CPSC 460/560 Computer Networks

Project 1: Finger Client/Server

Project Objectives:

- Practice socket programming.
- Understand the design of client-server model and multi-client servers.

Project Description:

In this project, you are required to develop one **network finger client** program and one **network finger server** program.

In the client program, the client (i.e., the client executable file name is ***fingerclient***) runs in command line like this: ***fingerclient username@hostname***, which sends "username" to the network finger server (i.e., ***fingerserver***) running on the host specified by ***hostname***, receives the information from ***fingerserver***, and prints out the received information.

In the server program, ***fingerserver*** is a multi-client server which can accept multiple client connections. You use system call "fork()" to fork child processes which do the actual jobs such as information receiving and sending. Instead of developing a new finger service on your own, you need to use the finger service provided by Unix/Linux. In other words, in the server program, you use system call "execl()" or its variants to run the finger daemon. The output information by running the "finger username" is then sent back to the client. In order to direct the output messages by finger service through the socket connection, we need to use system call "dup2()" to redirect the standard output (file descriptor is 1) and error (file descriptor is 2) to the socket. Thus, the output information is naturally written to the socket and finally reaches the client. We assume that the ***fingerserver*** process takes up to 5 concurrent connections.

Background -- Socket Programming:

In order to complete this project you will need to learn and become comfortable with programming sockets. There are a number of functions that you may need to use for this project (some of the functions have been discussed in lectures):

- **Parsing addresses:**
  - **inet_addr()**
    Convert a dotted quad IP address (such as 36.56.0.150) into a 32-bit address.
  - **gethostbyname()**
    Convert a hostname (such as cs1.seattleu.edu) into a 32-bit address.

- **Setting up a connection:**

socket()
Get a descriptor to a socket of the given type
connect()
Connect to a peer on a given socket
getsockname()
Get the local address of a socket

- **Creating a server socket:**
  bind()
  Assign an address to a socket
  listen()
  Tell a socket to listen for incoming connections
  accept()
  Accept an incoming connection

- **Communicating over the connection:**
  read(), write()
  Read and write data to a socket descriptor
  htons(), htonl(), ntohs() , ntohl()
  Convert between host and network byte orders (and vice versa) for 16 and 32-bit values

You can find the details of these functions in the Unix/Linux man pages (most of them are in section 2) and in the Stevens *Unix Network Programming* book, particularly chapters 3 and 4. You also need to read the supplemental socket programming materials including the Socket Programming Tutor I compiled.

<u>Background – finger service:</u>

Linux/Unix provides the finger service.
Run: finger username
Example: **finger zhuy**

```
[zhuy@cs1 ~]$ finger zhuy
Login: zhuy                    Name: (null)
Directory: /home/fac/zhuy           Shell: /bin/bash
On since Wed Jan 14 12:52 (PST) on pts/1 from 10.126.68.31
On since Wed Jan 14 12:41 (PST) on pts/2 from egrn530-1.seattleu.edu
  11 minutes 22 seconds idle
Mail last read Wed Sep 17 23:11 2008 (PDT)
No Plan.
```

If your ***fingersever*** process receives "zhuy" from the ***fingerclient*** process, the ***fingersever*** process needs send the above information back to the ***fingerclient*** by executing "finger zhuy".

“man finger” will give you detailed information about the finger service.

Testing Environment:

You have two Linux severs to test: the first is cs1.seattleu.edu, and the other is css2.seattleu.edu. The css2 server is sitting behind the firewall and cannot be accessed at home without running VPN. The department technician suggests you to run fingerserver at css2 and fingerclient at cs1 for this project. Your accounts on both servers were created if you registered this class in the beginning of this quarter. You may need to make sure if your account has been created ASAP. If not, send a request email to Renny Philipose philipr@seattleu.edu and cc it to me.

In fact, you can test both fingerclient and fingerserver in one single server by running them in two shell windows.

Port numbers between 10000 and 20000 are open in both servers for your project use. In order to avoid port conflicting among your classmates, I suggest you send me an email requesting an ID. Then you can use the ports between 10000 + (ID-1)* 10 and 10000 + ID*10 – 1.

Submission:

- Deadline: 3:40PM Wednesday, January 23, 2013
- Make your files into a tar package, named **project1.tar**, which includes Makefile, all source files, and a README file.
- Run the command to submit:

```
/home/fac/testzhuy/CPSC4560/submit    p1    project1.tar
```

You can submit your project multiple times before the deadline. Only the most recent copy is saved for grading.

Grading:

- The full score for this project is 10 points.
- Failure in compilation and execution will result in a zero point.
- Collaboration results in a zero point in both parties.
- You must have a Makefile in your submission. Otherwise, your submission will not be graded.
- Make your socket programming code robust. Otherwise, you will lose points due to fragile socket programming code.