#### COEN 146: Computer Networks – Fall 2018

**Lab assignment 2: Multithreading in C and network commands**

**Objectives**

##### To develop multithreading in C

##### To demonstrate network commands

##### To measure packet delays

##### To setup client-side connection to HTTP server

##### 

##### **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 for setting up threads.

#include <pthread.h>

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

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

**Part 1: Quiz [4 points]**

Complete the quiz on Part 1 of Google Lab2\_Turn-in Form through Camino.

##### **Part 2: Multithreading in C [8 points]**

Exploit parallelism in large scale multidimensional matrix multiplication.

Problem: Write a C program so that the multiplication of two large matrices with N x M and M x L dimensions is performed at a minimum time.

Analysis:

* Input matrices: double matrixA[N][M], matrixB[M][L]
  + note: the number of columns of the first matrix must be equal to the number of rows in the second matrix
  + The values of N, M, and L must be large to exploit parallelism
  + The entries of the matrixA and matrix may be filled with a random number generator

srand(time(NULL));

for (int i = 0; j < N; i++)

for (int j = 0; j < M; j++)

matrixA[i][j] = rand();

srand(time(NULL));

for (int i = 0; j < M; i++)

for (int j = 0; j < L; j++)

matrixA[i][j] = rand();

* Output matrix: double matrixC[N][L];
* Matrix multiplication is a loosely coupled problem and so decomposable, i.e. multiplication of each row of matrixA with all columns of matrix can be performed independently and in parallel
* The number of threads to be created in the program equals to N and each thread i would be performing the following task:

for (int j = 0; j < L; j++){

double temp = 0;

for (int k = 0; k < M; k++){

temp += matrixA[i][k] \* matrixB[k][j];

}

matrixC[i][j] = temp;

}

* The main thread needs to wait for all other threads before is displays the resulting matrixC

1. Write your C program for “matrix multiplication” with threads using good style practices, compile, debug, run, and test for large sizes of matrices

##### **Part 3: Network commands and tools [18 points]**

1. Run each of the following basic networking commands in Linux, and write down your observation and explain the usage of the command
   1. netstat: displays the contents of network interfaces, with – a option, the command displays the state of all active sockets (more to come on sockets as end points of communication). With – r option, the routing table is displayed. Please type man netstat to learn about all options.
   2. ifconfig: configures network interface parameters. With – a option, the state of all interfaces are displayed. Other options are also used to assign a new IP address to an interface, to assign a new network mask for an interface, to disable an interface, and more. Please type man ifconfig to learn about all options.
   3. hostname: displays and sets hostname of the system. Please type man hostname to learn about all options.
   4. ping: sends ECHO\_REQUEST datagram to a network host using ICMP protocol to elicit an ECHO\_RESPONSE from the host or gateway. Please type man ping to learn about all options.
   5. traceroute: displays the route packets trace to a network host using IP protocol “time to live”. Please type man tracerout to learn about all options.
   6. telnet: remote connection to server at a specific port (mainly 80 – http port)
   7. host/dig: performs DNS lookups.
   8. route: manipulates network routing tables.
   9. arp: displays and modifies the Internet-to-Ethernet address translation tables used by the address resolution protocol.

Note: some of the commands may require that you are logged in as a route.

1. Select three hosts in the internet (one in North America, one in Asia, and one in Europe), and experiment with pinging each host with different packet sizes ranging from 32-bytes to 1048-bytes. For each host,
   1. identify the packet loss.
   2. identify the RTT “Round-Trip-Time”.
   3. Explain the correlation between these measurements to the geographical location of the hosts.
2. Suppose that the IP address for one of the URL you selected in Step 3 is not cached in your local host, so a DNS lookup is necessary to obtain the IP address. Suppose that three DNS servers are visited before your host receives the IP address from DNS. The first DNS server visited is the local DNS cache, with an RTT delay of RTT0 = 3 msecs. The second and third DNS servers contacted have RTTs of 20 and 26 msecs, respectively. Initially, let's suppose that the Web page associated with the link contains exactly one object, consisting of a small amount of HTML text. Suppose the RTT between the local host and the Web server containing the object is RTTHTTP = 47 msecs.

Write a C program that computes the following:

* 1. Assuming zero transmission time for the HTML object, how much time elapses from when the client clicks on the link until the client receives the object?
  2. Now suppose the HTML object references 6 very small objects on the same web server. Neglecting transmission times, how much time elapses from when the client clicks on the link until the base object and all 6 additional objects are received from web server at the client, assuming non-persistent HTTP and no parallel TCP connections?
  3. Assume that the client is configured to support n parallel TCP connections, compute the response time in both persistent and non-persistent cases.

1. Setup a client side to connect to **gaia.cs.umass.edu 80**, as follows:
   1. Type **telnet gaia.cs.umass.edu 80** to connect to gaia.cs.umass.edu web server**,** explain what happens

Note: If you are using MacOS terminal, you may need to install telnet binary file to /usr/local/bin/

* 1. Type a GET HTTP request:

**GET /kurose\_ross/interactive/index.php HTTP/1.1**

**Host: gaia.cs.umass.edu**

Write down your observation.

* 1. Analyze the response message sent by the HTTP server, by answering the following questions:

 What is the name of the file that is being retrieved in this GET message?

 What version of HTTP is the client running?

 What formats of text and images, if any

1. Setup clients to connect to the three hosts you selected in Step 3, then experiment to connect to the public http port 80 and other ports, say: 3389. Write down your observations.

**Requirements to complete the lab**

1. Complete the quiz on Google Lab2\_Turn-in Form through Camino for Part 1.
2. Show the TA correct execution of the program you wrote for Part 2 and upload source code to Camino.
3. Show the TA correct execution of the program you wrote for Part 3 and upload source code to Camino
4. Submit your analysis and observations for Part 3 via Google Lab2\_Turn-in Form through 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)