Networked Systems Architecture 3: Laboratory Task 3

Dr. Colin Perkins

13 February 2008

The aim of the third laboratory task is to generalise and extend the simple web server you have developed. There are two main parts to this: extending your server to accept multiple sequential connections, and extending it to accept multiple requests per connection.

Multiple sequential connections: You will recall that the `accept()` function returns a file descriptor for the newly open connection, leaving the file descriptor of the listening socket untouched. A server may therefore accept a new connection, read the request, send its response, and close the connection, all without disturbing the listening socket. The first part of this task is to use this feature, extending your web server to accept and serve multiple connections, one after the other, rather than exiting after serving a single connection. Don't forget to increase the backlog in the `listen()` call, so multiple connections can be waiting.

Multiple requests per connection: Forcing a web browser to open a new TCP connection for each request is inefficient when multiple files are retrieved from a single web server. To avoid this inefficiency, HTTP allows several requests to be sent on a single connection. If the server *does not* include a `Connection: close` header in its response, the client can keep the connection open, and may send additional HTTP requests to the server. To allow the client to distinguish data from multiple requests, the server must include a `Content-Length:` header specifying the size of each response's data in bytes. You can retrieve the size of a file using the `fstat()` function:

```
#include <sys/stat.h>
...
struct stat    fs;
int            fd = open(filename, "O_RDONLY");
...
if (fstat(fd, &fs) == -1) {
     // Error...
}
printf("file size = %d\n", fs.st_size);
```

Update your web server to support multiple requests per connection, only closing the connection when the client does so (the `read()` function will return zero when the connection is closed). Demonstrate that this works using a standard web browser (e.g. Firefox) by printing details of each request handled by the server.

If you complete these two extensions, an optional third part of this task is to modify your server to detect the extension of the filename, and send an appropriate `Content-Type:` header:

Filename:	Content-Type:
*.html, *.htm	`Content-Type: text/html`
*.txt	`Content-Type: text/plain`
*.jpg	`Content-Type: image/jpeg`
(unknown)	`Content-Type: application/octet-stream`

(hint: use `strrchr()` to find the last `.` in the filename). Build a simple website containing images to test this. A worked solution to this task will be provided next week, when the final task is distributed.