22M0761 - ABHISHEK DIXIT
Lab: Introduction to Debugging Tools
Part A : Debugging with GDB

---

1)
Used command "run" in "gdb" and following error was generated (after compiling pointers.cpp and doing “gdb pointers”) -

Program received signal SIGSEGV, Segmentation fault.
0x0000aaaaaaaa0b0c in main (argc=1, argv=0xffffffff048) at pointers.cpp:13
13 cout << *p << endl;

This tells us that a Segmentation fault is present in line number 13 of the program.

---

2)
The error detected by me using the below commands in "gdb" is in line 16 and 17 where "second_last = last" is happening after "last = next" hence, the same value of variable "next" is getting stored in "last" and "second_last" variable, which should not happen as after every run of loop, second_last should take value of last and last should take value of next.
Steps followed by me - (i) I used “run” and observed errors in printed values. (Not shown below)
(ii) I used “break” at the 1st line of the program.
(iii) I used “run” to run the program and it stopped at the breakpoint.
(iv) I used “step” repeatedly to access the lines of code one by one, and observed discrepancy.
(v) I used “print” to check values of “last” and “second_last” variables after 1 loop completed and found the error.
{ Note - The error can be corrected by swapping line number 16 and 17 of the code }

(gdb) break fibonacci.cpp:1
Breakpoint 1 at 0xaaaaaaaa09e4: file fibonacci.cpp, line 6.
(gdb) run
Starting program: /home/abhishekdixit/Downloads/intro-debug-code/fibonacci
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/aarch64-linux-gnu/libthread_db.so.1".

```
Breakpoint 1, main (argc=1, argv=0xfffffffff048) at fibonacci.cpp:6
6       int n = 10;
(gdb) step
8       int second_last = 1;
(gdb) step
9       int last = 1;
(gdb) step
11      cout << second_last << endl << last << endl;
(gdb) step
1
1
13      for(int i=1; i<=10; i++) {
(gdb) step
14      int next = second_last + last;
(gdb) step
15      cout << next << endl;
(gdb) step
2
16      last = next;
(gdb) step
17      second_last = last;
(gdb) step
13      for(int i=1; i<=10; i++) {
(gdb) print second_last
$1 = 2
(gdb) print last
$2 = 2
```

Part B: Memory Check with Valgrind

After using the given command, following output was generated (shown below) and following issues were found -

1. Line 19: This type of error can occur when we write to or read from a variable which is not declared beforehand.
   Solution - Declare all variables before using them.

2. Line 26: Invalid write can happen when we try to write to a location which is already freed or going beyond the size of allocated data structure.
   Solution - Do not free a memory location if it is needed in future, allocate sufficient space to data structures as per program need.
3. Line 29: Invalid read can happen if we try to read from a memory location which was freed already.
   Solution - If a memory location is to be read from the future, don't free it.
4. Line 35: Invalid free can happen if we try to free a memory location block which has already been freed once.
   Solution - Take care of which memory blocks are being freed.

```
==5774== Syscall param write(buf) points to uninitialised byte(s)
==5774==    at 0x4977C50: write (write.c:26)
==5774==    by 0x1089E3: main (memory_bugs.c:19)
==5774==  Address 0x1ffefffe40 is on thread 1's stack
==5774==  in frame #1, created by main (memory_bugs.c:9)
==5774==
P<?>@<?>==5774== Invalid write of size 1
==5774==    at 0x108A00: main (memory_bugs.c:26)
==5774==  Address 0x4a4a0a0 is 0 bytes inside a block of size 12 free'd
==5774==    at 0x4867AD0: free (in
/usr/libexec/valgrind/vgpreload_memcheck-arm64-linux.so)
==5774==    by 0x1089F7: main (memory_bugs.c:23)
==5774==  Block was alloc'd at
==5774==    at 0x4865058: malloc (in
/usr/libexec/valgrind/vgpreload_memcheck-arm64-linux.so)
==5774==    by 0x1089EB: main (memory_bugs.c:22)
==5774==
==5774== Invalid read of size 1
==5774==    at 0x108A08: main (memory_bugs.c:29)
==5774==  Address 0x4a4a0a0 is 0 bytes inside a block of size 12 free'd
==5774==    at 0x4867AD0: free (in
/usr/libexec/valgrind/vgpreload_memcheck-arm64-linux.so)
==5774==    by 0x1089F7: main (memory_bugs.c:23)
==5774==  Block was alloc'd at
==5774==    at 0x4865058: malloc (in
/usr/libexec/valgrind/vgpreload_memcheck-arm64-linux.so)
==5774==    by 0x1089EB: main (memory_bugs.c:22)
==5774==
```

```
A
==5774== Invalid free() / delete / delete[] / realloc()
==5774==    at 0x4867AD0: free (in
/usr/libexec/valgrind/vgpreload_memcheck-arm64-linux.so)
==5774==    by 0x108A2F: main (memory_bugs.c:35)
==5774==  Address 0x1ffefffe40 is on thread 1's stack
==5774==  in frame #1, created by main (memory_bugs.c:9)
==5774==
==5774==
==5774== HEAP SUMMARY:
==5774==    in use at exit: 80 bytes in 2 blocks
==5774==   total heap usage: 4 allocs, 3 frees, 1,116 bytes allocated
==5774==
==5774== 30 bytes in 1 blocks are definitely lost in loss record 1 of 2
==5774==    at 0x4865058: malloc (in
/usr/libexec/valgrind/vgpreload_memcheck-arm64-linux.so)
==5774==    by 0x1089B7: main (memory_bugs.c:16)
==5774==
==5774== 50 bytes in 1 blocks are definitely lost in loss record 2 of 2
==5774==    at 0x4865058: malloc (in
/usr/libexec/valgrind/vgpreload_memcheck-arm64-linux.so)
==5774==    by 0x108A23: main (memory_bugs.c:32)
==5774==
==5774== LEAK SUMMARY:
==5774==    definitely lost: 80 bytes in 2 blocks
==5774==    indirectly lost: 0 bytes in 0 blocks
==5774==    possibly lost: 0 bytes in 0 blocks
==5774==    still reachable: 0 bytes in 0 blocks
==5774==           suppressed: 0 bytes in 0 blocks
```
