Lab One

by Jarred Parr and Alexander Fountain

1. The Synopsis section outlines the command and the options with which it can be used. It shows all combos that can be employed as well, this refers to the combinations of options flags, and positional arguments that are used in a particular sequence to use a piece of functionality.

   The Description section describes the flags and positional options in detail. Instead of just listing them like what can be see in the synopsis section, the description actually gives an explanation of the intended behavior of each option.

2. The `write` command is a user-level command which allows you to copy lines from your terminal session to that of another user in your local network. The system call `write()` is used to write a sequence of bytes into the system or another running program. This has higher associated privilege as you can inject the bytes into more locations than you otherwise would be able to with the `write` command.

3. Found in the function fseek() man page, it is used with the fseek() function and provides the utility of placing the file pointer at the beginning of the file.

4. You can use `ls -a/--all` to list all files in a directory, and to display them in a long list format you can make the command `ls -al` which allows you to have a much more readable list of files when it prints them all.

5. To give access to only yourself on a directory, you must use `chmod 700 <FOLDER>`

6.

```
This code was compiled with gcc -g -Wall <program.c>, g must be used to
allow gdb to recognize the binary and step through it.

(gdb) break 3
Breakpoint 1 at 0x1151: file gdbsample.c, line 5.
(gdb) start
Temporary breakpoint 2 at 0x1151: file gdbsample.c, line 5.
Starting program: /home/ghost/Code/cs/CIS452/lab1/a.out

Breakpoint 1, main () at gdbsample.c:5
5       double num = 0.0;
(gdb) n
6       printf ("Hello, world.\n");
(gdb) print num
$1 = 0
(gdb) n
Hello, world.
7       num = pow(2, 28);
```

```
(gdb) n
8       printf ("You are the %f person to write this program!\n", num);
(gdb) print num
$2 = 268435456
(gdb) n
You are the 268435456.000000 person to write this program!
9       return 0;
(gdb) n
10  }
(gdb) n
0x00007ffff7e06223 in __libc_start_main () from /usr/lib/libc.so.6
(gdb) n
```

7. The first problem is that on line 11 a do while loop occurs, so there will always be one extra allocation of `data1` and `data2` in memory even after adding the free for `data2` after the value prints. To get around this, we must add an additional free of `data1` and `data2` after the while loop declaration. The fixed code is as follows:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define SIZE 16

int main() {
    char* data1, *data2;
    int i;

    do {
       data1 = malloc (SIZE);
       printf ("Please input your eos username: ");
       scanf ("%s", data1);
       if (!strcmp (data1, "quit"))
          break;
       data2 = malloc (SIZE);
       for (i=0; i<SIZE; i++)
          data2[i] = data1[i];
       free (data1);
       printf ("data2 :%s:\n", data2);
       free(data2);
    } while (1);
    free(data1);
    free(data2);

    return 0;
}
```

8. Write is invoked as many times as the command enters the while loop + 1. So since the program runs once, that iteration is automatically going to happen and write will be called once. Every instance following that where the input is not equal to `quit` will incur another usage of the write command
9. The C library subroutine that is called is the `printf` function call which uses the write system call to write the bytes to a particular file descriptor whether it be `STDOUT` or otherwise.
10.

```
// This can also be accompisled with the getpass() function on *NIX
operating systems.

#include <iostream>
#include <cstdlib>
#include <cstring>
#include <termios.h>

int main() {
  std::cout << "Disabling echo." << std::endl;
  termios t;
  tcgetattr(1, &t);
  termios tt = t;
  tt.c_lflag &= ~ECHO;
  tcsetattr(1, TCSANOW, &tt);

  std::string s;
  std::cout << "Enter secret word/phrase: ";
  std::getline(std::cin, s);
  std::cout << "\nYou Entered: " << s << std::endl;

  tcsetattr(1, TCSANOW, &t);
  std::cout << "Default behavior restored." << std::endl;
  std::cout << "Enter visible word/phrase: ";
  std::getline(std::cin, s);

  return EXIT_SUCCESS;
}
```