Презентация лабораторной работы №13

Презентация лабораторной работы №13

Цель

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

Задание

- В домашнем каталоге создайте подкаталог ~/work/os/lab_prog.
- Создайте в нём файлы:calculate.h, calculate.c, main.c.
- С помощью gdb выполните отладку программы calcul.
- С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c.

Выполнение лабораторной работы

В домашнем каталоге создаем подкаталог ~/work/os/lab_prog

```
[lkkhanina@fedora ~]$ mkdir ~/work/os/lab_prog
[lkkhanina@fedora ~]$ cd ~/work/os/lab_prog
[lkkhanina@fedora lab_prog]$ vi calculate.c
[lkkhanina@fedora lab_prog]$ vi calculate.h
[lkkhanina@fedora lab_prog]$ vi main.c
```

Создаем подкаталог ~/work/os/lab_prog

Создаем в нём файлы: calculate.h, calculate.c, 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("Второе слагаемое: ");
        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("Множитель: ");
        scanf("%f",&SecondNumeral);
        return(Numeral * SecondNumeral);
      }
    else if(strncmp(Operation, "/", 1) == 0)
      {
        printf("Делитель: ");
        scanf("%f",&SecondNumeral);
        if(SecondNumeral == 0)
```

calculate.c

```
[lkkhanina@fedora lab_prog]$ cat calculate.h
////////////////////////////////////
// calculate.h

#ifndef CALCULATE_H_
#define CALCULATE_H_

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

#endif /*CALCULATE_H_*/
[lkkhanina@fedora lab_prog]$
```

calculate.h

```
[lkkhanina@fedora lab_prog]$ cat main.c
////////////////////////////////////
// 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;
}
[lkkhanina@fedora lab_prog]$
```

main.c

Выполняем компиляцию программы посредством gcc. Синтаксических ошибок нет.

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

Компиляцию программы посредством gcc

Создаем Makefile. Этот файл нужен для автоматической компиляции файлов и объединения их в один исполняемый файд calcul. clean удаляет все файлы. Переменная CC отвечает за утилиту компиляции; CFLAFS — опции данной утилиты; LIBS — опции для объединения объектных файлов в один исполняемый файл

```
[lkkhanina@fedora lab_prog]$ vi Makefile
[lkkhanina@fedora lab_prog]$ cat Makefile
#
# 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
[lkkhanina@fedora lab_prog]$
```

Makefile

Исправляем Makefile и выполняем компиляцию файлов

```
#
# Makefile
#

CC = gcc
CFLAGS = -g
LIBS = -lm

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

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 *~

# End Makefile
```

Испрвляем Makefile

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

Выполняем компиляцию файлов

Запускаем отладчик GDB

```
[lkkhanina@fedora lab_prog]$ gdb ./calcul
GNU gdb (GDB) Fedora 10.2-9.fc35
Copyright (C) 2021 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...
(gdb)
```

Запускаем отладчик GDB

Запускаем программу с помощью команды run

```
(gdb) run
Starting program: /home/lkkhanina/work/os/lab_prog/calcul
Downloading separate debug info for /home/lkkhanina/work/os/lab_prog/system-supplied DSO
 at 0x7ffff7fc9000...
Downloading separate debug info for /lib64/libm.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: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): +
Второе слагаемое: 6
 11.00
[Inferior 1 (process 4170) exited normally]
```

Запускаем программу

Для постраничного просмотра исходного кода используем команду list. Для просмотра строк с 12 по 15 используем команды list 12,15. А чтобы посмотреть определенные строки неосновного файла, то нужно использовать list с параметрами. Например, list calculate.c:20,29

```
(gdb) list
1       ////////////////////////////////////////
2       // main.c
3
4       #include <stdio.h>
5       #include "calculate.h"
6
7       int main (void) {
8           float Numeral;
9           char Operation[4];
10          float Result;
(gdb) list 12,15
12          scanf("%f",&Numeral);
13          printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
14          scanf("%s",&Operation);
15          Result = Calculate(Numeral, Operation);
(gdb) list calculate.c:20,29
20            {
21              printf("Вычитаемое: ");
22              scanf("%f",&SecondNumeral);
23              return(Numeral - SecondNumeral);
24            }
25          else if(strncmp(Operation, "*", 1) == 0)
26            {
27              printf("Множитель: ");
28              scanf("%f",&SecondNumeral);
29              return(Numeral * SecondNumeral);
```

Просмотр строк

Устанавливаем точку останова в файле calculate.c на строке номер 21 и выводим информацию обо всех имеющихся точках останова. Запускаем программу и убеждаемся, что она остановится в момент прохождения breakpoint'а

```
(gdb) list calculate.c:20,27
20              {
21                printf("Вычитаемое: ");
22                scanf("%f",&SecondNumeral);
23                return(Numeral - SecondNumeral);
24              }
25            else if(strncmp(Operation, "*", 1) == 0)
26              {
27                printf("Множитель: ");
(gdb) break 21
Breakpoint 1 at 0x40120f: file calculate.c, line 21.
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x000000000040120f in Calculate at calculate.c:21
```

Устанавливаем точку остановки

```
(gdb) run
Starting program: /home/lkkhanina/work/os/lab_prog/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): -

Breakpoint 1, Calculate (Numeral=5, Operation=0x7fffffffdf34 "-") at calculate.c:21
21                printf("Вычитаемое: ");
(gdb) backtrace
#0  Calculate (Numeral=5, Operation=0x7fffffffdf34 "-") at calculate.c:21
#1  0x00000000004014eb in main () at main.c:15
```

Остановка программы

Посмотрим значение переменной Numeral и сравним его с результатом вывода команды display Numeral

```
(gdb) print Numeral
$1 = 5
(gdb) display Numeral
1: Numeral = 5
```

Numeral

Уберем точки останова

```
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x000000000040120f in Calculate at calculate.c:21
        breakpoint already hit 1 time
(gdb) delete 1
(gdb) info breakpoints
No breakpoints or watchpoints.
```

Уберем точки остановки

С помощью утилиты splint узнаем, что в файле calculate.c просходит сравнение вещесвтенного числа с нулем, а в main.c есть функция scanf, которая возращает целое число, которое нигде не используется

```
[lkkhanina@fedora lab_prog]$ splint calculate.c
Splint 3.1.2 --- 23 Jul 2021

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:16:9: 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:22:9: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:28:9: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:34:9: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:35:12: 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:38:19: Return value type double does not match declared type float:
                     (HUGE_VAL)
  To allow all numeric types to match, use +relaxtypes.
calculate.c:46:8: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:47:14: Return value type double does not match declared type float:
                     (pow(Numeral, SecondNumeral))
calculate.c:50:15: Return value type double does not match declared type float:
                     (sqrt(Numeral))
calculate.c:52:15: Return value type double does not match declared type float:
                     (sin(Numeral))
calculate.c:54:15: Return value type double does not match declared type float:
```

splint calculate.c

```
[lkkhanina@fedora lab_prog]$ splint main.c
Splint 3.1.2 --- 23 Jul 2021

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:12:5: 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:14:16: 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:14:13: Corresponding format code
main.c:14:5: Return value (type int) ignored: scanf("%s", &Ope...

Finished checking --- 4 code warnings
```

splint main.c

Выводы

Я научилась запускать откладчик GDB, работать с ним и тестировать программы