CS450 Operating Systems (Spring 2018)

Josiah Hunt (jhunt5@hawk.iit.edu) (20350987)

Mayank Bansal (mbansal5@hawk.iit.edu) (20392482)

Programming Assignment 3

Part 1 (Memory Leaks)

There were two test programs for this part.

The first one is memTest, and it simply shows the difference between properly freeing memory and not freeing memory. It allocates memory using malloc and populates the memory. Then, a flag variable calle NEAT is used to determine if or if not, the program will correctly free that memory. We ran the program twice, once freeing memory, once without, in order to show the difference. (Execution and results in screencast)

The program ran normally, both with and without freeing the allocated memory. Running the code in the GDB debugger yielded no visible errors. However, using the Valgrind leak-check command on the program revealed the memory leaks, specifying the number of bytes leaked, and how many blocks were left un-freed. It provides the address, and the associated line of code where each memory block was allocated, allowing for easy debugging.

```
--2458-- REDIR: 0x4ebe4f0 (libc.so.6:free) redirected to 0x4c2ed80 (free)
==2458==
==2458== HEAP SUMMARY:
==2458==     in use at exit: 100 bytes in 6 blocks
==2458==   total heap usage: 7 allocs, 1 frees, 1,124 bytes allocated
==2458==
==2458== Searching for pointers to 6 not-freed blocks
==2458== Checked 62,304 bytes
==2458==
==2458== 20 bytes in 1 blocks are definitely lost in loss record 1 of 2
==2458==    at 0x4C2DB8F: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==2458==    by 0x400696: main (memTest.c:11)
==2458==
==2458== 80 bytes in 5 blocks are definitely lost in loss record 2 of 2
==2458==    at 0x4C2DB8F: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==2458==    by 0x400791: main (memTest.c:24)
==2458==
==2458== LEAK SUMMARY:
==2458==    definitely lost: 100 bytes in 6 blocks
==2458==    indirectly lost: 0 bytes in 0 blocks
==2458==      possibly lost: 0 bytes in 0 blocks
==2458==    still reachable: 0 bytes in 0 blocks
==2458==         suppressed: 0 bytes in 0 blocks
==2458==
==2458== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)
==2458== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)
jhunt5@jhunt5-VirtualBox:~/CS450 HW/Hunt_Josiah_PA3$
```

The second program is dataTest. This program does exactly as specified in project assignment, allocating an array of 100 integers, populating it, then freeing it. It then tries to access that memory.

(Execution and results in screencast)

This program also compiled and ran without errors. Using GDB again revealed no errors. Valgrind, however, revealed that, although all memory was feed, there were errors in the memory usage. It summarized the problem, saying that the program was reading from memory that had already been freed, and provided the address and line of code where the action took place. Again, this allows for easy, and efficient debugging.

```
==2441==
==2441== HEAP SUMMARY:
==2441==     in use at exit: 0 bytes in 0 blocks
==2441==   total heap usage: 2 allocs, 2 frees, 1,424 bytes allocated
==2441==
==2441== All heap blocks were freed -- no leaks are possible
==2441==
==2441== ERROR SUMMARY: 100 errors from 1 contexts (suppressed: 0 from 0)
==2441==
==2441== 100 errors in context 1 of 1:
==2441== Invalid read of size 4
==2441==    at 0x400731: main (dataTest.c:20)
==2441==  Address 0x5204040 is 0 bytes inside a block of size 400 free'd
==2441==    at 0x4C2EDEB: free (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==2441==    by 0x400709: main (dataTest.c:17)
==2441==  Block was alloc'd at
==2441==    at 0x4C2DB8F: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==2441==    by 0x40065E: main (dataTest.c:6)
==2441==
==2441== ERROR SUMMARY: 100 errors from 1 contexts (suppressed: 0 from 0)
jhunt5@jhunt5-VirtualBox:~/CS450 HW/Hunt_Josiah_PA3$
```

(Execution and results in screencast)