1. **Briefly describe the purpose of (what is found in) any two sections commonly found on a man page (e.g. the synopsis section of the malloc() system call)**

Aside from the ‘**SYNOPSIS’**, a man page generally always has a ‘**NAME**’, which gives the name and a short phrase explaining what the command does. A man page will also usually have an ‘**EXAMPLES**’ section, which includes practical applications of the command in different scenarios.

1. **Describe the difference between the UNIX shell command write and the UNIX system call write()’**

**write** allows you to communicate with other users, by copying lines from your terminal to theirs. 

**write()** writes up to count bytes from the buffer starting at buf to the file referred to by the file descriptor fd. 

**write** is for user to user communication, while **write()** is lower level for writing data directly to files / devices.

1. **What is the meaning of the stream-based SEEK\_SET macro?**

* **Start by using a keyword search on the man pages to find possible "seek" function calls applying to a stream**

****

fseek (3) - reposition a **stream**

seekdir (3) - set the position of the next readdir() call in the directory **stream**.

* **Use the man pages to discover which include files are referenced by those function calls**

****

stdio.h

* **Scan the include files to discover the meaning of the macro**

The fseek() function sets the file position indicator for the stream pointed to by stream. The new position, measured in bytes, is obtained by adding offset bytes to the position specified by whence. If whence is set to **SEEK\_SET**, SEEK\_CUR, or SEEK\_END, the offset is relative to the start of the file, the current position indicator, or end-of-file, respec‐ tively. A successful call to the fseek() function clears the end-of-file indicator for the stream and undoes any effects of the ungetc(3) function on the same stream.

The **SEEK\_SET** is used to set the new position offset bites away from the start of the file.

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

**ls -a**

ls: List information about the FILEs (the current directory by default).

-a: do not ignore entries starting with .



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

**chmod 700 dir\_name**

A numeric mode is from one to four octal digits (0-7), derived by adding up the bits with values 4, 2, and 1. Omitted digits are assumed to be leading zeros. The first digit se‐ lects the set user ID (4) and set group ID (2) and restricted deletion or sticky (1) at‐ tributes. The second digit selects permissions for the user who owns the file: read (4), write (2), and execute (1); the third selects permissions for other users in the file's group, with the same values; and the fourth for other users not in the file's group, with the same values.



1. **Perform the following operations:**

**• Create the above sample program (via copy/paste)**

**• Compile the program (remember to include debugging information and link all necessary libraries)**

**• Run the sample program**

**• Start the debugger on your program (gdb sampleProgramOne)**

**• Set a breakpoint at the function main**

**• Take a screenshot (screenshotOne)**

****

**• Run your program within the debugger (run) and step through it**

**• Use the debugger to print the value of num before and after it changes**

**• Take a screenshot (screenshotTwo)**

**A computer screen shot of a program

Description automatically generated**

1. **Describe precisely (nature of problem, location) the memory leak(s) in sample program2. Fix the problem(s) and submit your corrected source code. Note: it would be a good idea to use valgrind to verify that you have fixed the problem(s)!**

A screen shot of a computer program

Description automatically generated

The issue is that when we dynamically allocate space using malloc, we must free up that space when we are done using it. The do loop continues to run until the user types ‘quit’, then the loop breaks. When the loop breaks, we don’t reach the free(data1), so we have to add another one up above inside the if. Additionally, data2 is never freed, so we must do so after it is done being used in the printf.

1. **How many times is the write()system call invoked when running sample program 2? Experiment with executing the loop a different number of times. Then express your answer as a formula.**

The write() system call is invoked every time printf() is being called, so if **n** is the number of times the user inputs text, then the number of times write() is invoked is:

**f(n) = 2n-1**

because on the last input, it will be ‘quit’ so the next printf is not called.

1. **Examine the source code and output to answer the question: what is the primary 'C' library subroutine that causes the write()system call to be invoked while executing sample program 2?**

The primary ‘C’ library subroutine that causes the write() system call to be invoked while executing sampleProgramTwo is stdio.h.