Practical Assignment #2

Introduction to Computer Networks

Description

Through Practical Assignment #1 to #4, you will build by the end of the semester a simple Unix-based Web server. That server will be developed in C on a Unix-based OS. This simple Web server will be capable of serving one request at a time. To simplify the programming task and to proceed incrementally, we will lead you through the simple Web server implementation in four stages. At the first stage, you have been asked to get on a Unix-based system and practice a number of basic commands to get around the Unix system.

From the second stage and on, you will be implementing towards a simplified HTTP/1.0 (as defined in RFC 1945) Web server. In that, you will implement a Web server, which listens for a Web client at a time. The entire Web server implementation will be divided into three parts: 1) the echoer, 2) the parser, and 3) the responder. One will be implemented based on another.

At the second stage, you will need to implement the echoer part of the server that simply sends back and displays the contents of the received data stream from the client. To test your echoer, you will be asked to use the telnet program as the client, type in random characters, and check whether the echoer is able to send the same character stream back.

Developing Environment

You will be developing your code in C on a Unix-based system. The convention is that you edit your code in a text editor (for example: emacs) and compile with the GNU C compiler by:

```
% gcc -o server server.c
```

gcc	: the GNU C compiler
-o	: the flag to specify the output filename (i.e. the resulting executable code)
server	: the filename you specify for the resulting executable code
server.c	: the filename of your C code for the Web server

After compilation, you may start the server program by:

```
% server &
```

This will start the server and your server will be listening to any client attempt to make a connection. You may check all the processes running under your username by:

```
% ps -u <your username>
```

For example, I might find the list of processes running under user phuang by:

```
% ps -u phuang
   PID TTY        TIME CMD
 24360 pts/5      0:00 server
 14679 pts/5      0:00 tcsh
```

PID	: the process ID
TTY	: controlling terminal of the process
TIME	: cumulative execution time of the process
CMD	: command of the process

Since the server is running in a infinite loop waiting for connections to be made, please be aware that you need to terminate the server process after your are done testing or when you need to run a re-compiled version of the server executable. Termination of the process can be done by:

```
% kill 24360
```

An example server.c is available from http://nslab.ee.ntu.edu.tw/courses/intro-cn-fall-06/practical/server.c. In this example, the server prints "Hello, world!" upon accepting a connection. Suppose the server is started on ccws3.ee.ntu.edu.tw port 3490. "Hello, world!" will be sent to and displayed on the machine that we use to telnet to the server.

```
    % telnet ccws3.ee.ntu.edu.tw 3490
    Trying 140.112.18.73...
    Connected to ccws3.ee.ntu.edu.tw.
    Escape character is '^]'.
    server: got connection from 140.112.18.7
    Hello, world!
    Connection closed by foreign host.
```

You are free to use this server.c as a base to develop the echoer for PA#2. We expect your echoer to print whatever stream that is typed in.

```
    % telnet ccws3.ee.ntu.edu.tw 3490
    Trying 140.112.18.73...
    Connected to ccws3.ee.ntu.edu.tw.
    Escape character is '^]'.
    Hi, test, 1, 2, 3
    Hi, test, 1, 2, 3
    …
```

You could also try to request to your echoer from any Web browser and see from the echoer output the exact message sent from the Web browser:

```
    http:// ccws3.ee.ntu.edu.tw:3490/
```

Submission

You will rename your server.c following the assignment naming convention to, for example, P2-B78901001-1223-1843.c. Please note that the code for Practical Assignment #2 will be P2. Then, upload the file to the ftp server (140.112.42.157) by the due date and time.

Caution

As you are developing the code, remember that you **SHOULD NOT** be serving through the standard port 80, so you need to specify the port number that your echoer’s running on in the test telnet session. For example, if your machine's name is *host.someschool.edu*, your server is listening at port *6789*, you would telnet to the specific port:

```
% telnet host.someschool.edu 6789
```

or type in the URL as follows from the web browser:

```
http://host.someschool.edu:6789/index.html
```

If you omit ":6789", the browser will assume port 80 which most likely will not have a server listening on it. To avoid conflicts, each student will be using a specified port number as shown below in the port assignment file provided by the TA.