BY SUBMITTING THIS FILE TO CARMEN, I CERTIFY THAT I HAVE STRICTLY ADHERED TO THE TENURES OF THE OHIO STATE UNIVERSITY’S ACADEMIC INTEGRITY POLICY WITH RESPECT TO THIS ASSIGNMENT.

THIS IS THE README FILE FOR LAB 1.

Name: Yifan Yao.740

1. UNIX is 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?
   1. Although Linux is free to use for everyone, but in production environment, it is not free to get support from distributer (for immediate access to crucial security patch and reliability problems).
2. Name another difference between Unix/Linux from your required reading.
   1. Linux and Unix are using different file system, and everyone can build their own kernel as they wish.
3. What made UNIX different from all other commercially available software systems when it was initially developed?
   1. Other commercially available software was specifically designed for one system, but Unix kernel is able to adapt to every specific system.
4. What operating system covers the widest range of hardware in the world?
   1. Linux
5. Why was the C programming language initially created?
   1. To “create the UNIX system”.
6. How is white space (tabs/newlines/returns, spaces, etc.) handled within a C program file?
   1. “Ignored by compiler for the most part.”
7. What did you learn when you “mistyped” (e.g. “fat fingered”) the compile command?
   1. Create multiple copy of your source code or use version-control system.
8. 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?
   1. False
9. What is the value of **maxEntries** variable at GDB item #3?
   1. 6
10. What is the value of **getchar\_return\_value** at GDB item #4? Is this what you expected to see?
    1. 49, Yes because 49 represents ‘1’ in ASCII.
    2. A screenshot of a cell phone

       Description automatically generated
11. 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? Did you have an “Aha!” moment? What was it?
    1. 49 '1', 1 is corresponding with 49 in ASCII table, Yes gdb gives me additional information (it converted the ASCII to character for me), the getchar\_return\_value will only return the character’s representation in ASCII but the gdb will provide conversion for value storied in array.
    2. A screenshot of a cell phone

       Description automatically generated
12. What are the values in the first 6 keyboard\_value elements?

**keyboard\_value[0]= 1**

**keyboard\_value[1]= 2**

**keyboard\_value[2]= 3**

**keyboard\_value[3]= 4**

**keyboard\_value[4]= 5**

**keyboard\_value[5]= 6**

1. How is the output different when the **print keyboard\_value** instruction is used?
   1. "123456\000\000\000\000\000\000\000\000\000\000\220\022@\000\000\000\000\000\200"
   2. A screenshot of a cell phone

      Description automatically generated
2. How does the output of the program change at GDB item #6 from the output seen prior?
   1. The first character should be ‘1’, but it has been changed to ‘@’ by setting “set keyboard\_value[0] = 64”
   2. A screenshot of a cell phone

      Description automatically generated
3. How does the output of the program change at GDB item #7 from the output seen prior?
   1. The fourth character should be ‘4’, but it has been changed to ‘@’ by setting “set keyboard\_value[4] = ‘@’”
   2. A screenshot of a cell phone

      Description automatically generated
4. 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?

**1**

**2**

**3**

**4**

**5**

**A**

**B**

**C**

**D**

A screenshot of a cell phone

Description automatically generated

**keyboard\_value[0]= 49 '1'**

**keyboard\_value[1]= 50 '2'**

**keyboard\_value[2]=** **51 '3'**

**keyboard\_value[3]=** **52 '4'**

**keyboard\_value[4]=** **53 '5'**

**keyboard\_value[5]=** **10 '\n'**

**keyboard\_value[6]=** **65 'A'**

**keyboard\_value[7]=** **66 'B'**

**keyboard\_value[8]=** **67 'C'**

**keyboard\_value[9]=** **68 'D'**

**keyboard\_value[10]=** **0 '\000'**

A screenshot of a computer

Description automatically generated

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

Yes, I was expected to see these values, because ‘\n’ also count as a character.

1. 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?

**b**

**a**

**n**

**a**

**n**

**a**

**A screenshot of a cell phone

Description automatically generated**

**keyboard\_value[0]=** **98 'b'**

**keyboard\_value[1]=** **97 'a'**

**keyboard\_value[2]=** **110 'n'**

**keyboard\_value[3]=** **97 'a'**

**keyboard\_value[4]=** **110 'n'**

**keyboard\_value[5]=** **97 'a'**

**keyboard\_value[6]=** **10 '\n'**

**keyboard\_value[7]=** **0 '\000'**

**keyboard\_value[8]=** **0 '\000'**

**keyboard\_value[9]= 0 '\000'**

**keyboard\_value[10]= 0 '\000'**

A screenshot of a cell phone

Description automatically generated

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

* 1. Yes, I was expected to see these values, because there are only 7 character were inputted in this case, it is normal to leave array position 7 to 10 empty.

1. What is the value of **maxEntries** in GDB item #10? Why is this the correct value?
   1. 7, when there are no sufficient characters provided into the loop, the maxEntries will set to the total number of keyboard\_value.
   2. A screenshot of a cell phone

      Description automatically generated
2. Run lab1 through gdb using lab1.input2 a second time, but set a breakpoint at “**if (getchar\_return\_value != EOF){“** (line 29 in my program) 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.
   1. -1, the value in the extended ASCII table says 255 represents “non-breaking space”, -1 might means not an ASCII character, from man-page shows that end of file (EOF) or error.
   2. A screenshot of a cell phone

      Description automatically generated
3. Describe how you created the breakpoint in your program while using **ddd** instead of **gdb**.
   1. Select the statement and click the STOP sign from tool bar.
   2. A screenshot of a social media post

      Description automatically generated
4. Have you ever used the gdb/ddd debugger before? If so, for what class(es)?
   1. Yes, for Intro to C++.