CS2003 week 3 practical:
Client-server programming

Due 21:00 Friday Week 5
25% of practical mark for the module

Goal

To build a client-server system modelled on the World Wide Web (but a lot simpler).

Background

The World Wide Web is a distributed information system, as we all know. It was originally designed to serve HTML pages, but has been extended to serve other data formats as well.

Those of us with long memories remember other distributed information services that pre-date the web. Examples include WAIS[1] (Wide-Area Information Server,) and Gopher[2]. Like the web, these served pages over the internet. Unlike the internet, they (a) didn't really do hypertext and (b) failed to catch on. They did, however, have a couple of significant advantages: (a) they were text-based, making them easy to read on any device – including as text-to-speech for the blind, which modern web browsers have real problems with; and (b) they were also based on very simple exchange protocols.

Specification

Write a simple client-server system that serves text-only pages. Your server should provide a service loop that does the following:

- Set up a server socket on a well-known port
- Accept a connection
- Read the name of a file from the client
- Look up that file in the same directory as the server was set-up in
- If the file is there, send its contents back; if not, send back a suitable error
- Close the connection and go back to waiting on client connections

Your client should read the name of a document and request it from the server, displaying the result or an error message.

This is a very basic specification. Feel free to add additional features for extra credit. You might, for example, allow the client to access multiple different servers on different machines or ports (easy/medium); provide some form of hyperlinking, so users can “click links” (hard); provide a more complex protocol for requesting documents and handling errors (medium); or provide more complex server configurations, such as getting files from several directories or locations (easy/medium).

---

1 https://en.wikipedia.org/wiki/Wide_area_information_server
2 https://en.wikipedia.org/wiki/Gopher_%28protocol%29

Requirements

You should submit three elements to MMS:

1. Your client and server programs
2. A short (approximately 500—1000 words) report describing how you designed and built the system and showing appropriate evidence of testing, at least showing successful and unsuccessful requests for pages

Marking

The practical will be graded according to the guidelines in the student handbook:

https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html

Hints for completing the practical

(Use or ignore as you see fit.)

Always get the basics working first. Debug the connection establishment and management first. Make sure connections are opened and closed properly.

Tests need to be reproducible. How will you test the client and the server reproducibly? Can you test them separately? Make sure you do actually test both sides in a way that shows you can identify where the errors are. This can be made easier by server-side (or indeed client-side) logging.

Hyperlinking is the thing that really got the web going. Since a text-only system doesn't really support clicking links, some text-only browsers mark-up links with numbers and let the user enter the number of a link to follow it. For example, you might see a page like this:

```
This page contains two embedded hyperlinks (1) of which this is
the second (2).
```

and the user can press 1 or 2 to follow the link. Of course this means you need to be able to describe the target of the link somehow, either in the file itself or in another file that just stores link descriptions: there are lots of options.

Errors are a real problem for distributed systems so having a protocol that handles them well is important. Examples include reporting fatal errors, pages that have been moved to other servers and the like.