ECE 428/CS 425 Distributed Systems, Spring 2008, University of Illinois at Urbana-Champaign

Machine Problem 0 - Tutorial

Due Date - None

Overview

The machine programming (MP) part of the course this semester involves building a peer to peer application. We will build this application in stages - each stage will be an individual MP. In all, there will be three to four MPs.
The purpose of this MP is to give you basic information about the projects, as well as get you started on learning to write multithreaded networking code.

Programming Language for MPs

The implementation language of choice for all MPs will be C (preferred) or C++ (we will not support Java). The choice of the preferred language C is motivated by the fact that the code templates we will give you are written in C, making it easier for you to focus on the essentials of the coding, and for us to automatically test your code. If you have never programmed in C before, but have programmed in either Java or C++, one or more of the following tutorials should help you close the gap. If you have never programmed in either C, Java, or C++, please meet the instructor.

1. Steve Holmes' guide to C Programming, http://www.strath.ac.uk/IT/Docs/Ccourse/
2. A. D. Marshall's guide to C, http://www.cs.cf.ac.uk/Dave/C/CE.html
3. Other online guides for C programming, http://www.google.com/search?q=c+tutorial
4. "C programming language", B. W. Kernighan, D. M. Ritchie, Prentice Hall, 2ed, 1988, ISBN: 0131103628.

Learning to Write Multithreaded Networking Code

All the MPs for this course will require you to write code that can send messages on a network, and most of them will also require you to use multiple threads. If you have not had experience with multithreaded programming or network programming, you should spend some time to figure these out now, so that you do not have trouble with future assignments. The communication interface of choice for this class is the sockets library. You can choose your own library for threads, yet for most MPs, using pthreads will suffice.

Sockets Programming: Online tutorials on Sockets programming and Unix network programming are given below:

1. "Beej's Guide to Network Programming" http://www.ecst.csuchico.edu/beej/guide/net
2. "Unix network programming", W. R. Stevens, (Addison-Wesley, 3ed, 2002, Vols. 1 and 2 – ISBN: 0130810819 and ISBN: 0131411551 OR Prentice Hall, 1ed, 1990, ISBN: 0139498761).

Reference 1 above is a good place to start - the code examples will help you with the Suggested Activities given later in this MP.

Multithreaded Code: For spawning new threads, you should look into the `pthreads` library. Use the Unix `man` command to find more information about these. Also, https://computing.llnl.gov/tutorials/pthreads/ has a good reference

Suggested Activities

Write two programs: a TCP server and a TCP client. The server should passively wait for clients to connect to it, and the client should initiate a connection to the server. Any useful server needs to be able to give the client information, so you should have your client send the server a request, and have the server respond. You also need to make sure the server is able to handle multiple client requests at the same time by handling each request in a separate thread. We recommend that you start by looking at some of the sample code in the above tutorials on sockets programming (especially Beej's online tutorial).

If you have not programmed in C before, we suggest you look at the chapters on "Programming with pointers" in the above tutorials. If you do not feel comfortable using pointers, your progress on future projects might be slowed down. Feel free to post questions about any of these topics on the Google group or to come by office hours.

Laboratory

We will be using the Engineering Workstations Lab. You will find more information about access (including remote access) and installed software at `http://www.ews.uiuc.edu/`. One suggestion is to use the linux machines dcllnx[1-70].ews.uiuc.edu which are running Redhat Enterprise v5. You are free to code and develop your MPs on any Linux platform - before submitting, please make sure your code is running on the above EWS machines! (otherwise our tests may not run with your code, and you will get a 0!)

Time-Saving Advice

For data structures that you need to use for your MP's (e.g., linked lists or hash tables), you may want to take advantage of the available C libraries (rather than implementing these from scratch). One suggestion would be to use GLib, a general-purpose utility library that is documented very well: see
`http://library.gnome.org/devel/glib/unstable/index.html`.
To debug memory errors, your first and most important resource is `gdb` - remember, this is far more systematic an approach than using printf's. Further, you learn something new that you can use later in your career! If you keep facing pesky bugs for which gdb does not help much, try Valgrind (`http://valgrind.org`). Valgrind is a suite of tools for debugging and profiling Linux programs, and especially useful in fixing memory programming errors that make your program `SEGFAULT` on you. EWS machines have version 3.2.1 installed.