ELECTRICAL & COMPUTER ENGINEERING
TEXAS A&M UNIVERSITY

# PA0: Environment Setup, AddressSanitizer and GDB

CSCE 313 – 512

Due Date: August 28, 2020

**Debugging with DGB:**

After setting up the system and compiling the buggy code, I fixed the errors by adding the following code into the blanks:

Blank A:

```
#include <vector>

using namespace std;
```

Blank B:

```
public:
```

After filling in the blankes in the code, I corrected the statements in likes 15, 16, 21, 28, and 29 to make a member variable of an object accesed through a pointer (ex. ret += ptr->val).

After this, I went on to fix the runtime errors. I ran my program as usual and launched it under gdb. The printed variable names or line numbers to locate the errors were in some internal address format, so I compiled my code with the -g option. The following error was caught on line 17:

```
Program received signal SIGSEGV, Segmentation fault.
0x00005555555552f7 in create_LL (
    mylist=std::vector of length 3, capacity 3 = {...}, node_num=3)
    at buggy.cpp:17
17              mylist[i]->val = i;
```

After using "backtrace", I found where the segmentation fault was occuring.

```
(gdb) backtrace
#0  0x00005555555552f7 in create_LL (
    mylist=std::vector of length 3, capacity 3 = {...}, node_num=3)
    at buggy.cpp:17
#1  0x000055555555540a in main (argc=1, argv=0x7fffffffe158) at buggy.cpp:40
```

After setting the breakpoint and running the program from the beginning, I printed the contents of mylist.

```
(gdb) print*(mylist._M_impl._M_start)@mylist.size()
$1 = {0x0, 0x0, 0x0}
```

To fix the segmentation error, I filled in Blank C.

Blank C:

```
mylist[i] = new node;
```

After this, I got another segmentation fault in the sum_LL function. This was due to going "out of range" in mylist on the last itteration. I fixed this fault by itterating the loop till the condition i < node_num – 1 and got the following output:

```
(gdb) run
Starting program: /home/osboxes/Documents/CSCE313/PA/PA0/buggy
The sum of nodes in LL is 3
[Inferior 1 (process 21098) exited normally]
```

Next, I filled in Blank D to free the dynamically allocated memory from the heap for elements of mylist to avoid memory leaks.

Blank D:

```
for (int i=o, I < NODE_NUM; i++){
        delete mylist[i];
        mylist[i] = nullptr;
}
```

**Debugging with AddressSanitizes:**

After setting up the system and compiling the buggy code, I fixed the errors by adding the following code into the blanks:

Blank A:

```
#include <vector>
using namespace std;
```

Blank B:

```
public:
```

After filling in the blankes in the code, I corrected the statements in likes 15, 16, 21, 28, and 29 to make a member variable of an object accesed through a pointer (ex. ret += ptr->val).

After running the program, I obtained the following fault:

```
osboxes@osboxes:~/Documents/CSCE313/PA/PA0$ ./buggy_san
AddressSanitizer:DEADLYSIGNAL
=================================================================
==21227==ERROR: AddressSanitizer: SEGV on unknown address 0x000000000000 (pc 0
x55a85f5945a7 bp 0x7ffe8c1a3200 sp 0x7ffe8c1a3160 T0)
==21227==The signal is caused by a WRITE memory access.
==21227==Hint: address points to the zero page.
    #0 0x55a85f5945a6 in create_LL(std::vector<node*, std::allocator<node*> >&
, int) /home/osboxes/Documents/CSCE313/PA/PA0/buggy_san.cpp:16
    #1 0x55a85f594898 in main /home/osboxes/Documents/CSCE313/PA/PA0/buggy_san
.cpp:39
    #2 0x7ffb3b4950b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x
270b2)
    #3 0x55a85f59436d in _start (/home/osboxes/Documents/CSCE313/PA/PA0/buggy_
san+0x236d)

AddressSanitizer can not provide additional info.
SUMMARY: AddressSanitizer: SEGV /home/osboxes/Documents/CSCE313/PA/PA0/buggy_s
an.cpp:16 in create_LL(std::vector<node*, std::allocator<node*> >&, int)
osboxes@osboxes:~/Documents/CSCE313/PA/PA0$
```

Thus, I filled in Blank C:

Blank C:

```
mylist[i] = new node;
```

I received the following output before I fixed the “out of range” error in mylist on the last itteration. I fixed this fault by itterating the loop till the condition i < node_num – 1.

```
  0x0c067fff7fd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c067fff7fe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c067fff7ff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x0c067fff8000: fa fa 00 00 00[fa]fa fa fa fa fa fa fa fa fa fa
  0x0c067fff8010: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff8020: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff8030: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff8040: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c067fff8050: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
  Partially addressable: 01 02 03 04 05 06 07
  Heap left redzone:       fa
  Freed heap region:       fd
  Stack left redzone:      f1
  Stack mid redzone:       f2
  Stack right redzone:     f3
  Stack after return:      f5
  Stack use after scope:   f8
  Global redzone:          f9
  Global init order:       f6
  Poisoned by user:        f7
  Container overflow:      fc
  Array cookie:            ac
  Intra object redzone:    bb
  ASan internal:           fe
  Left alloca redzone:     ca
  Right alloca redzone:    cb
  Shadow gap:              cc
==21211==ABORTING
osboxes@osboxes:~/Documents/CSCE313/PA/PA0$
```

After fixing that error, I ran my code again and recived an erorr that detected memory leaks.

```
==21265==ERROR: LeakSanitizer: detected memory leaks

Direct leak of 16 byte(s) in 1 object(s) allocated from:
    #0 0x7fb28fb68947 in operator new(unsigned long) (/lib/x86_64-linux-gnu/libasan.so.5+0
x10f947)
    #1 0x556d96344559 in create_LL(std::vector<node*, std::allocator<node*> >&, int) /home
/osboxes/Documents/CSCE313/PA/PA0/buggy_san.cpp:15
    #2 0x556d963448ea in main /home/osboxes/Documents/CSCE313/PA/PA0/buggy_san.cpp:39
    #3 0x7fb28f6920b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x270b2)

Indirect leak of 32 byte(s) in 2 object(s) allocated from:
    #0 0x7fb28fb68947 in operator new(unsigned long) (/lib/x86_64-linux-gnu/libasan.so.5+0
x10f947)
    #1 0x556d96344559 in create_LL(std::vector<node*, std::allocator<node*> >&, int) /home
/osboxes/Documents/CSCE313/PA/PA0/buggy_san.cpp:15
    #2 0x556d963448ea in main /home/osboxes/Documents/CSCE313/PA/PA0/buggy_san.cpp:39
    #3 0x7fb28f6920b2 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x270b2)

SUMMARY: AddressSanitizer: 48 byte(s) leaked in 3 allocation(s).
```

Thus, I addressed the deletion of dynamically allocated memory by filling in Blank D:

<u>Blank D:</u>

```
for (int i=o, I < NODE_NUM; i++){
        delete mylist[i];
        mylist[i] = nullptr;}
```

The output of the code after fixing all of the errors is as follows:

```
osboxes@osboxes:~/Documents/CSCE313/PA/PA0$ g++ buggy_san.cpp -g -fsanitize=address -o bug
gy_san
osboxes@osboxes:~/Documents/CSCE313/PA/PA0$ ./buggy_san
The sum of nodes in LL is 3
osboxes@osboxes:~/Documents/CSCE313/PA/PA0$
```

**IDE:**

The following image shows the 10th step of the PA of repeating the process in the steps above.