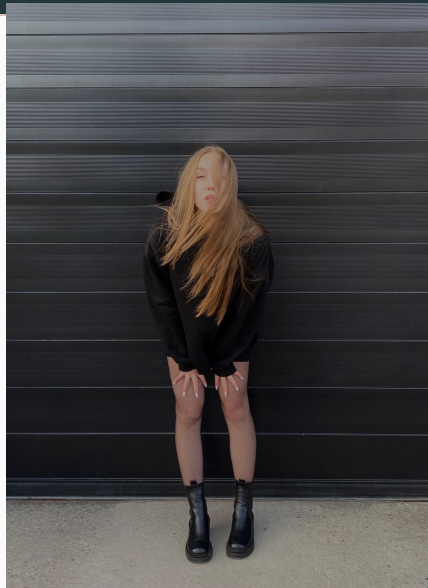
Миронова М. В.

4 мая 2023

Российский университет дружбы народов, Москва, Россия

Информация

- Миронова Мария Вадимовна
- 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

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


```
[mvmironova@fedora ~]$ cd ~/work/os/lab_prog  
[mvmironova@fedora lab_prog]$ touch calculate.c calculate.h main.c  
[mvmironova@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("Складываем: ");  
        scanf("%f", &SecondNumeral);  
        return(Numeral + SecondNumeral);  
    }  
    else if(strncmp(Operation, "-", 1) == 0)  
    {  
        printf("Вычитаемое: ");  
        scanf("%f", &SecondNumeral);  
        return(Numeral - SecondNumeral);  
    }  
    else if(strncmp(Operation, "*", 1) == 0)  
    {  
        printf("Умножаем: ");  
        scanf("%f", &SecondNumeral);  
        return(Numeral * SecondNumeral);  
    }  
    else if(strncmp(Operation, "/", 1) == 0)  
    {  
        printf("Делим: ");  
        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;  
}
```

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

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

```
• Makefile
~/work/04/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/mveironova/work/os/lab_prog/calcul
Downloading separate debug info for /home/mveironova/work/os/lab_prog/system-
d1ed 050 at 0x7ffff7c4000...

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: 3
операция (+,-,*,/,pow,sqrt,sin,cos,tan): +
множитель: 6
18.00
[Inferior 1 (process 94674) exited normally]
```

```
(gdb) list
Downloading source file /usr/src/debug/glibc-2.35-4.fc36.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 u32 __attribute__((mode(SI)));
5 static const u32 __FRAME_END__ = 0;
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 u32 __attribute__((mode(SI)))
```

```
(gdb) print Numeral
```

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

```
(gdb) break 21
```

Анализ с помощью утилиты splint

```
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))
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:
```

```
Finished checking -- 25 code warnings
[mvmironova@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
```

9. При первом запуске компилятор не выдал никаких ошибок, но в коде программы `main.c` допущена ошибка, которую компилятор мог пропустить (возможно, из-за версии 8.3.0-19): в строке `scanf("%s", &Operation);` нужно убрать знак `&`, потому что имя массива символов уже является указателем на первый элемент этого массива.

10. Система разработки приложений UNIX предоставляет различные средства, повышающие понимание исходного кода. К ним относятся:

`cscope` – исследование функций, содержащихся в программе,

`lint` – критическая проверка программ, написанных на языке Си.

12. Утилита `splint` анализирует программный код, проверяет корректность задания аргументов использованных в программе функций и типов возвращаемых значений, обнаруживает синтаксические и семантические ошибки.

значения которых отображаются при достижении точки останова программы
`finish` – выполнить программу до момента выхода из функции
`info breakpoints` – вывести на экран список используемых точек останова
`info watchpoints` – вывести на экран список используемых контрольных выражений
`list` – вывести на экран исходный код (в качестве параметра может быть указано название файла и через двоеточие номера начальной и конечной строк)
`next` – выполнить программу пошагово, но без выполнения вызываемых в программе функций
`print` – вывести значение указываемого в качестве параметра выражения
`run` – запуск программы на выполнение
`set` – установить новое значение переменной
`step` – пошаговое выполнение программы
`watch` – установить контрольное выражение, при изменении значения которого программа будет остановлена
Для выхода из `gdb` можно воспользоваться командой `quit` (или её сокращённым вариантом `q`) или комбинацией клавиш `Ctrl-d`. Более подробную информацию по работе с `gdb` можно получить с помощью команд `gdb -h` и `man gdb`. Схема отладки

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.