#### COEN 146: Computer Networks

**Lab 2: Multithreading in C and File transfer**

**Objectives**

##### To develop multithreading in C

##### To write C programs for copying files

##### 

##### **Guidelines**

##### For all COEN 146L you need to develop a good knowledge of C in a Linux development environment. You are highly encouraged to use command line tools in developing, compiling, and running your programs.

##### Please pay attention to your coding style and good programming practices, if your program is not worth documenting, it will not worth running.

##### Skills in multithread programming is required for developing client-server applications, as a way to create parallelism. For example, a server spawns a separate thread for every client connection to look after the specific client requests independently. A thread is a single sequence stream within in a process, and it is often referred to as a lightweight process. Threads operate faster than processes during their creation and termination, context switching, and communication. Threads are not independent like processes and they share with other threads their code and data, open file descriptors. Threads maintain their own program counters, registers, and stack.

##### Recall from Lab 1 the following library function are used for setting up threads.

#include <pthread.h>

int pthread\_create(pthread\_t \*thread, pthread\_attr\_t \*attr,

##### void \*(\*start\_routine) (void \*arg), void \*arg);

##### **Multithreading in C – Copying files**

Problem: Write a C program to copy binary/ text files simultaneously.

Analysis:

* Input: pass file names as arguments to main(), so your main function needs to be defined as follows:

int main(int argc, char \* argv[])

Your files: src.dat and dest.dat files.

* File reading can be accomplished by using either:
  + Functions: fopen, fwrite, and fread for binary files or fprintf and fscanf for text files
    - FILE \*fopen(const char \*filename, const char \*mode)
    - fwrite( ptr, int size, int n, FILE \*fp ); or fprintf() (for text files)
    - fread( ptr, int size, int n, FILE \*fp ); or fscanf() (for text files)
    - fclose(ptr);

e.g.

FILE \*fp;

fp = fopen(“src.dat”,"r”);

fp = fopen(“dest.dat","w”);

fwrite(&buf,sizeof(buf),1,fp);

fread(&buf,sizeof(buf),1,fp);

fclose(fp);

OR

* + System calls: open, read, write
    - int open (const char\* Path, int flags [, int mode ]);
    - size\_t read (int fd, void\* buf, size\_t cnt);
    - size\_t write (int fd, void\* buf, size\_t cnt);

e.g.:

int fd = open("foo.txt", O\_RDWR);

           int nw = write(fd, buf, strlen(buf));

int nr = read(fd, buf, 40);

close (fd);

You need to include the following libraries:

* + - #include<sys/types.h>
    - #include<sys/stat.h>
    - #include <fcntl.h>

You may create files of random data with different sizes/ bytes. You may use “cat” and “head” commands ( i.e. $cat /dev/random | head -c <bytecount>). /dev/random are special files that serve as pseudorandom number generator. “cat” is used to display the content and “head” is used to display the specified number of lines. So the result of “cat” is sent to the upstream end of PIPE and “head” receives these results and redirects the content of the specified bytes to a file.

$cat /dev/random | head -c 100000 > src1.dat 🡪 creates a file with 100KB

$cat /dev/random | head -c 1000000 > src2.dat 🡪 creates a file with 1MB

Check the size of the files with the command “ls -la”

1. Write your C program to copy files (binary and text) using functions, compile, debug, run, and test
2. Write your C program to copy files (binary and text) using system calls, compile, debug, run, and test
3. Calculate the time taken to copy files for both step 1 and 2. You may use clock()function (in time.hlibrary). Call the clock function at the beginning and end of the code to measure the time, subtract the values, and then divide by CLOCKS\_PER\_SEC (the number of clock ticks per second) to get processor time. You may use the following code snippet:

#include <time.h>

clock\_t start, end;

**double** cpu\_time\_used;

start = clock();

... /\* Time consuming process. \*/

end = clock();

cpu\_time\_used = ((**double**) (end - start)) / CLOCKS\_PER\_SEC;

1. Write your C program to copy multiple files (say 10 files) by creating threads, each of which will be responsible of copying one file. Use good style practices, compile, debug, run, and test for large sizes of matrices.

Demonstrate your programs to the TA

**Requirements to complete the lab**

Show the TA correct execution of the program you wrote and upload source code to Camino.

Be sure to retain copies (machine and/or printed) of your source code. You will want these for study purposes and to resolve any grading questions (should they arise)