Слуцкий Никита | гр. 053506 (ФКСиС, ИиТП)

Лабораторная работа № 6

Задание 1

1 . Когда вы находитесь в сеансе gdb, как вы устанавливаете аргументы, которые будут передаваться программе при её запуске.

После запуска gdb можно запустить программу, используя “r ags”.

Если код запускается с помощью *$ main.exe …args*, то в gdb это будет происходить как *$ gdb* main, *$ r …args*

2 . Как создать точку останова ?

Можно создать по номеру строки или по названию процедуры.

$ b main или *$ b 3*

3 . Как выполнить следующую строку кода в Си-программе после остановки в точке останова ?

С помощью *n*. При нажатии *n* отладчик пройдёт функцию, не заходя в неё.

4 . Если следующая строка кода является вызовом функции, вы выполните весь вызов функции сразу, если воспользуетесь своим ответом на вопрос №3. Как вы сообщите gdb, что хотите вместо этого перейти внутрь функции ?

Необходимо сделать “шаг внутрь” (*step-in*) или нажать клавишу *s*.

5 . Как продолжить выполнение программы после остановки на точке останова ?

Continue (*c*) продолжит выполнение работы до следующей точки останова или ошибки.

6 . Как можно распечатать значение переменной в gdb ? (или даже выражение типа 1 + 2)

С помощью команды *print*.

7 . Как настроить gdb так, чтобы оно отображал значение переменной после каждого шага ?

С помощью команды *watch*.

8. Как показать список всех переменных и их значений в текущей функции ?

С помощью команды *info args*.

9. Как выйти из gdb ?

Команда *quit* или ввести знак конца файла.

Задание 2

Заменил прямой ввод с клавиатуры на чтение из файла.

Задание 3

1 . Почему важны инструменты вроде Valgrind и санитайзеров, какая в них польза ?

Отслеживание утечек памяти, валидности указателей и т.д.

2 . Как запустить программу в Valgrind ?

Во-первых, это нужно делать в Linux. Во-вторых, *valgrind ./segfault_ex*

3 . Как использовать санитайзер G++ ?

Флаг *-fsanitize=address* включает санитайзер, *-static-libasan* статически связывает программу с библиотекой санитайзера. Флаг *-g* нужен для отображения символов отладки при запуске. После того, как программа собрана с включённым санитайзером, она запускается обычным способом.

4 . Как интерпретировать сообщения об ошибках ?

Для простейшего подхода получение информации об ошибках состоит в том, чтобы читать вывод, например:

==17413== at 0ч47Ф2349: ns_name_ntop (ns_name.c: 147)
можно взглянуть на имя файла, номер строки, имя функции.

5 . Почему неинициализированные переменные могут приводить к появлению Гейзенбагов ?

Использование неинициализированных переменных схоже с использованием неинициализированной памяти и может приводить к ошибкам разного рода в процессе работы программы.
Предположим, что мы не инициализировали переменную в функции. Объявили, но без значения. В отдельных случаях там может оказаться нулевое значение (например, глобальные переменные по умолчанию имеют значение 0). Но в общем случае программа может вести себя неправильно из-за того, что, например, в int переменные при отсутствии дефолтного значения попадает “мусорное” значение, например, 461818864.

Вопросы

1 . Почему не произошёл segfault При no_segfault_ex ?

Потому что мы не присваиваем элемент по индексу, превышающему размер массива, а всего лишь пытаемся получить значение по этому индексу. Segfault вылетел бы при попытке получить именно доступ к памяти, а не просто значение в ней.

2 . Почему no_segfault_ex выдаёт несогласованные выходные данные ?

Потому что в определении цикла мы задаём *j < sizeof(a)*, а так нельзя, потому что *sizeof* возвращает размер массива в байтах, а не количество его элементов, и при итерации по массиву мы обращаемся к несуществующим элементам и, следовательно, при вычислении суммы мы получаем несогласованные выходные данные.

3 . Почему указан неправильный размер ? Как можно было исправить код, используя *sizeof* ?

sizeof вернёт размер всего массива в байтах. Можно (и нужно) разделить размер всего массива на размер одного элемента. *sizeof(a)/sizeof(int)*

Задание 4

1 . Какая цель является частью правила, удаляющего все скомпилированные программы ?

clean:

*-rm -rf core *.o *~ "#"*"#" Makefile.bak $(BINARIES) *.dSYM*

2 . Какая цель является частью правила, которое создаёт все скомпилированные программы ?

all: $(BINARIES)

3 . Какой компилятор сейчас используется ?

GNU

4 . Какой стандарт Си мы используем в настоящее время ?

C11

5 . Как мы можем ссылаться на переменную “FOO” в Make-файле ?

$(FOO)

6 . Для чего используется переменная CFLAGS ?

В переменной CFLAGS лежат флаги, которые передаются компилятору для компиляции.

7 . Какая строка создаёт программу bit-ops из её объектных файлов ?

$(BIT_OPS_PROG): $(BIT_OPS_OBJS)

$(CC) $(CFLAGS) -g -o $(BIT_OPS_PROG) $(BIT_OPS_OBJS) $(LDFLAGS)

<u>Задание 5</u>

```
C:\Users\User\Desktop\Other\HomeWork\Programming\3_semester\LR6>make bit_ops
MAKE Version 5.41  Copyright (c) 1987, 2014 Embarcadero Technologies, Inc.
        g++ -g -Wall -g -o bit_ops bit_ops.o test_bit_ops.o

C:\Users\User\Desktop\Other\HomeWork\Programming\3_semester\LR6>bit_ops

Testing get_bit()

get_bit(0x4e, 0): 0x0, Correct
get_bit(0x4e, 1): 0x1, Correct
get_bit(0x4e, 5): 0x0, Correct
get_bit(0x1b, 3): 0x1, Correct
get_bit(0x1b, 2): 0x0, Correct
get_bit(0x1b, 9): 0x0, Correct

╨ó╨╡╤ü╤é╨╕╤Ç╨╛╨▓╨░╨╜╨╕╨╡ set_bit()

get_bit(0x4e, 0): 0x0, Correct
set_bit(0x4e, 2, 0): 0x4a, Correct
set_bit(0x6d, 0, 0): 0x6c, Correct
set_bit(0x4e, 2, 1): 0x4e, Correct
set_bit(0x6d, 0, 1): 0x6d, Correct
set_bit(0x4e, 9, 0): 0x4e, Correct
set_bit(0x6d, 4, 0): 0x6d, Correct
set_bit(0x4e, 9, 1): 0x24e, Correct
set_bit(0x6d, 7, 1): 0xed, Correct

╨ó╨╡╤ü╤é╨╕╤Ç╨╛╨▓╨░╨╜╨╕╨╡ flip_bit()

flip_bit(0x4e, 0): 0x4f, Correct
flip_bit(0x4e, 1): 0x4c, Correct
flip_bit(0x4e, 2): 0x4a, Correct
flip_bit(0x4e, 5): 0x6e, Correct
flip_bit(0x4e, 9): 0x24e, Correct
```

Пример работы с GNUDeBugger (GDB):

```
C:\Users\User\Desktop\Other\HomeWork\Programming\3_semester\LR6>g++ -g -o hello hello.cpp

C:\Users\User\Desktop\Other\HomeWork\Programming\3_semester\LR6>gdb hello
C:\MinGW\bin\gdb.exe: warning: Couldn't determine a path for the index cache directory.
GNU gdb (GDB) 8.3.1
Copyright (C) 2019 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-w64-mingw32".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://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 hello...
(gdb) b main
Breakpoint 1 at 0x401564: file hello.cpp, line 4.
(gdb) run
Starting program: C:\Users\User\Desktop\Other\HomeWork\Programming\3_semester\LR6\hello.exe
[New Thread 14552.0x74c]
warning: Can not parse XML library list; XML support was disabled at compile time

Thread 1 hit Breakpoint 1, main (argc=1, argv=0xc614d0) at hello.cpp:4
4           int count = 0;
(gdb) step
5           int *p = &count;
```