THIS IS THE README FILE FOR LAB 1.

Name: Sam Espanioly

- UNIX is a proprietary system (i.e., you must purchase a license) while Linux is an Open-Source system. An Open-Source system, however, is not always "free". Why?

Some companies require license to use their version of Linux, or others who create Linux versions would charge to make their version supprotable for certain apps or services.

- Name another difference between Unix/Linux from your required reading.

Linux is free open-source. Unix is a command base meanwhile Linux is GUI interface.

- What made UNIX different from all other commercially available software systems when it was initially developed?

The fact that it's portable on all devices from small chips like rasberry pi to big server and commercial computers and everything in between.

- What operating system covers the widest range of hardware in the world?

Linux/Unix does.

- Why was the C programming language initially created?

It was created to make Unix system.

- How is white space (tabs/newlines/returns, spaces, etc.) handled within a C program file?

They are handled only if there was a backlash they have to be entered inside a printf command, a space being a space, new line \n tab \t vertical tab \v so it is not understood from keyboard buttons and instead from backlash + letter (n,r,h,v....) but they are ignored by the compiler for the most part.

- What did you learn when you "mistyped" (e.g. "fat fingered") the compile command?

Mistyping can cause run-time errors and it can be easy to spot depending on where mistypo happens.

- When a line of code is printed out after the **next** command in gdb, that line of code has just finished executing. True or False?

FALSE

- What is the value of **maxEntries** variable at GDB item #3?

6

- What is the value of **getchar_return_value** at GDB item #4? Is this what you expected to see?

54

- What is the value of **keyboard_value[0]** at GDB item #5? Did gdb give you additional information this time? What did you find out when you referred to the Extended ASCII table?

49

- What are the values in the first 6 keyboard_value elements?

keyboard_value[0]=49
keyboard_value[1]=50

keyboard_value[2]=51
keyboard_value[3]=52
keyboard_value[4]=53
keyboard_value[5]=54

- How is the output different when the **print keyboard_value** instruction is used?

print keyboard_value is of type char therefore gave a decimal value for the numbers.

- How does the output of the program change at GDB item #6 from the output seen prior?

This changes the first character in the var keyboard_value to "@23456" since 64 in the decimal values on the ASCII table is @.
the output becomes:

```
@
2
3
4
5
6
```

- How does the output of the program change at GDB item #7 from the output seen prior?

That changes the fourth value which is 4 to the character % therefore the output becomes:

```
1
2
3
%
5
6
```

- How does the output of the program change at GDB item #9 from the output seen prior? What are the values in the first 11 **keyboard_value** elements?

keyboard_value[0]=49
keyboard_value[1]=50
keyboard_value[2]=51
keyboard_value[3]=52
keyboard_value[4]=53
keyboard_value[5]=10
keyboard_value[6]=65
keyboard_value[7]=66
keyboard_value[8]=67
keyboard_value[9]=68
keyboard_value[10]=0

Are these values what you expected to see? If so, why? If not, what did you observe/learn?

After running this so many times it makes sense but the first time I was surprised to see \n there.

- How does the output of the program change at GDB item #10 from the output seen prior? What are the values in the first 11 keyboard_value elements?

keyboard_value[0]=98
keyboard_value[1]=97
keyboard_value[2]=110

keyboard_value[3]=97
keyboard_value[4]=110
keyboard_value[5]=97
keyboard_value[6]=0
keyboard_value[7]=0
keyboard_value[8]=0
keyboard_value[9]=0
keyboard_value[10]=0

Are these values what you expected to see? If so, why? If not, what did you observe/learn?

Yes because that's all what was in the input file 'banana'

- What is the value of **maxEntries** in GDB item #10? Why is this the correct value?

The value is 6 even though in the input file it is 10 but that is only because there are only 6 characters in banana.

- Run lab1 through gdb using lab1.input2 a second time but set a breakpoint at Line 29 and check the value in **getchar_return_value** each time through the loop. What value does it have the last time through the loop? The value in the extended ASCII table says 255 represents "Latin small letter y with diaeresis" is that what we got? Might -1 mean something else in this case? Might the value not be a char? Use the linux command **man getchar** to help you answer this question.

getchar starts at 0 then 98 then 97 then 110, 97, 110 , 97 which spells banana in the decimal value. It returns the value of the character in decimal value on the ASCII table.Returns -1 when there's an error reading the character or if its EOF.

- Have you ever used the gdb debugger before? If so, for what class(es)?

First time.