CPSC 5510 Computer Networks

Project 1: A Simple Client/Server Warmup Project

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:server_port***, which sends "username" to the network finger server (i.e., ***fingerserver***) running on *server_port* at the computer specified by ***hostname***, receives the information from ***fingerserver***, and prints out the received information. No space is between *hostname* and *server_port*.

In the server program, ***fingerserver*** is a multi-client server which can accept multiple client connections. You use system call fork() to create a child process that handles a client's connection and does 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 redirected to the socket and finally reaches the client. Being a multi-client server, the ***fingerserver*** process can handle multiple concurrent connections. Your undergraduate OS course should have prepared you with these skills.

Background – Socket Programming:

There are a number of system calls that you may need to use for this project (some of the system calls have been discussed in lectures):

- **Parsing addresses:**
  - **getaddrinfo()**
    Given node and service, which identify an Internet host and a service, getaddrinfo() returns one or more addrinfo structures, each of which contains an Internet address that can be specified in a call to bind() or connect().

- **Setting up a connection:**
  - **socket()**
    Get a descriptor to a socket of the given type

connect()
Connect to a peer on a given 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/or Google search.

Background – finger service:

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

```
[zhuy@cs1 ~]$ finger zhuy
Login: zhuy                             Name:
Directory: /home/fac/zhuy               Shell: /bin/bash
On since Tue Sep 26 11:04 (PDT) on pts/5 from 10.192.8.126
   5 seconds idle
No mail.
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:

Port numbers between 10000 and 13000 are open in both servers for your project use. In order to avoid port collision with your classmates, you can use your order (let it be x) in class roster on Canvas. Then you can use the ports between 10000 + (x-1)* 10 and 10000 + x*10 – 1. For example, if you are the first one on the roster, then you can use the ports between 10000 + (1-1)*10 and 1000+1*10 – 1.

Understand the Protocol

In this project, TCP is the transport protocol used for data communication between the finger client and server. TCP provides a reliable in-order byte stream and it has no message boundary. The finger client and the finger server thus have to define their own application-layer protocol to communicate with each other. One question you have to answer in your design is: what is your application protocol for the finger client and server processes?

Grading: 10 points

Label	Grading components	Comments
1-A	[2 pts] Complete file submission: all source files , readme, & Makefile	1. Missing Makefile & Readme: -1 pt 2. Makefile does not work: -1 pt
1-B	[5 pts]Functionality: 1. Work with valid username. 2. Work with invalid username. 3. The right messages are displayed by the client program for both valid and invalid user names. 4. It is a multi-client server.	
1-C	[2 pts] Source code quality 1. Resource release (e.g., close sockets) 2. Right way to do data sending/receiving (TCP is a byte stream protocol) 3. No memory leak problems 4. Byte ordering	
1-D	[1 pt] Clean output, no debugging messages. Output the messages exactly coming from finger program	
1-E	Either the program cannot be compiled Or fatal segmentation faults	Zero points for this project!

Submission:

- Deadline: 3:45PM, Thursday, October 5, 2017
- Submit your project code on cs1.seattleu.edu
- In cs1, make your files into a tar package, named **project1.tar**, which includes Makefile, all source files, and a README file. E.g.,
  tar -cvf project1.tar fingerclient.cpp fingerserver.cpp Makefile README
- Run the command to submit:
  $ /home/fac/zhuy/class/CPSC5510/submit p1 project1.tar

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