Washington State University Vancouver

Systems Programming - CS 360

Assignment 8

Instructor:
Ben McCamish

Overall Assignment - 100 points

You will be creating one C program, `assignment8.c`, using the socket routines discussed in the slides.

Your program will have the following interface:

```
./assignment8 client address

Where: client is the first parameter specifying you wish to create a client that will attempt
to connect to "address". Address may be some hostname like "localhost" or an IP like "127.0.0.1". No quotes o

./assignment8 server

Where: server instructs the program to run a server forever. You should bind your server to port
49999. It is also suggested (read: required) that you use the socket options in the slides to
allow for a server to be spun up immediately upon being killed without the port already in use error.
```

The server process will listen on an ephemeral TCP port for a connection. When it receives a connection, it logs the hostname of the client and the number of times a client has connected to `stdout` and writes the current date and time to the client (as text) via the socket connection. Example output: `localhost 1\n` for a single connection so far from localhost. **Suggestion:** start by writing some string of your own choosing to the client. The server should behave as discussed in class. That is, it should fork when a new connection is received and go back to waiting for more connections.

The client process takes one additional argument on the command line. That argument is the server's host name or IP address (in dotted decimal). It makes a TCP connection to the indicated host and appropriate port number (or indicates the error, if it cannot) and then writes the date received from the server to `stdout`. The output must be exactly 18 bytes long (enough to fit the date minus the last character, but no more) with the addition of a newline at the end (totaling 19 bytes). Example: `'Wed Apr  1 12:30:2\n'` would be April 1st at 12:30 and 20 some seconds PM.

Use port number 49999.
This assignment might be a bit more difficult, since you will not be getting any hints from Autolab. This means you should make your error checking robust, check tests locally, and make sure that you are thinking of the edge cases. I provide the input, but you should be able to calculate the exact output.

Specifications and Restrictions

- (80 points) Your program should use the non-obsolete versions of the code and match my output exactly. Follow the descriptions of the output outlined above.
- (20 points) Must be robust, including error catching. You must catch errors and print out an appropriate error message containing the `errno` and the message produced by that error. This means you will need to use `errno.h` and `string.h`, libraries at least. Remember to handle the `getnameinfo` and `getaddrinfo` errors differently. Your error format should be `Error: <error string here>` exactly.
- See library routines `time()` and `ctime()` to obtain and format the time of day.

What to turn in on Autolab:

- assignment8.c