CMPUT 379 Assignment 2, Fall 2014

Due Friday, Nov 07 (submit by 11:55 pm)

Worth 60 marks (however, 15% of total course weight)
(threads & processes, IPC, sockets, synchronization)

Objective

In this assignment you will implement a web server which implements a minimal subset of the HTTP 1.1 protocol. This assignment will expose you to many issues related to writing daemons and network servers. Your task is to implement this web server in two different ways (please see next section).

The web server is started with three command line arguments. The first argument is a TCP port on which it will listen to service requests made by clients. The second argument is a directory from where the server serves documents to the clients. The last argument is a log file in which the server logs all transactions with all clients.

The Web Servers

You must implement two servers. **`server_f`** and **`server_p`**. Each server is invoked in exactly the same way. An example invocation:

```
server_f   8000   /some/where/documents   /some/where/logfile
```

would invoke `server_f` listening on the local machine on port 8000, serving up documents from the directory `/some/where/documents`, and logging transactions to `/some/where/logfile`.

- The servers must properly *daemonize* on startup.
- The servers place no limit on the size of files they can send to the clients.
- If the supplied command-line argument is not a valid directory, then the servers print an error message and exit.
- **`server_f`** must be implemented using processes and `fork()`. That is, `server_f` must fork a new child process to service each new request it gets from the network.
- **`server_p`** must be implemented using `pthreads`. In `server_p` each request from the network must be serviced in a different posix thread.

The servers accept and respond to client requests as documented in the file *ClientRequestFormat.pdf*.

The server logs all connections to themselves as documented in the file *LogFormat.pdf*. Even if the client disconnects before the server can serve the response, the server should still log the request.

Marking

This assignment will be marked out of 60 marks.

- `server_f` implementation will be worth 30 marks
- `server_p` implementation will be worth 30 marks

Deliverables

You must deliver the code for two servers, as well as a `Makefile` to build them, where the command "`make all`" will build both servers.

Mandatory Coding Style

Consistent and readable C coding style is *very* important to spotting bugs and having your code readable by others. You **MUST** comply with following coding style:

- Proper indentation should be included
- TAB should be used for indentation
- NO line may be longer than 80 chars
- All functions should be prototyped
- All variables are declared at the start of a block
- C style comments must be used (not C++ style)

Important: please always remember to remove from your machine the running server daemon(s) before logging off. Generally, make sure there are no stray processes left on your machine before leaving.