Jacob Stone

1. **Browse a number of manual pages of your choice from section 1, 2, and 3. Observe that they contain common "headings" (NAME, SYNOPSIS, DESCRIPTION, SEE ALSO). Most manual pages in Sections 2 (System Calls) and 3 (Subroutines) also include RETURN VALUE and ERRORS. Briefly describe the purpose of any two "headings"**
   The purpose of the DESCRIPTION heading is to give a brief summary explaining the section of man, how to invoke the calls, and examples of some commands. This section also will give a guide on how to find more information by giving the reader a command to type to find more (like “see syscalls(2)”).
   The purpose of the RETURN VALUE heading is to give the user a variety of different outcomes that could occur depending on what they use a command with. It shows the error messages and why they happen, It shows the correct return value when used properly, and specific cases are shown towards the end like if the user had to give any more specifics when typing the command.

2. **Describe the difference between the Unix user command time and the System call time().**
   The difference between the unix command time and the system call time is the user command time will run program and summarize system resource usage. The system call time() will display the time in seconds since the epoch (Jan 1, 1970).

3. **What is the meaning of the stream-based SEEK_CUR macro?**
   - **Start by using a keyword search on the man pages to find possible seek() function calls (either from section 2 or 3)**
   - **Use the man pages to discover which include files are referenced by those function calls**
   - **Scan the include files to discover the meaning of the macro -Alternatively, use grep to search for the desired string**

   The SEEK_CUR macro is added to function calls like fseek to find the offset relative to the current position indicator

4. **What is the command to list the contents of a directory in "long mode", including hidden files?**

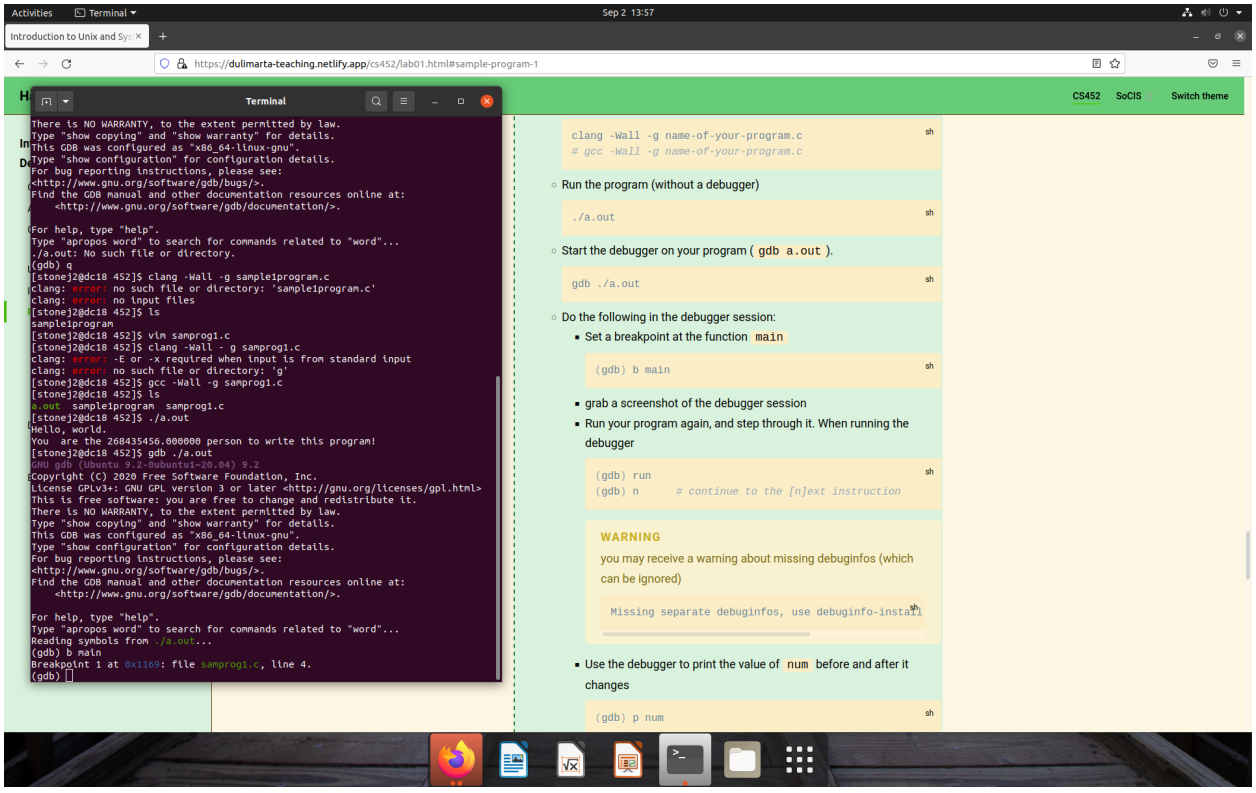
   ls -la

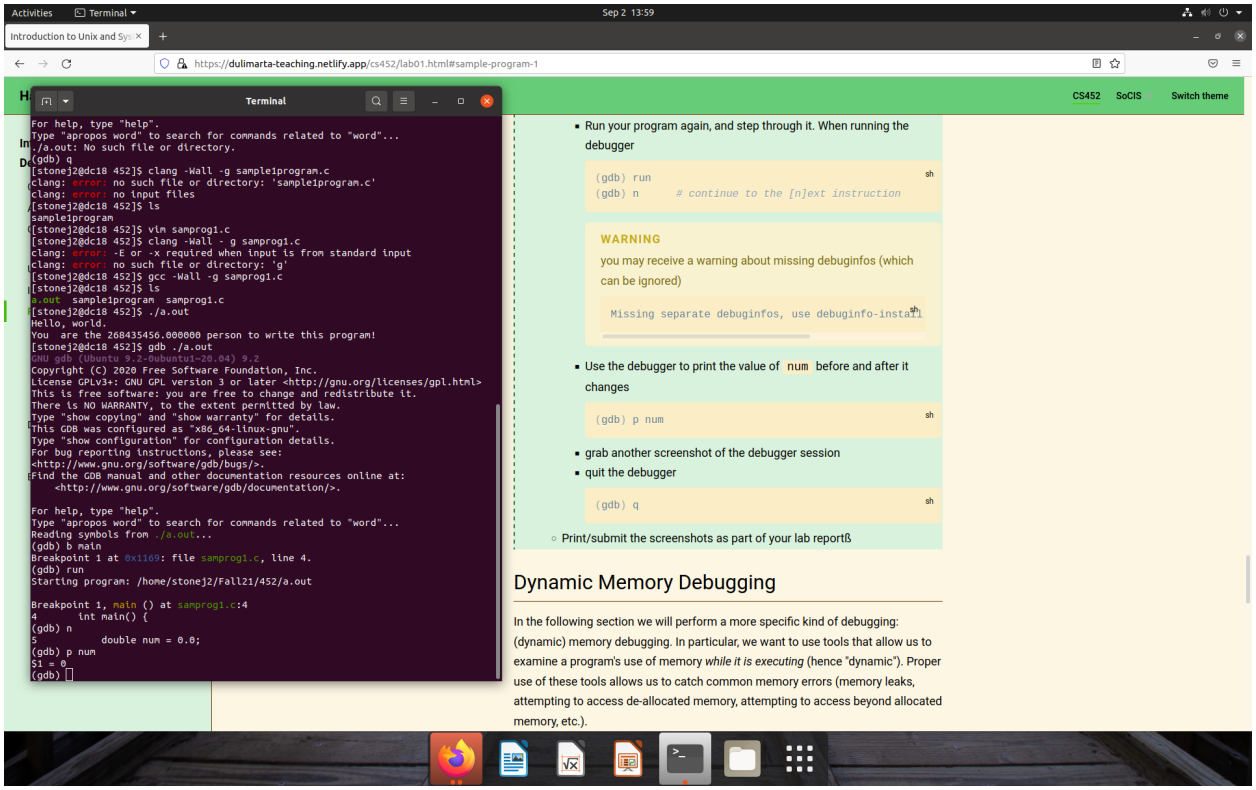
5. **What is the command syntax to make a directory readable/writable only by you?**

   chmod [u][[+][rw]]

6. **Perform the following steps:**

- **Create the above program**
- **Compile the program (remember to include debugging flag and link all necessary libraries)**
- **Run the program (without a debugger)**
- **Start the debugger on your program (gdb a.out).**
- **Do the following in the debugger session:**
- **Set a breakpoint at the function main**
- **grab a screenshot of the debugger session**
- **Run your program again, and step through it. When running the debugger**
- **Use the debugger to print the value of num before and after it changes**
- **rab another screenshot of the debugger session**
- **quit the debugger**





7. Describe precisely (nature of problem, location) the memory leak(s) in the above program.

- **Fix the problem(s) by adding necessary free() call(s). But, don't change the malloc() calls, mark your modification(s) with readable C comments**
  - **Add your name(s) as a C comment at the beginning of source code.**
  - **Print and submit the modified source code**
  - **To confirm that the memory leaks have been fixed, use valgrind again on your revised code.**

The memory leaks occurred because data 2 was never freed and data 1 would not have been freed when exiting the program through the use of quit

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


#define SIZE 16

//Jacob Stone
```

```
int main()
{
        char *data1, *data2;
        int k;
        do {
        data1 = malloc(SIZE);
        printf ("Please input your EOS username: ");
        scanf ("%s", data1);
        if (! strcmp (data1, "quit")){
                free(data1); // added braces and free command
                break;
        }
        data2 = malloc(SIZE);
        for (k = 0; k < SIZE; k++)
        data2[k] = data1[k];
        free (data1);
        printf ("data2 :%s:\n", data2);
        free (data2); // added free command
        } while(1);
        return 0;
}
```

8. How many times is the write() system call invoked when running Sample 2?. Note: run several different experiments, then try to answer as a formula.

write() = 32 + 9 + strlength() // length of string entered

9.Use the example the source code and experiment to answer the question: what is the primary C library subroutine that causes the write() system call to be invoked by Sample 2?

The primary library subroutine is printf.