Christopher Cao
ch282858

<u>Programming Assignment #2</u>

1. First, write a simple program called “null.c” that creates a pointer to an integer, sets it to “NULL”, and then tries to dereference it. Compile this into an executable called “null”. What happens when you run this program?

   A segmentation fault.

2. Next, compile this program with symbol information included (with the “-g” flag). Doing so let’s put more information into the executable, enabling the debugger to access more useful information about variable names and the like. Run the program under the debugger by typing “gdb null” and then, once gdb is running, typing “run”. What does gdb show you?

   gdb shows which line is causing the segmentation fault.

3. Finally, use the “valgrind” tool on this program. We’ll use the “memcheck” tool that is a part of valgrind to analyze what happens. Run this by typing in the following: “valgrind --leak-check=yes null”. What happens when you run this? Can you interpret the output from the tool?

   When we run this, it provides us a list of errors with the program. Address 0x0, which the integer pointer is trying to access, has not been assigned.

4. Write a simple program that allocates memory using “malloc()” but forgets to free it before exiting. What happens when this program runs? Can you use “gdb” to find any problems with it? How about “valgrind” (again with the “--leak-check=yes flag”)?

   The program runs normally. gdb did not find any issues with the program. valgrind was able to detect that we forgot to free 4 bytes of memory.

5. Write a program that creates an array of integers called “data” of size 100 using “malloc”; then, set “data[100]” to zero. What happens when you run this program? What happens when you run this program using “valgrind”? Is the program correct?

   When running the program, nothing abnormal appears to happen. When using valgrind, it warns that there was an error relating to memory. I think the program is incorrect, as we did not allocate memory properly.

6. Create a program that allocates an array of integers (as above), frees them, and then tries to print the value of one of the elements of the array. Does the program run? What happens when you use "valgrind" on it?

   The program runs and prints out a 0. When using valgrind, it says there are an invalid read and write error.

7. Now pass a funny value to free (e.g., a pointer in the middle of the array you allocated above). What happens? Do you need tools to find this type of problem?

   The program crashes and generates an error message saying what is causing the issue. It is detailed enough to where we do not need any other tools to find the problem.