Sumedh Tadimeti
1/19/2020
Programming Assignment 0 Report

The first task in this assignment was to set up a Ubuntu development environment in order to install and take advantage of GDB and Address Sanitizer. I did this by using Amazon Web Services and its cloud-based IDE called Cloud9 which runs an Ubuntu Instance Virtual Machine. This service come with a Ubuntu Terminal which allowed me to use features such as GDB and Address Sanitizer.

Next, I compiled the buggy.cpp file, and fixed all the compiler errors starting with including the needed header filer (vectors, and namespace std). In Blank A is put #include <vector>. In Blank B I included the work 'public' in order to make the variables accessible. In lines 15, 16, 21, 28 and 29 I changed "." to "->" because they are referring to pointer variables. A screenshot of all the errors before debugging in below:

```
Running /home/ubuntu/environment/PA0/buggy.cpp
/home/ubuntu/environment/PA0/buggy.cpp:9:16: error: variable or field 'create_LL' declared void
 void create_LL(vector<node*>& mylist, int node_num){
                ^~~~~~
/home/ubuntu/environment/PA0/buggy.cpp:9:16: error: 'vector' was not declared in this scope
/home/ubuntu/environment/PA0/buggy.cpp:9:16: note: suggested alternative:
In file included from /usr/include/c++/7/vector:64:0,
                 from /home/ubuntu/environment/PA0/buggy.cpp:2:
/usr/include/c++/7/bits/stl_vector.h:216:11: note:   'std::vector'
     class vector : protected _Vector_base<_Tp, _Alloc>
           ^~~~~~
/home/ubuntu/environment/PA0/buggy.cpp:9:27: error: expected primary-expression before '*' token
 void create_LL(vector<node*>& mylist, int node_num){
                           ^
/home/ubuntu/environment/PA0/buggy.cpp:9:28: error: expected primary-expression before '>' token
 void create_LL(vector<node*>& mylist, int node_num){
                            ^
/home/ubuntu/environment/PA0/buggy.cpp:9:31: error: 'mylist' was not declared in this scope
 void create_LL(vector<node*>& mylist, int node_num){
                               ^~~~~~
/home/ubuntu/environment/PA0/buggy.cpp:9:31: note: suggested alternative: 'va_list'
 void create_LL(vector<node*>& mylist, int node_num){
                               ^~~~~~
                               va_list
/home/ubuntu/environment/PA0/buggy.cpp:9:39: error: expected primary-expression before 'int'
 void create_LL(vector<node*>& mylist, int node_num){
                                       ^~~
/home/ubuntu/environment/PA0/buggy.cpp: In function 'int sum_LL(node*)':
/home/ubuntu/environment/PA0/buggy.cpp:28:20: error: request for member 'val' in 'ptr', which is of pointer type 'node*' (maybe yo
u meant to use '->' ?)
         ret += ptr.val; // This is line 28
                    ^~~
/home/ubuntu/environment/PA0/buggy.cpp:29:19: error: request for member 'next' in 'ptr', which is of pointer type 'node*' (maybe y
ou meant to use '->' ?)
         ptr = ptr.next; // This is line 29
                   ^~~~
/home/ubuntu/environment/PA0/buggy.cpp: In function 'int main(int, char**)':
/home/ubuntu/environment/PA0/buggy.cpp:36:5: error: 'vector' was not declared in this scope
     vector<node*> mylist;
     ^~~~~~
/home/ubuntu/environment/PA0/buggy.cpp:36:5: note: suggested alternative:
In file included from /usr/include/c++/7/vector:64:0,
                 from /home/ubuntu/environment/PA0/buggy.cpp:2:
/usr/include/c++/7/bits/stl_vector.h:216:11: note:   'std::vector'
     class vector : protected _Vector_base<_Tp, _Alloc>
           ^~~~~~
/home/ubuntu/environment/PA0/buggy.cpp:36:16: error: expected primary-expression before '*' token
     vector<node*> mylist;
                ^
/home/ubuntu/environment/PA0/buggy.cpp:36:17: error: expected primary-expression before '>' token
     vector<node*> mylist;
                 ^
/home/ubuntu/environment/PA0/buggy.cpp:36:19: error: 'mylist' was not declared in this scope
     vector<node*> mylist;
                   ^~~~~~
/home/ubuntu/environment/PA0/buggy.cpp:36:19: note: suggested alternative: 'va_list'
     vector<node*> mylist;
                   ^~~~~~
                   va_list
/home/ubuntu/environment/PA0/buggy.cpp:38:5: error: 'create_LL' was not declared in this scope
     create_LL(mylist, NODE_NUM);
     ^~~~~~~~~
/home/ubuntu/environment/PA0/buggy.cpp:40:5: error: 'cout' was not declared in this scope
     cout << "The sum of nodes in LL is " << ret << endl;
     ^~~~
/home/ubuntu/environment/PA0/buggy.cpp:40:5: note: suggested alternative:
In file included from /home/ubuntu/environment/PA0/buggy.cpp:1:0:
/usr/include/c++/7/iostream:61:18: note:   'std::cout'
   extern ostream cout;  /// Linked to standard output
                  ^~~~
/home/ubuntu/environment/PA0/buggy.cpp:40:52: error: 'endl' was not declared in this scope
```

Sumedh Tadimeti
1/19/2020
Programming Assignment 0 Report

After fixing the runtime error, I re-compiled the code with the "-g" flag to include the symbol table and to able to run GDB. I did not use any optimizations so that the job of AddressSanitizer as a memory-error-catcher is easier.

After compiling I started GDB and ran the program. The GDB stopped after hitting a segmentation fault. As a result, I used backtrace to see the last place the program ran before stopping. A screenshot is below:

```
ubuntu:~/environment/PA0 $ g++ -g buggy.cpp
ubuntu:~/environment/PA0 $ gdb a.out
GNU gdb (Ubuntu 8.1-0ubuntu3.2) 8.1.0.20180409-git
Copyright (C) 2018 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-linux-gnu".
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/>.
---Type <return> to continue, or q <return> to quit---
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from a.out...done.
(gdb) r
Starting program: /home/ubuntu/environment/PA0/a.out

Program received signal SIGSEGV, Segmentation fault.
0x0000555555554bb4 in create_LL (mylist=std::vector of length 3, capacity 3 = {...}, node_num=3) at buggy.cpp:18
18              mylist[i]->val = i; // This is 15
(gdb)
```

After that, I set a breakpoint at line 15 and re-ran the code from the beginning. After that, I printed what is stored in mylist[i] using the GDB and not cout nor printf. A screenshot is below:

```
 1  #include <iostream>
 2  #include <vector> //Blank A
 3  using namespace std;
 4
 5  class node {
 6   public://Blank B
 7      int val;
 8      node* next;
 9  };
10
11  void create_LL(vector<node*>& mylist, int node_num){
12      mylist.assign(node_num, NULL);
13
14      //create a set of nodes
15      for (int i = 0; i < node_num; i++) {
16          //Blank C
17         // mylist[i]= new node(); // you have to create the new node before you can use it
18          mylist[i]->val = i; // This is 15
19          mylist[i]->next = NULL; // This is 16
20      }
21
22      //create a linked list
23      for (int i = 0; i < node_num-1; i++) {
24          mylist[i]->next = mylist[i+1]; // This is line 21
25      }
26  }
27
```

Sumedh Tadimeti
1/19/2020
Programming Assignment 0 Report

```
(gdb) print mylist[i]
$1 = (node *) 0x0
(gdb)
```

After this, the solution to fixing the first segmentation fault lies in filing in Blank C. I wrote a value stored in mylist[i]. In Blank C I wrote mylist[i]= new node();

After fixing the first segmentation fault in Blank C, I re-ran the code and came across another segmentation fault. I used GDB and found that this error is in the second for loop in create_LL function. I fixed this by changing node_num in the for loop to node_num-1 because as of right now it is pointing to a random memory address.

After this, to avoid memory loss, I created a for loop to free the nodes that were dynamically created earlier. A screenshot is below:

```
    //Step4: delete nodes
    //Blank D
    for (int i=0; i<NODE_NUM;i++){ // This deletes the nodes
        delete mylist[i];
    }
}
```

A screenshot is provided below for the final output:

```
Starting program: /home/ubuntu/environment/PA0/a.out
The sum of nodes in LL is 3
[Inferior 1 (process 16179) exited normally]
(gdb)
```

This is the end of debugging with GDB. From now on all debugging will happen through Address Sanatizer.

Sumedh Tadimeti
1/19/2020
Programming Assignment 0 Report

Firstly, I fixed all the compiler and runtime errors that were present before shifting my focus to the segmentation faults. I compiled the file using the "-fsanitize=address" flag and ran into the first Segmentation Fault. A screen shot is below:

```
ubuntu:~/environment/PA0 $ g++ -g -fsanitize=address buggyASAN.cpp
ubuntu:~/environment/PA0 $ ./a.out
ASAN:DEADLYSIGNAL
=================================================================
==9391==ERROR: AddressSanitizer: SEGV on unknown address 0x000000000000 (pc 0x55dbe05c3394 bp 0x7ffd47cded90 sp 0x7ffd47cdecf0 T0)
==9391==The signal is caused by a WRITE memory access.
==9391==Hint: address points to the zero page.
    #0 0x55dbe05c3393 in create_LL(std::vector<node*, std::allocator<node*> >&, int) /home/ubuntu/environment/PA0/buggyASAN.cpp:18
    #1 0x55dbe05c3668 in main /home/ubuntu/environment/PA0/buggyASAN.cpp:41
    #2 0x7f20c8950b96 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x21b96)
    #3 0x55dbe05c3139 in _start (/home/ubuntu/environment/PA0/a.out+0x1139)

AddressSanitizer can not provide additional info.
SUMMARY: AddressSanitizer: SEGV /home/ubuntu/environment/PA0/buggyASAN.cpp:18 in create_LL(std::vector<node*, std::allocator<node*
> >&, int)
==9391==ABORTING
ubuntu:~/environment/PA0 $
```

I fixed this Segmentation Fault, by filling in Blank C with mylist[i]= new node(); just like I did before. I re-ran the file and encountered the second Segmentation Fault. A screenshot of this is below:

Sumedh Tadimeti
1/19/2020
Programming Assignment 0 Report

```
ubuntu:~/environment/PA0 $ g++ -g -fsanitize=address buggyASAN.cpp
ubuntu:~/environment/PA0 $ ./a.out
=================================================================
==9771==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x603000000028 at pc 0x564933bce552 bp 0x7fff27a8c3f0 sp 0x7fff27
a8c3e0
READ of size 8 at 0x603000000028 thread T0
    #0 0x564933bce551 in create_LL(std::vector<node*, std::allocator<node*> >&, int) /home/ubuntu/environment/PA0/buggyASAN.cpp:25
    #1 0x564933bce714 in main /home/ubuntu/environment/PA0/buggyASAN.cpp:42
    #2 0x7f2a2f984b96 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x21b96)
    #3 0x564933bce139 in _start (/home/ubuntu/environment/PA0/a.out+0x1139)

0x603000000028 is located 0 bytes to the right of 24-byte region [0x603000000010,0x603000000028)
allocated by thread T0 here:
    #0 0x7f2a303d5458 in operator new(unsigned long) (/usr/lib/x86_64-linux-gnu/libasan.so.4+0xe0458)
    #1 0x564933bcfbcc in __gnu_cxx::new_allocator<node*>::allocate(unsigned long, void const*) /usr/include/c++/7/ext/new_allocato
r.h:111
    #2 0x564933bcfb84 in std::allocator_traits<std::allocator<node*> >::allocate(std::allocator<node*>&, unsigned long) /usr/inclu
de/c++/7/bits/alloc_traits.h:436
    #3 0x564933bcfb43 in std::_Vector_base<node*, std::allocator<node*> >::_M_allocate(unsigned long) /usr/include/c++/7/bits/stl_
vector.h:172
    #4 0x564933bcf9e4 in std::_Vector_base<node*, std::allocator<node*> >::_M_create_storage(unsigned long) /usr/include/c++/7/bit
s/stl_vector.h:187
    #5 0x564933bcf408 in std::_Vector_base<node*, std::allocator<node*> >::_Vector_base(unsigned long, std::allocator<node*> const
&) /usr/include/c++/7/bits/stl_vector.h:138
    #6 0x564933bcee33 in std::vector<node*, std::allocator<node*> >::vector(unsigned long, node* const&, std::allocator<node*> con
st&) /usr/include/c++/7/bits/stl_vector.h:297
    #7 0x564933bceab4 in std::vector<node*, std::allocator<node*> >::_M_fill_assign(unsigned long, node* const&) /usr/include/c++/
7/bits/vector.tcc:242
    #8 0x564933bce8e0 in std::vector<node*, std::allocator<node*> >::assign(unsigned long, node* const&) /usr/include/c++/7/bits/s
tl_vector.h:502
    #9 0x564933bce2f2 in create_LL(std::vector<node*, std::allocator<node*> >&, int) /home/ubuntu/environment/PA0/buggyASAN.cpp:13
    #10 0x564933bce714 in main /home/ubuntu/environment/PA0/buggyASAN.cpp:42
    #11 0x7f2a2f984b96 in __libc_start_main (/lib/x86_64-linux-gnu/libc.so.6+0x21b96)

SUMMARY: AddressSanitizer: heap-buffer-overflow /home/ubuntu/environment/PA0/buggyASAN.cpp:25 in create_LL(std::vector<node*, std:
:allocator<node*> >&, int)
Shadow bytes around the buggy address:
  0x0c067fff7fb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c067fff7fc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  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
==9771==ABORTING
```

I solved this segmentation fault like I did earlier by changing node_num to node_num-1 like I did earlier.

Sumedh Tadimeti
1/19/2020
Programming Assignment 0 Report

After this, I got a memory leak warning from AddressSanitizer. I fixed this by creating a for loop that frees all the memory that I allocated, just like I did before using GDB.

A screenshot of the corrected code is below:

```
#include <iostream>
#include <vector> //Blank A
using namespace std;

class node {
 public://Blank B
    int val;
    node* next;
};

void create_LL(vector<node*>& mylist, int node_num){
    mylist.assign(node_num, NULL);

    //create a set of nodes
    for (int i = 0; i < node_num; i++) {

       mylist[i]= new node(); // you have to create the new node before you can use it (Blank C)
        mylist[i]->val = i; // This is 15
        mylist[i]->next = NULL; // This is 16
    }

    //create a linked list
    for (int i = 0; i < node_num-1; i++) { //it has to be nod_num-1 becasue it points to a random mem addr
        mylist[i]->next = mylist[i+1]; // This is line 21
    }
}

int sum_LL(node* ptr) {
    int ret = 0;
    while(ptr) {
        ret += ptr->val; // This is line 28
        ptr = ptr->next; // This is line 29
    }
    return ret;
}

int main(int argc, char ** argv){
    const int NODE_NUM = 3;
    vector<node*> mylist;

    create_LL(mylist, NODE_NUM);
    int ret = sum_LL(mylist[0]);
    cout << "The sum of nodes in LL is " << ret << endl;

    //Step4: delete nodes
    //Blank D
    for (int i=0; i<NODE_NUM;i++){ // This deletes the nodes
        delete mylist[i];
    }
}
```