CSCD 340
Lab 4

You answers will be in the form of a PDF/C Code. You will create one PDF name your last name first letter of your first name lab4.pdf. You will submit a pdf file named your last name first letter of your first name lab4.pdf (Example steinerslab4.pdf)

1. In class we are discussing simple system calls in particular as system calls related to "The Universality of I/O". I have provided lab4prob1.c. This program contains a main function. The main function simply opens a file named test.txt, reads line by line, as it reads it prints that line to the screen. At the end of the program the file is closed.

   Compile the code with the executable named prob1. The input file is included.

   Now execute the command **strace -o prob1out.txt -rf ./prob1**. Open prob1out.txt and annotate the system calls and what is happening on. You don't need to annotate every line primarily focus on open, write, read and close if they exist. Note: you are expected to try and explain what is happening; not just say "look here is the system call".

   Place the comments and the output from above into the PDF.

2. I have provided Lab4Prob2.java. Compile the java code and then execute the command **strace -o prob2out.txt -rf java Lab4Prob2** You don't need to comment any of the open, close, read, write for this portion of the lab; however, in your PDF you must write at least a 2 sentence comparison of the system calls from the Java world compared to the the system calls in the C world. Explain what is java doing that C isn't or vice versa. You will not be providing the output of the Java run.

3. There are a set of questions below. Include an answer to each question as a comment at the end of the C file. If you don't have a 64 bit Linux distribution or you don't have gcc-multilib installed for your 64 bit distribution use cslinux.eastern.ewu.edu.

   Start by creating the following simple C program, named lab4Prob3.c

```
#include <stdlib.h>

int main()
{

   write(1, "hello\n", 6);

   write(1, "goodbye\n", 8);

   exit(0);

}
```

Compile and execute the program. Your executable will be named prob3. The program should print “hello” and “goodbye” to stdout, and then exit.

Now add this function to your C file above main:

```
int write(int fd, char *str, int len)
{
  /* do nothing */
  return 0;
}
```

a) When you compile, link, and run, nothing will print out. Why?

Change the write line **in main** for “goodbye” to

```
syscall(1, 1, "goodbye\n", 8);
```

Compile and run in 64 bit mode. NOTE: You will receive an implicit declaration of syscall, simply ignore it. If you don’t have a 64 bit machine compile and run on cslinux. NOTE: This must be done on a 64 bit machine.

b) Now, “goodbye” is printed, while “hello” still isn't. Why?

Without changing the program, compile using the -m32 flag to gcc, so that you compile for 32-bit mode. To complete this portion use gcc –m32 lab4Prob3.c NOTE: This must be done on a 64 bit machine.

c) Where are the 32 bit libraries as compared to the 64 bit libraries?

d) Where would one look in the libraries to figure out why when you run the program nothing is printed?

e) What differences in these library files would cause it to print in one case but not the other?

Add this function to your C file above main,

```
int syscall(int n, ...)  // this is correct don’t change anything it is … for the parameter
{
  /* do nothing */
  return 0;
}
```

Compile and execute your code in both 64 and 32 bit mode.

f) The program doesn't print output in either 32-bit or 64-bit mode. Why?

Add the following to the syscall function

```
   asm("mov %0, %%ecx\n"
  "mov $1, %%ebx\n"
  "mov $8, %%edx\n"
  "mov $4, %%eax\n"
 "int $0x80\n"
  :
  : "r" ("goodbye\n")
 );
```

Note: The assembly must be exactly as written.

Compile in 32-bit mode with -m32 and execute the code. DO NOT compile in 64 bit mode, it won't compile.

g) The "goodbye" printout is back. Why?

h) Can any C function declaration interfere with this variant? Why or why not?

Compile the code in 32-bit mode and execute the command **strace –f –o prob3out.txt ./prob3**. Open the text file and annotate the system calls and what is going on. You don't need to do every line mainly focus on and try to identify where the output switches from user to kernel mode. Note: you are expected to try and explain what is going on.

Add the answers to the above questions to the PDF as well as the output from strace.

TO TURN IN

Your PDF – Clearly denote each problem!