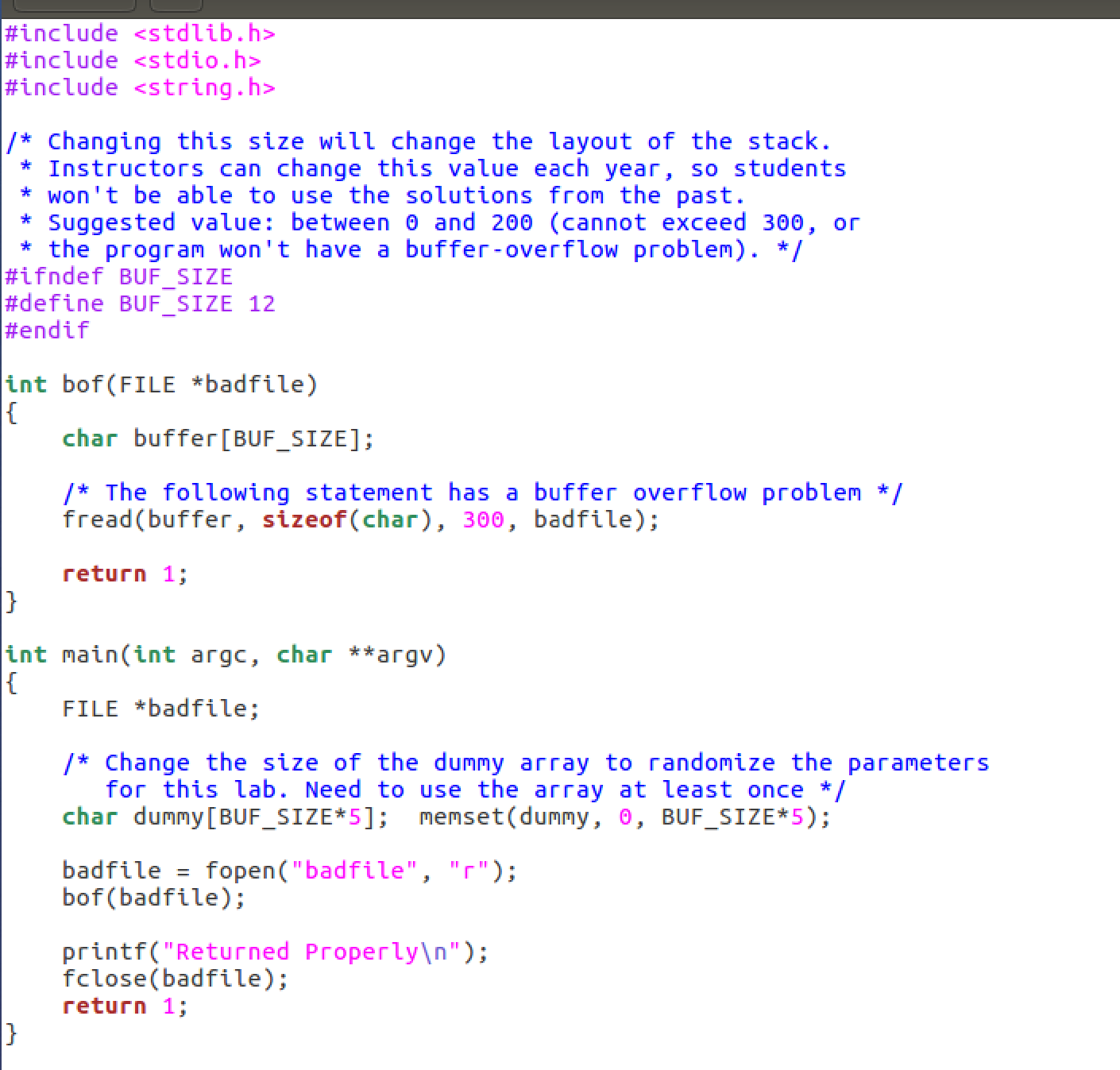
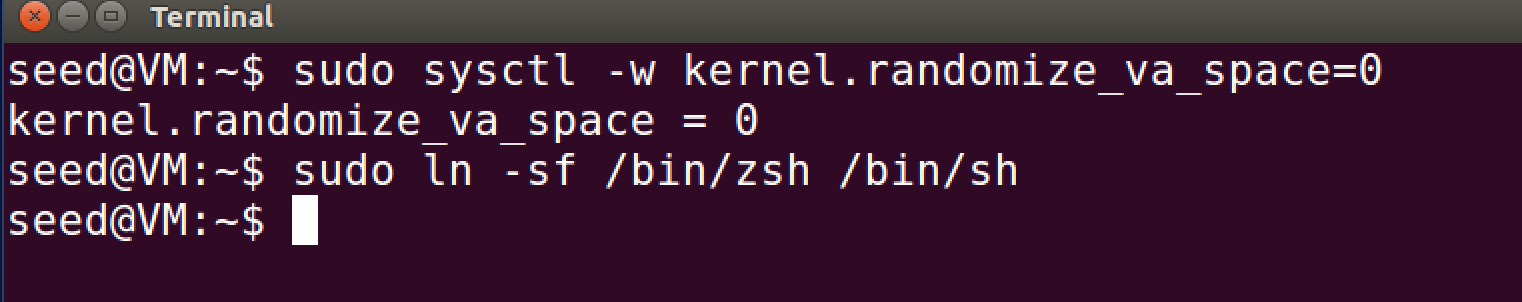
Lab 5

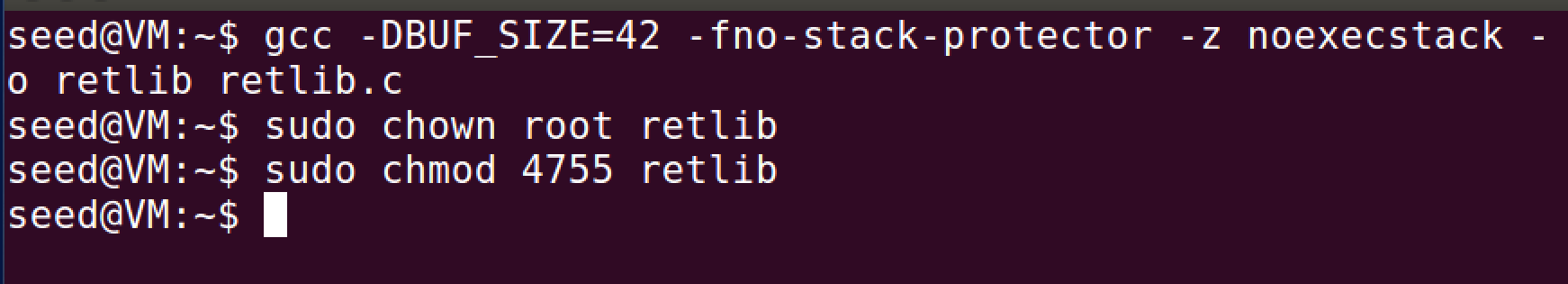
Task 1: Finding out the addresses of libc functions

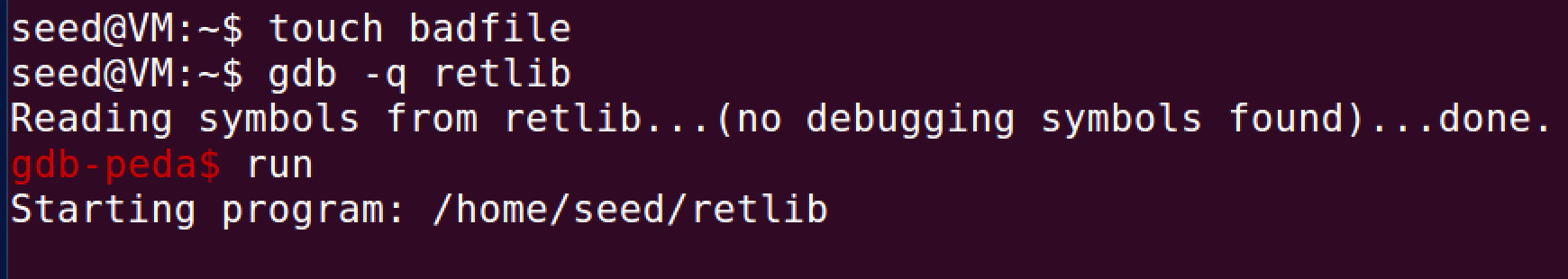
Code Used:

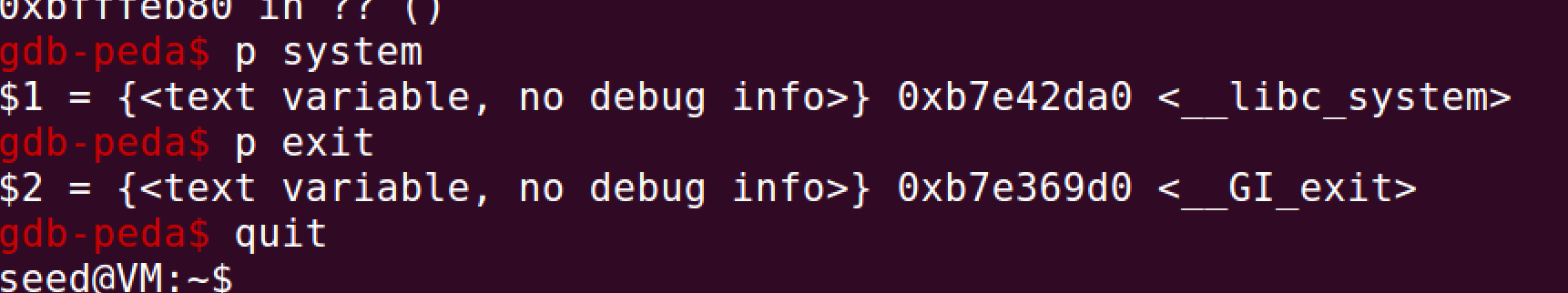


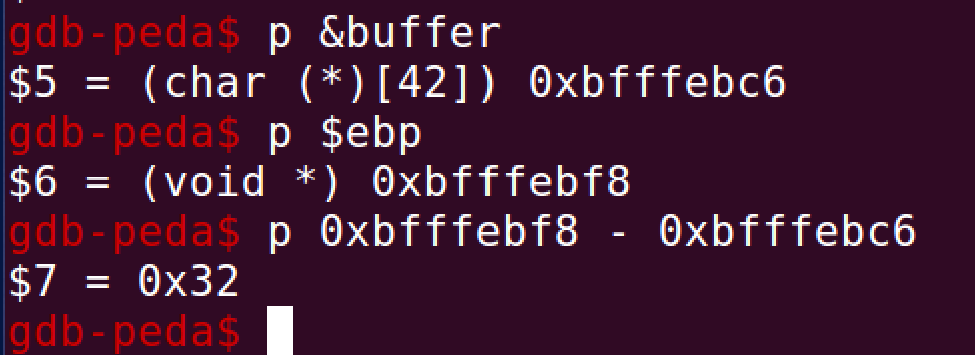
Commands Used:









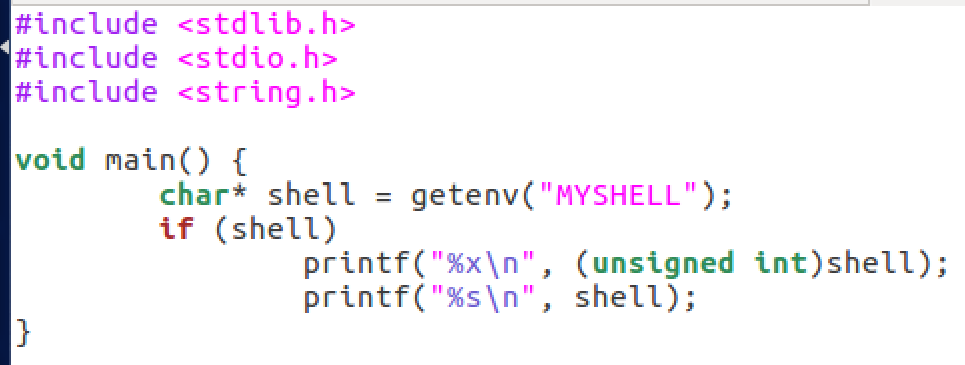
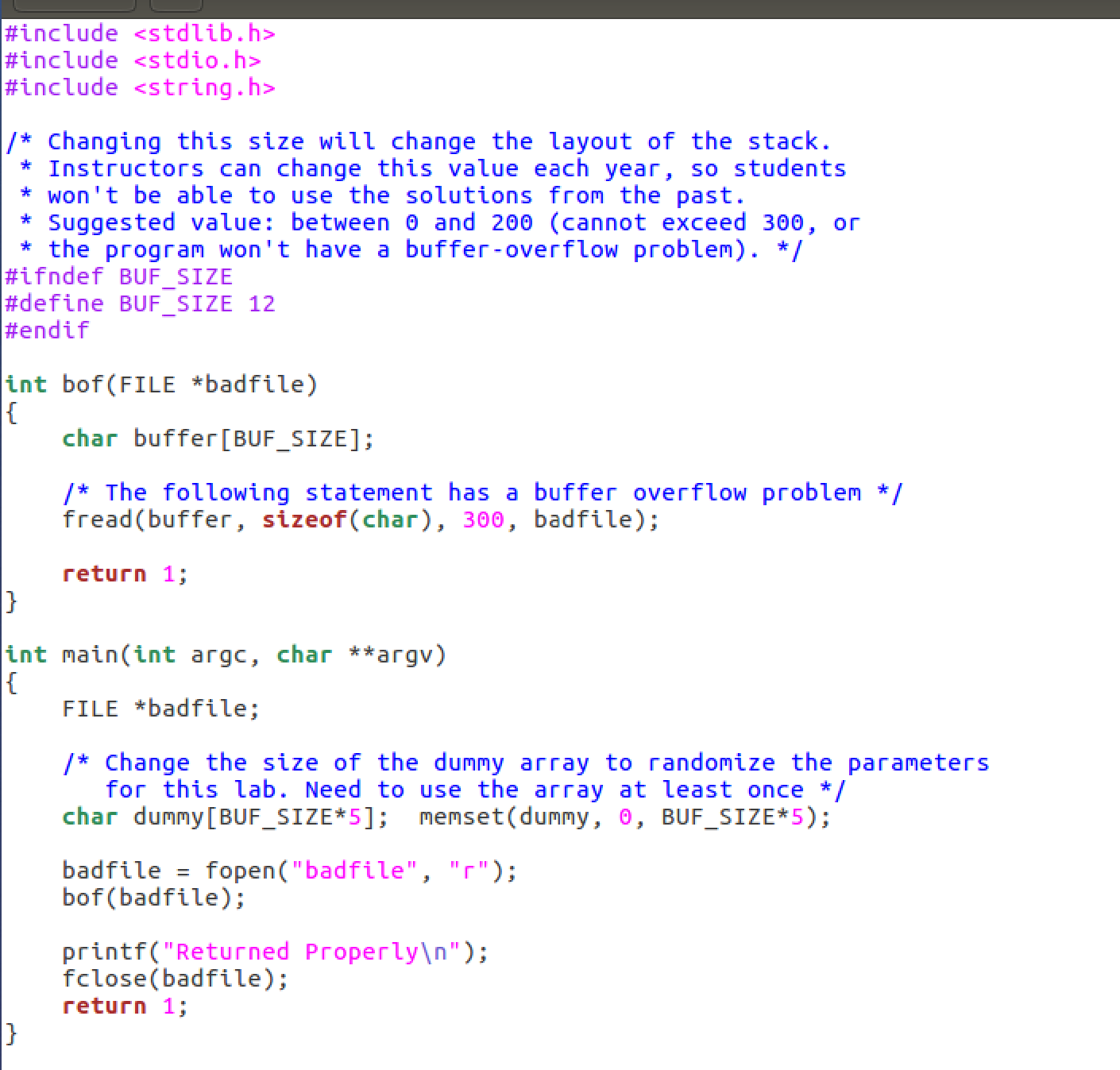


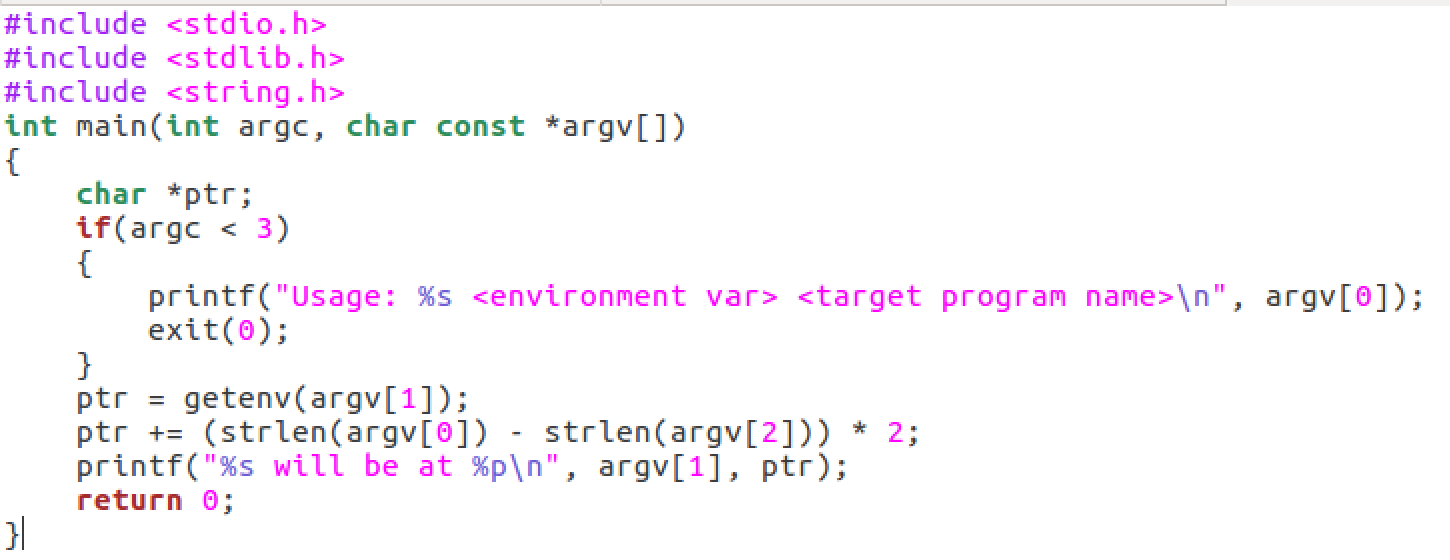
Comments:

First I had to turn off the randomization of the addresses to make sure that the addresses would not change each time I looked for them. I set the value of DBUF\_SIZE to 42 as per the lab instructions. I disabled the countermeasures when I compiled the code and changed the owner and made it into a Set-UID program. I created the badfile and then ran the debugger to obtain the system, exit, and buffer addresses. I had some trouble figuring this part out, but eventually I figured it out.

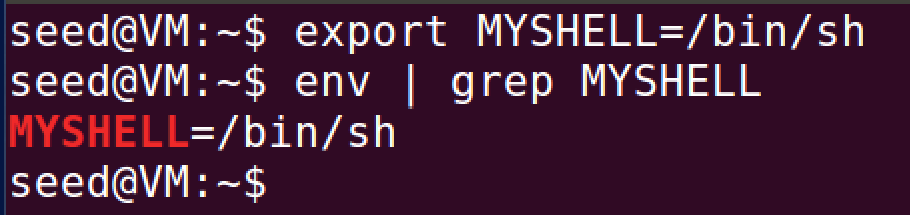
Task 2: Putting Shell String in Memory

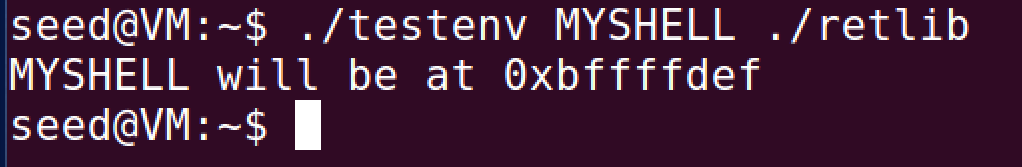
Code Used:





Commands Used:



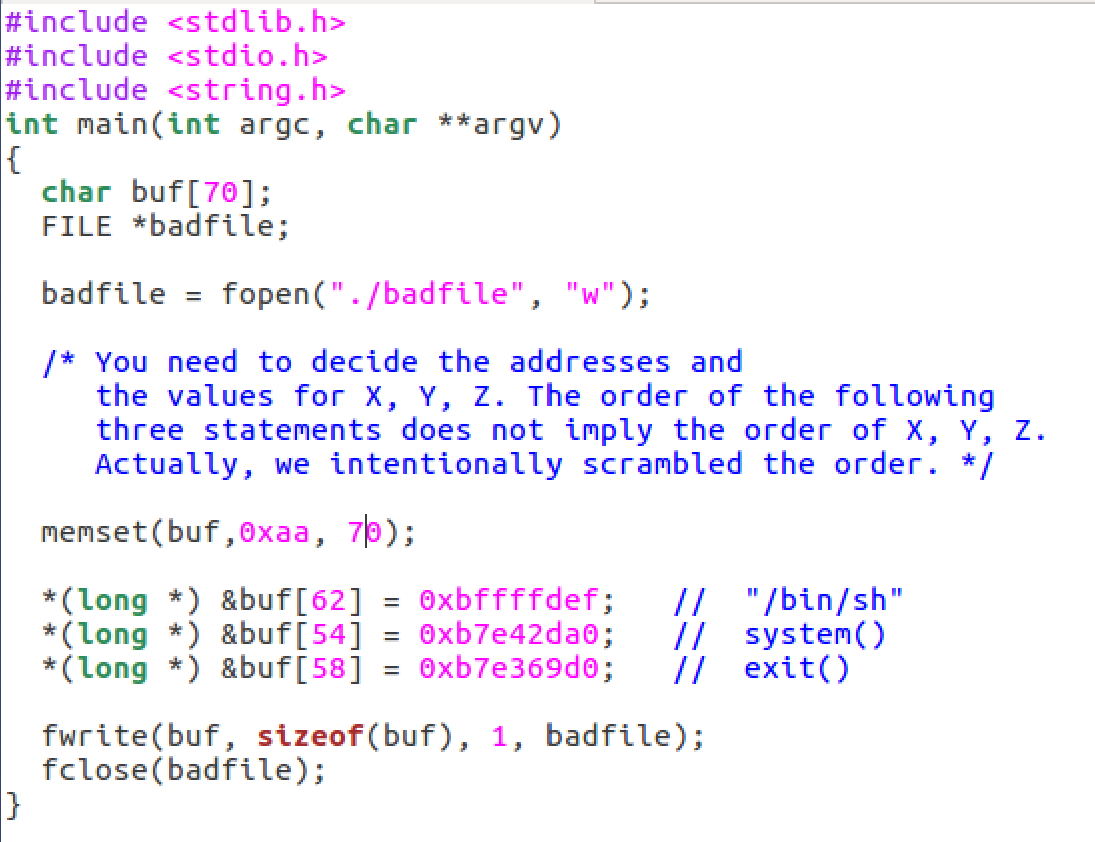


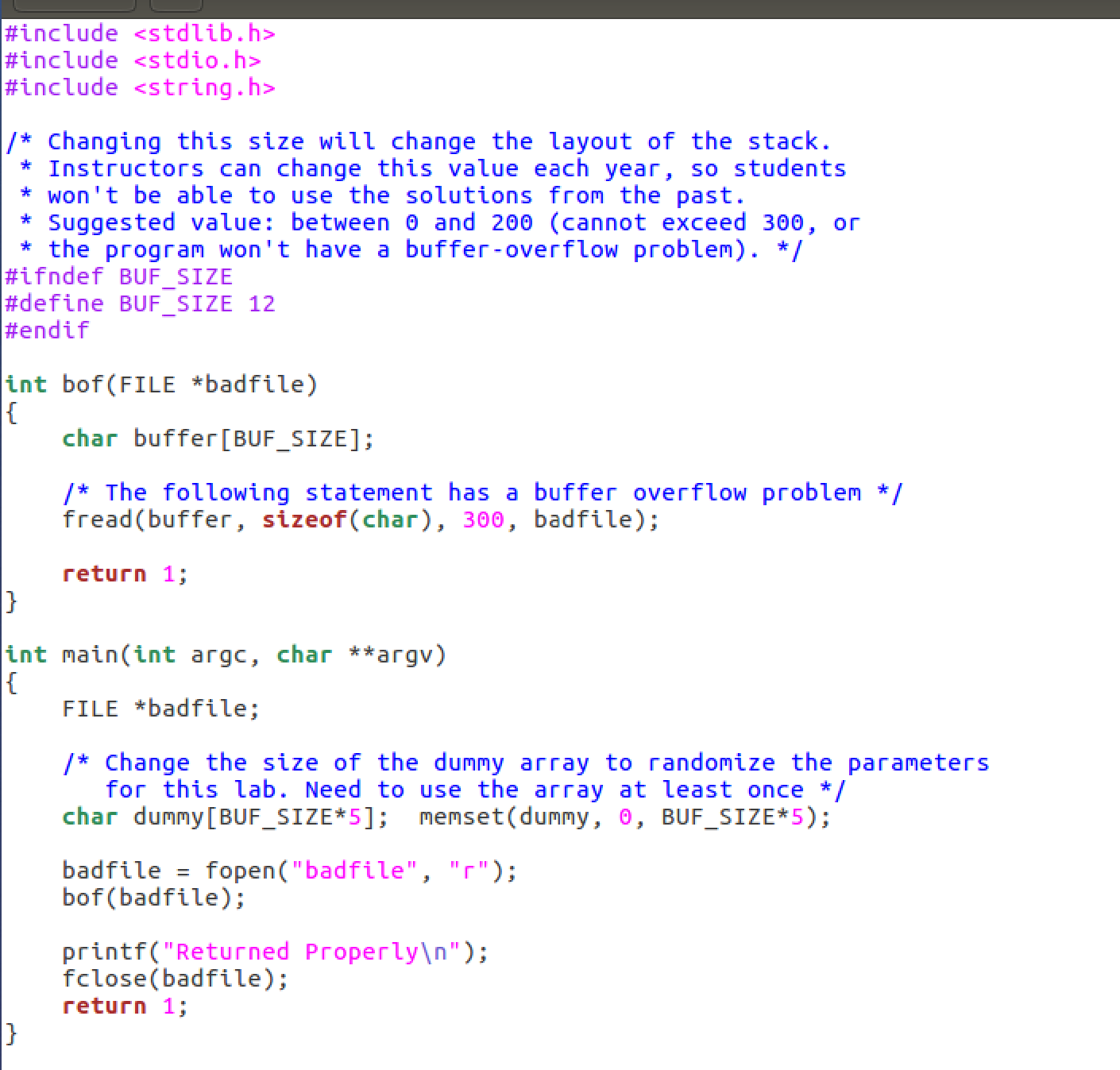
Comments:

For this one I had to find the address of /bin/sh by exporting an environment variable equal to it and made and then ran numerous codes to figure out the address of it just to be sure what I had was correct.

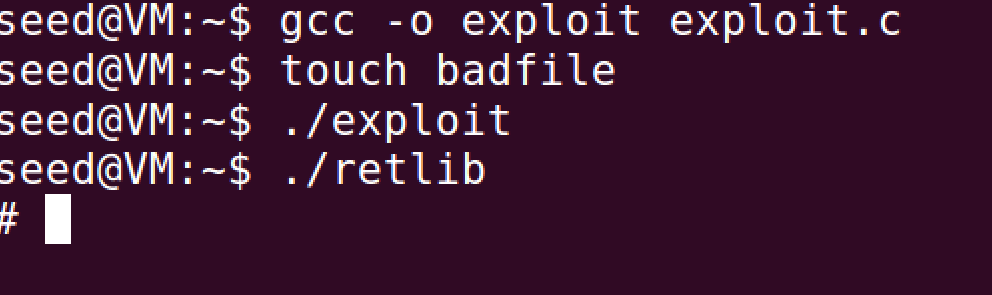
Task 3: Exploiting the buffer-overflow vulnerability

Code Used:





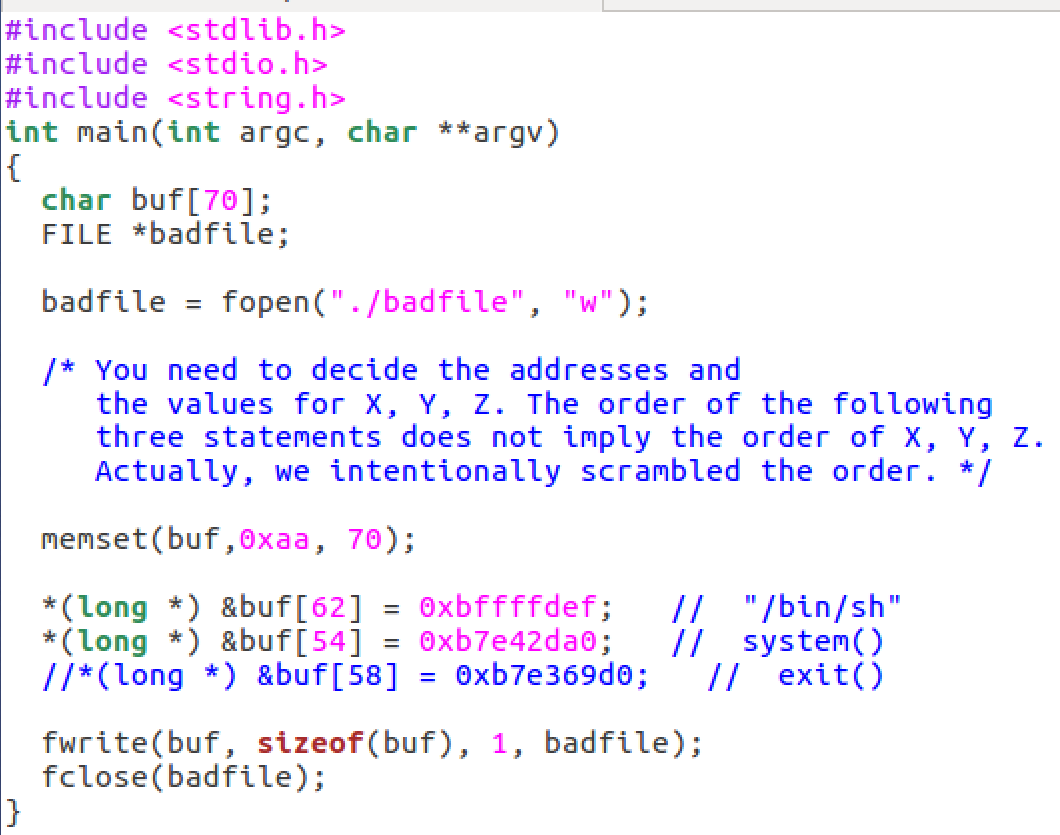
Commands Used:

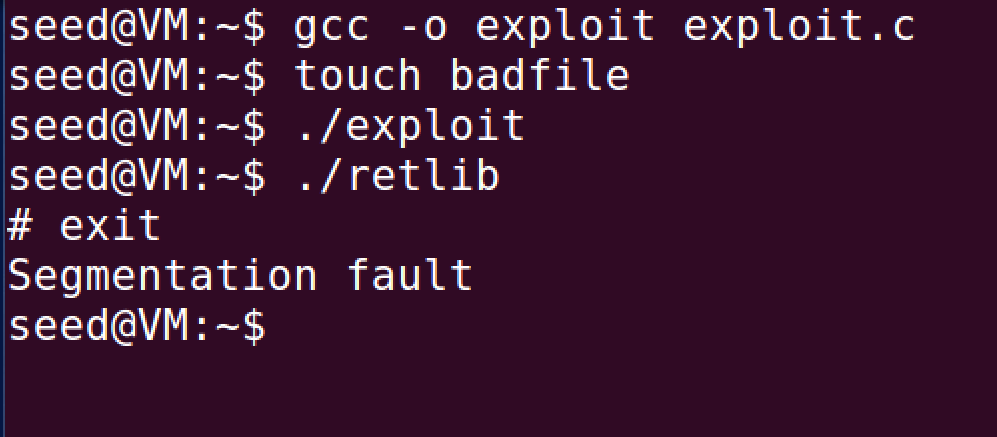


Comments:

The way to find the indexes of the buffer where the address are was tricky but by subtracting the values of the ebp and the buffer then converting it into decimal. It turned out to be 50, the system address is +4 that, exit is +8, and the /bin/sh is +12 that. I had to modify the buf size to fit it all in and 70 seemed to work. Next I had to fill the buf with something so the attack would work and it did. I got the root shell.

Attack variation 1:

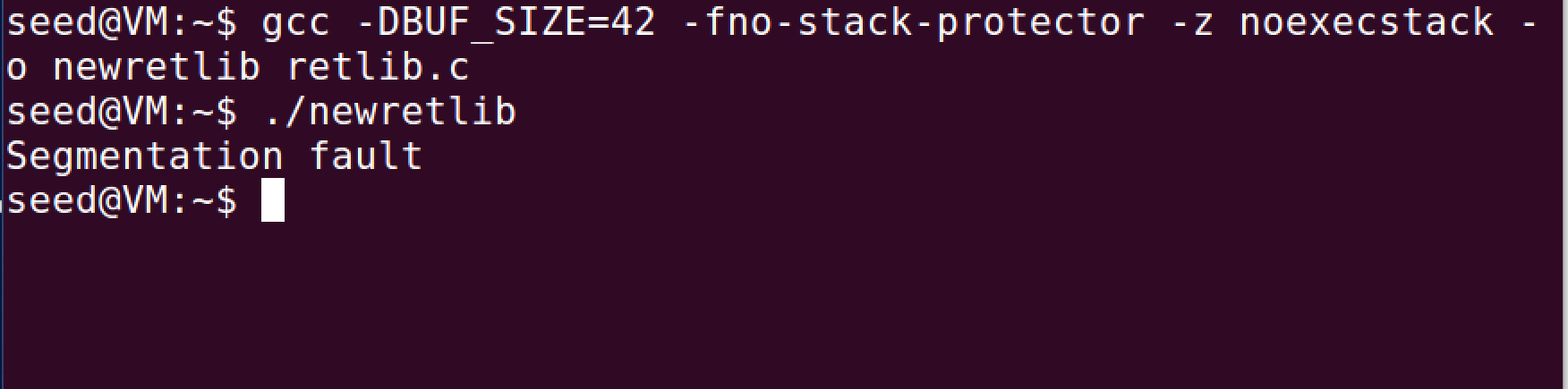




Comments:

The attack was successful but when I tried to run the exit command it gave me a segmentation fault because it doesn’t have access to that memory address since I had to comment it out.

Attack variation 2:

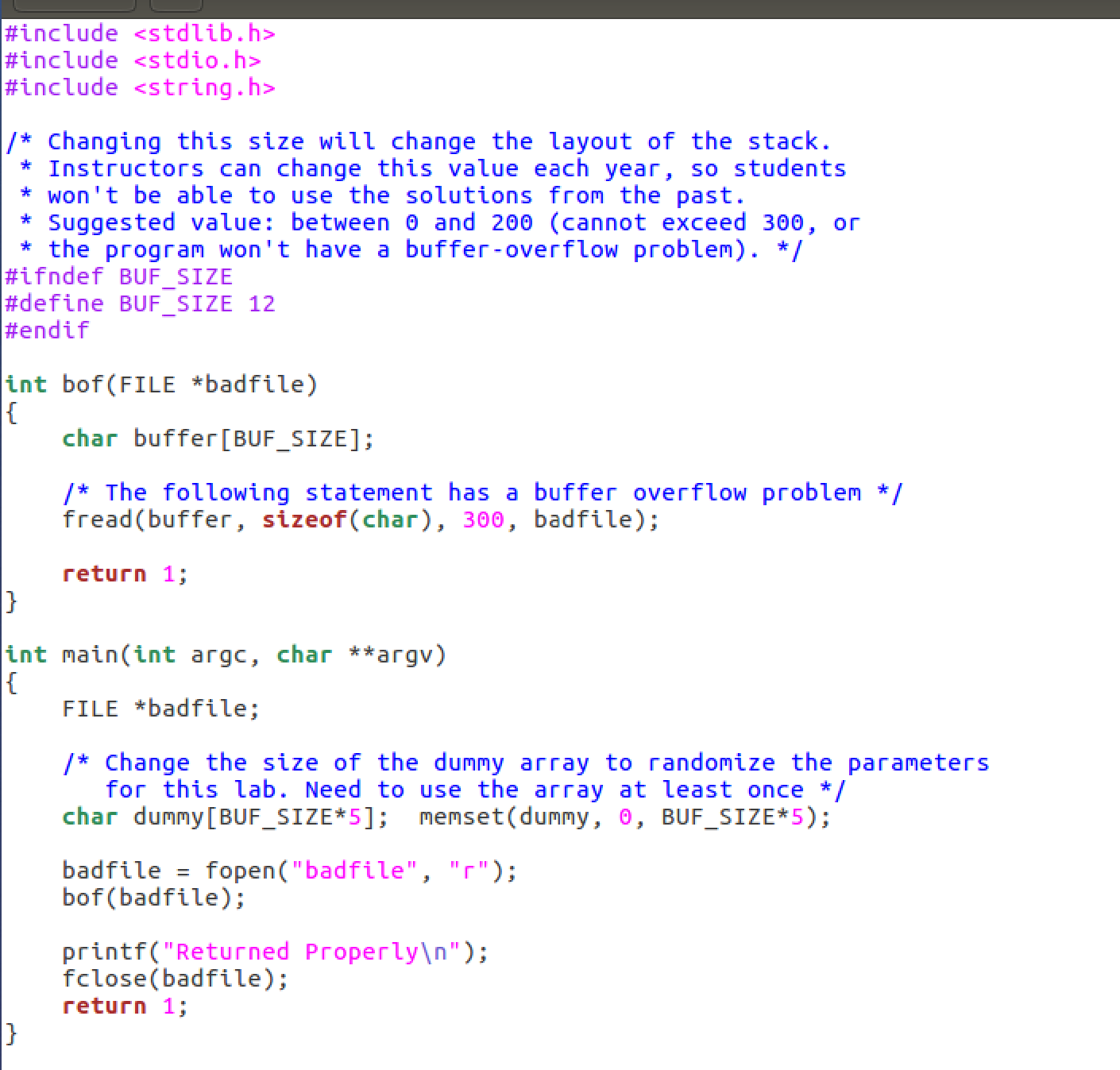


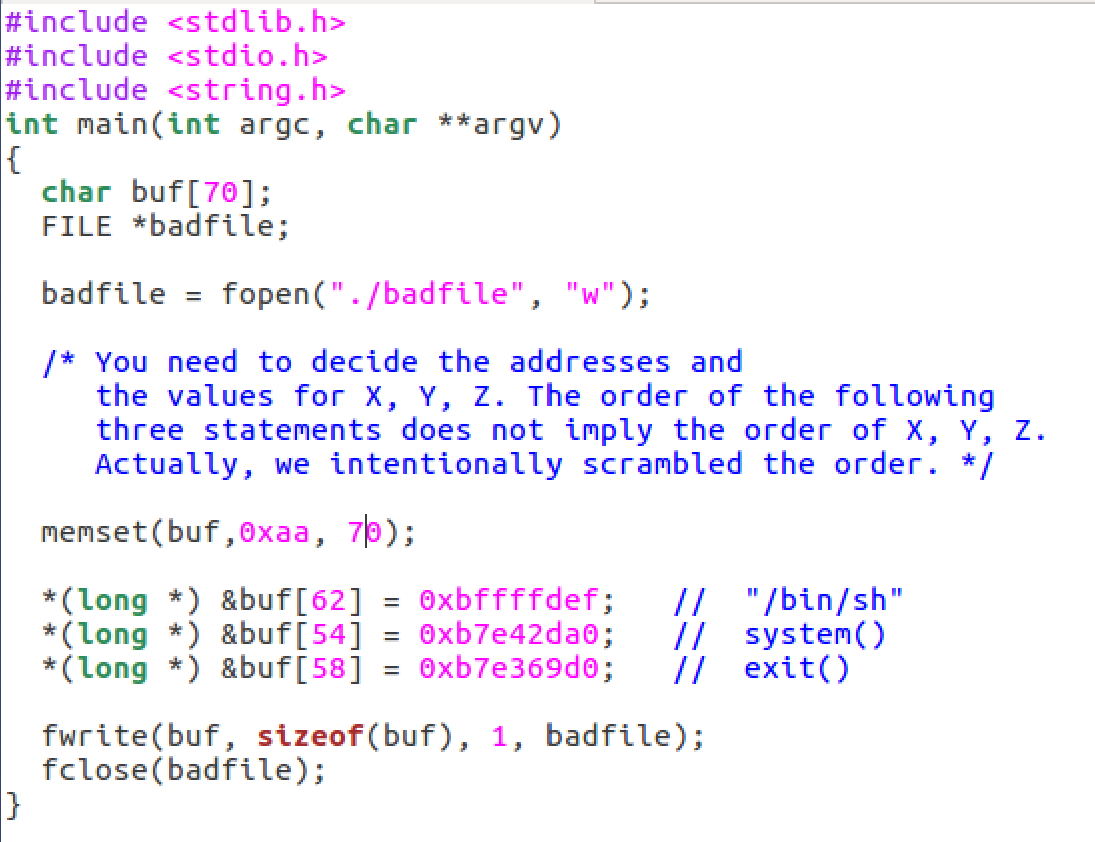
Comments:

When I changed the name to newretlib it changed the addresses and threw everything out of whack. So it gives me a segmentation fault because the memory addresses in the file are no longer the same as they were before.

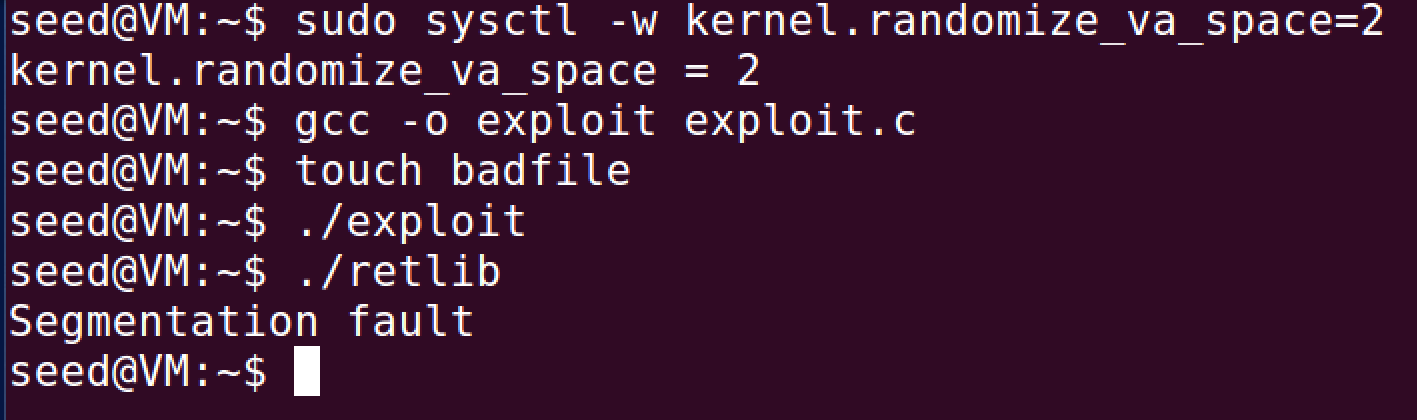
Task 4: Turning on address randomization

Code Used:

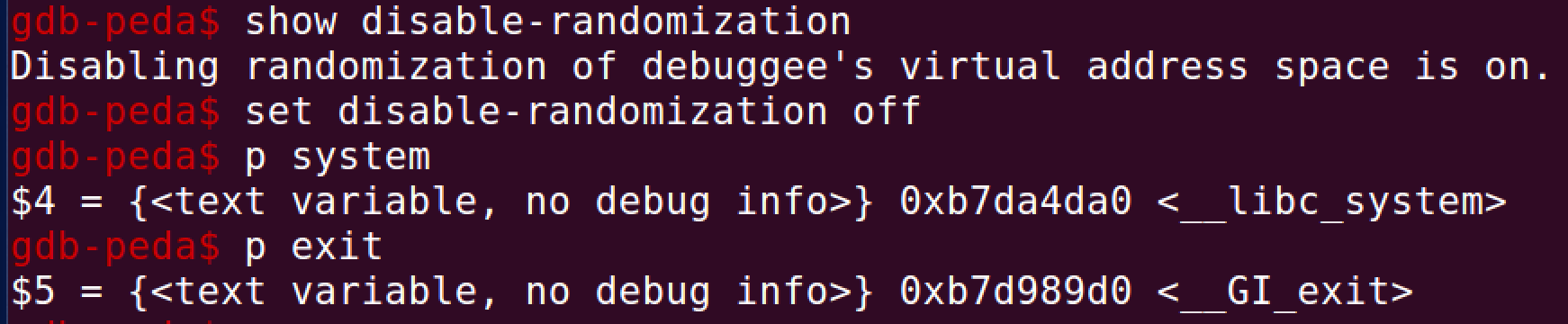


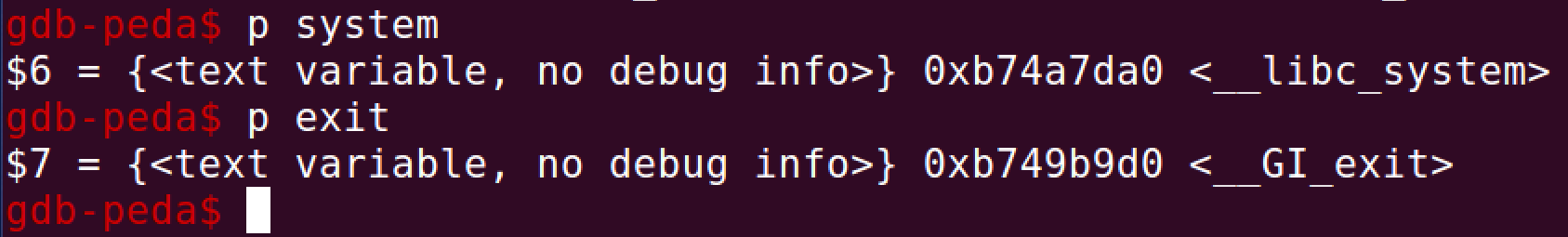


Commands Used:









Comments:

When I turned on randomization on again it would not work because it keeps changing the addresses whenever it is ran. Every address was different than before, so turning off randomization is key to getting this attack to work for this lab.