SIENA*college*
Computer Science

Computer Science 330
Operating Systems
Siena College
Spring 2012

Lab 7: Socket Programming

Due: 9:20 AM, Friday, April 13, 2012

In this week's lab, you will do some programming with *sockets*.

You may work alone or with a partner on this lab.

Create a file in your directory for this lab in which you will answer the questions scattered throughout the lab.

Socket Concept

Remember that we have seen a number of ways to have processes cooperate. Threads share memory. Processes can be made to share memory. Processes that do not share memory may communicate over pipes.

Processes running on different systems must communicate across a network - most commonly this is done using *sockets*.

Sockets and pipes provide only a very rudimentary interprocess communication. Each "message" sent through a pipe or across a socket has a unique sender and unique receiver and is really nothing more than a stream of bytes. The sender and receiver must add any structure to these communcations.

Socket Tutorial

There are many excellent resources on socket programming, so please point a browser window to Beej's Guide to Network Programming, Using Internet Sockets by Brian "Beej Jorgensen" Hall. You are welcome to read through this entire document if you wish, but for the rest of this section, you will be directed to specific sections and examples.

Section 1 of the guide has basic background information, and you may wish to skim through it quickly.

Section 2 has the basics of sockets and the underlying protocols. Read this a bit more carefully unless it is already familiar to you.

Try out the `telnet` interaction with a web server by retrieving the file `/courses/message.txt` from our course web server. You should issue the command

```
telnet www.teresco.org 80
```

This makes a connection to `www.teresco.org` on port 80 (the default port for `http` connections). It's the same thing browsers do.

Question 1: What output is printed to your terminal when you issue this command? (1 point)

Now, request a web page:

```
GET /courses/message.txt HTTP/1.0
```

then hit return twice.

Question 2: What is printed? (1 point)

Section 3 continues some background information.

In Section 3.1, it is worth a read if you are unfamiliar, but we will not be concerned about the IPv6 addresses. As long as you know what an IP address is and a port number is, you're set here.

Section 3.2 is a bit more important in general, but since we are likely to be using sockets only on Linux or Mac systems with x86 family processors, we would be OK if we ignored this. But any serious socket program will need to use these.

Section 3.3 and the start of Section 3.4 discuss some functions we'll need to do socket programming. Read these more carefully. Section 3.4.1 is interesting but not directly relevant.

Section 4 doesn't matter to us much since we won't be translating old socket code, we'll be looking at new socket code that will use the modern functions right from the start.

Section 5 lists and describes all of the important socket functions we need. Read through this section carefully.

Question 3: Run the program `showip.c` from Section 5.1. Try it with some hostnames you are familiar with. Show the output for at least one that has a single IPv4 address and at least one that has multiple IPv4 addresses. (3 points)

Section 6 brings things together with some complete examples. Read the section and example programs.

Run all 4 example programs. Be sure to run the clients and servers on different machines to ensure you are communicating over the network.

Section 7 advances things a bit. Read through all of Sections 7.1 and 7.2. You can safely skim or ignore the remainder of the section.

The remainder of the document is worth a skim.

You now have all of the background you need to write your own socket programs.

Programming Task

Your programming task is open-ended. You should write a simple client and/or server that uses sockets for communication. You may base your programs on any of the sample code from the tutorial site, but be sure to give credit clearly when you have borrowed from those examples.

Submission and Evaluation

This lab will be graded out of 20 points.

By 9:20 AM, Friday, April 13, 2012, submit documented source code and a `Makefile` to allow easy compilation for your program in addition to the answers to the lab questions. All necessary files should be submitted by email to *jteresco@siena.edu*.

Grading Breakdown	
Answers to questions	5 point
`Makefile`(s)	1 point
Program correctness	5 points
Use of socket functions	5 points
Documentation	4 points