Thomas Gibbons

October 21, 2016

ECE 3220

LAB 6

Objective

The objective of this lab was to learn how to use the debugger to find segmentation faults in our code. It allows us to go through line by line and see where everything is at like variables and such and see exactly which line crashes.

Discussion

In this lab we had a handful of programs that we had to fix using the debugger or sometimes just using the warnings the compiler gave. There was one main program and then 8 small programs to debug.

The main program we used had one line that accessed a location outside the array and tried to place an integer inside which resulted in a fault. This was fixed by expanding the array size.

Next we had to go through 8 programs. In program 1, I found going through the debugger that scanf was returning a null char pointer because it was not allocated so I fixed that and I was unclear exactly what the program was supposed to do so for the next part I changed it to receive each character from the file and print to the screen as a solution to it not finding the EOF.

In program 2, I simply changed the unallocated pointer to an array.

In program 3, I found there was an overload. This was caused by the factorial function. When examining it I realized it would never stop running, so I made a base case where it would return some value.

In program 4, I changed the size of arr1 because there was an issue with strcpy. It would want to place a bigger array into a smaller one. After that issue there was nothing being printed so I changed the final character to a null terminator so it would display.

In program 5, I am unsure why it works, but it has worked on both the linux computers and my windows computer. All I did was remove the free since no memory was allocated.

In program 6, the compiler did this one as I made line 6 match on both sides of the equal sign. Then it worked fine regardless of the warning given.

In program 7, the problem came when trying to access a value in the null terminator during the print. This was easily solved by making the pointer not null and instead point to x.

In program 8, the problem came when trying to scan in a string to an unallocated pointer. I fixed it by making a statically allocated array.

As always a link to my github account is https://github.com/tgibbons95/Lab6.

Results from all 9 programs.

```
C:\Users\Thomas\Desktop\C code\LAB6>lab6-ex1
Specify file name you would like to print to:
lalala.txt
Hungry Hungry
Hippos
CTRL+d is a correct ending
C:\Users\Thomas\Desktop\C code\LAB6>lab6-ex2
MizPou


C:\Users\Thomas\Desktop\C code\LAB6>lab6-ex3
Enter the number
6
The factorial is 720

C:\Users\Thomas\Desktop\C code\LAB6>lab6-ex4
AAAAAAAAAAAAAAAAAA

C:\Users\Thomas\Desktop\C code\LAB6>lab6-ex5
hello

C:\Users\Thomas\Desktop\C code\LAB6>lab6-ex6
Character: 3 and a
Character: 2 and a
Character: 1 and a
 main:0040138C
 ch:00671598
 ch+1:00671599
 ch+2:0067159A

C:\Users\Thomas\Desktop\C code\LAB6>lab6-ex7

 The value of x is = 1309
 The pointer points to the value = 1309

C:\Users\Thomas\Desktop\C code\LAB6>lab6-ex8
Enter the input string
Thomas
Thomas
C:\Users\Thomas\Desktop\C code\LAB6>Lab6_SegFault_Example.c

C:\Users\Thomas\Desktop\C code\LAB6>Lab6_SegFault_Example
Enter an integer between 0 to 9999: 1234
x[1234] = 1234
```

Screenshot of the files committed and fixed.