COMP3301/COMP7308 Assignment 1

- Due: 8pm Friday 21 August 2015
- Revision: $Revision: 37 $

1 mirrord(8) HTTP Server

This assignment asks you to write your own basic web server.

The web server makes files in a specified directory available over HTTP. It is to be implemented as a non-blocking event driven daemon on OpenBSD using its `libc`, `libevent` and Joyent's `http-parser`.

The purpose this assignment is to expose you to event driven programming, working against existing APIs and libraries, and the use of the OpenBSD code style and tool chain.

This is an individual assignment. You should feel free to discuss aspects of C programming and the assignment specification with fellow students. You should not actively help (or seek help from) other students with the actual coding of your assignment solution. It is cheating to look at another student's code and it is cheating to allow your code to be seen or shared in printed or electronic form. You should note that all submitted code may be subject to automated checks for plagiarism and collusion. If we detect plagiarism or collusion, formal misconduct proceedings will be initiated against you. If you're having trouble, seek help from a member of the teaching staff. Don't be tempted to copy another student's code. You should read and understand the statements on student misconduct in the course profile and on the school web-site: http://www.itee.uq.edu.au/itee-student-misconduct-including-plagiarism

2 Specifications

2.1 Code Style

Your code is to be written according to OpenBSD's style guide, as per the `style(9)` man page.

2.2 Compilation

Your code is to be built on an amd64 OpenBSD 5.7 system. It must compile a binary called `mirrord` as a result of running `make` in the root directory of your submitted assignment.

2.3 Invocation

When run with no arguments, or extra or unknown options, `mirrord` should print a usage message to `stderr`:

```
usage: mirrord [-46d] [-a access.log] [-l address] [-p port] directory
```

and terminate with a non-zero exit code.

`mirrord` takes the following command line arguments:

-4 Force `mirrord` to use IPv4 addresses only.
-6 Force `mirrord` to use IPv6 addresses only.
-a access.log Log completed requests to the specified file. By default `mirrord` will not log requests.
-d Do not daemonise. If this option is specified, `mirrord` will run in the foreground and log access requests to `stdout`. By default `mirrord` will daemonise.
-l address Listen on the specified address. By default `mirrord` listens on wild card addresses.
-p port Listen on the specified port. By default `mirrord` listens on the port indicated in the `http` service description; see `services(5)`
directory The directory `mirrord` will use as the base for files it will serve.

If `mirrord` is unable to use the specified directory, listen on its addresses, or write to the specified access log file, it should generate an appropriate error and terminate with a non-zero exit status.

2.4 Functionality

- `mirrord` should listen on TCP sockets with the addresses specified by the command line arguments for HTTP requests
  - it must support both IPv4 and IPv6 sockets
  - addresses may be numeric IP addresses or host names
  - ports may be numeric or a name listed in `/etc/services`
- Repeated specification of command line options may overwrite previously set values
- `mirrord` is only required to implemented a simplified HTTP/1.0
  - it only has to handle a single HTTP request per TCP connection
  - it does not need to implement keep-alives or chunked encoding
  - it does not need to support virtual hosts and can ignore the `Host` header
  - it does not need to handle URL decoding, ie, it is not required to handle file names with spaces encoded with %20 or +, and is therefore not required to handle file names with spaces at all
  - it does not need to handle `Range`, `If-Match`, `If-Modified-Since`, `If-None-Match`, `If-Range`, or `If-Unmodified-Since` request headers
- URLs are mapped to the directory specified on the command line, eg, if `mirrord` was started as `mirrord /var/www/htdocs`, a request for `/a/file` should serve the content of `/var/www/htdocs/a/file`
  - it must not serve files outside the directory it was invoked with
  - it can serve symbolic links within the directory that link to files outside the webroot
- it must only support the `HEAD` and `GET` HTTP methods
- it may return the following HTTP response codes:

  200 OK The requested file was found and can be served
  400 Bad Request `mirrord` should return this if it cannot interpret the HTTP request from the client.
  403 Forbidden `mirrord` should return this if it does not have permission to open the requested file
  404 Not Found The requested file does not exist or is not a file or symbolic link to a file
  405 Method Not Allowed The client made a request with an unsupported method
  500 Internal Server Error A transient system error prevents the handling of the current request from proceeding, e.g., if a required memory allocation or file descriptor creation failed
- HTTP bodies are optional for responses except those for file requests
- `mirrord` responses must include at least the following HTTP headers to all requests:

  - a `Date` header which contains the current system time in RFC822 format, e.g., `Tue, 7 Jul 2015 00:10:31 GMT`
  - a `Server` header in the format `mirrord/sXXXXXXX` where sXXXXXXX is your student user name
  - a `Connection` header which contains `close`

- `mirrord` responses to successful requests for files must include at least the following HTTP headers:
  - a `Content-Length` header which includes the number of bytes in the requested file.
  - a `Last-Modified` header which includes the modification time of the requested file in RFC822 format

- Response headers must be correctly terminated by "\r\n"

- `mirrord` access log format should be 'remote_addr [rfc822date] "request_method request_url" response_code data_bytes'
  - eg, a successful request would look like `127.0.0.1 [Tue, 7 Jul 2015 00:10:31 GMT] "GET /index.html" 200 2259`
  - `mirrord` should append to the specified log file
  - header output does not count to the data bytes counter
  - all connections should generate a log entry
  - if a client disconnects before sending a HTTP request, the method and URL should be each represented by "-"
  - if a client disconnects before an appropriate HTTP response can be generated, you may log 444 as the response code
  - eg, an unsuccessful request would look like `127.0.0.1 [Tue, 7 Jul 2015 00:11:40 GMT] "- -" 444 0`

2.5 Required Dependencies

`mirrord` must only use the following libraries and the APIs they provide to implement the above functionality.

2.5.1 `libc`

`libc` refers to the ISO or POSIX standard C library provided by UNIX and UNIX like operating systems. A `libc` contains the APIs required by `mirrord` for such tasks as resolving host and service names to IP addresses and ports, creating network sockets and connections, and opening and reading files.

2.5.2 `libevent`

According to http://libevent.org/, `libevent`:

> ... provides a mechanism to execute a callback function when a specific event occurs on a file descriptor or after a timeout has been reached. Furthermore, libevent also support callbacks due to signals or regular timeouts.

OpenBSD ships with `libevent` 1.4 with local patches. It is this version of `libevent` in the base system what you are required to write `mirrord` against.

2.5.3 `joyent/http-parser`

To quote the README.md:

> This is a parser for HTTP messages written in C. It parses both requests and responses. The parser is designed to be used in performance HTTP applications. It does not make any syscalls nor allocations, it does not buffer data, it can be interrupted at anytime. Depending on your architecture, it only requires about 40 bytes of data per message stream (in a web server that is per connection).

The code and documentation for `http-parser` can be found at https://github.com/joyent/http-parser. A copy of the `http-parser` repository is in `courses/COMP3301` on the materials drive on the EAIT labs network file server. It is available on `moss.labs.eait.uq.edu.au` or `lichen.labs.eait.uq.edu.au` at `/home/material/courses/COMP3301/http-parser`.

Howerver, you must include your own copy of `http-parser` as part of your assignment submission.

2.6 Restrictions

- `mirrord` must operate as a single process/thread, therefore it may not fork (after daemonising) or create threads.

2.7 Recommendations

The focus of this assignment is event driven programming, not on creating new implementations of common functionality. It is strongly recommended that the following APIs are used as part of your program:

- `getopt(3)` - get option character from command line argument list
- `err(3), warn(3), etc` - formatted error messages
- `getaddrinfo(3), freeaddrinfo(3)` - host and service name to socket address structure
- `getnameinfo(3)` - socket address structure to hostname and service name
- `gai_strerror(3)` - get error message string from EAI_xxx error code
- `daemon(3)` - run in the background
- `queue(3)` - implementations of singly-linked lists, doubly-linked lists, simple queues, and tail queues

3 Submission

Submission must be made electronically by committing to your Subversion repository on `source.eait.uq.edu.au`. In order to mark your assignment the markers will check out `ass1` from your repository. Code checked in to any other part of your repository will not be marked.

As per the `source.eait.uq.edu.au` usage guidelines, you should only commit source code and Makefiles

The due date for this assignment is 8pm on Friday the 21st of August 2015. Note that no submissions can be made more than 120 hours past the deadline under any circumstances.

4 Marking Scheme

WARNING If your code doesn't compile and run, you won't get credit for it.

1. For this assignment, break the functionality into stages or pieces that you can work on independently.
2. Get one piece fully functional and thoroughly tested.
3. Go on to the next piece.

4. If you've tried to implement everything but haven't completed any pieces, you won't get credit for anything.

Individual functionality will be checked, tested, and marked.

For each piece of functionality, percentages of marks will be allocated as:

- 0 - 20% Function not implemented, or no working code
- 20 - 40% Some simple but incorrect functionality
- 40 - 60% Significant problems, such as frequent crashes or infinite loops
- 60 - 80% Moderate problems, fails relatively often
- 80 - 100% Few or no problems

Total marks out of 100:

- Makefile works, and `mirrord` compiles successfully [5]
- Coding style, readability, organization [10]
- Command line argument processing and `usage()` [5]
- Listening socket creation [5]
- specified HTTP responses and headers are generated [5]
- `mirrord` correctly serves files [10]
- `mirrord` is able to handle concurrent connections [20]
- `mirrord` generates access logs [5]
- communications errors are handled correctly [10]
- `mirrord` resists requests for files outside the web root directory [5]
- `mirrord` doesn't leak resources [5]
- `mirrord` correctly logs unsuccesful requests [5]

5 Testing

A set of automated tests will be provided. These will test a subset of all features. A wider set of tests will be used in final testing. It is your responsibility to ensure your program meets all aspects of the specification.

6 Revisions

Changes to the assignment specification can be reviewed at https://source.eait.uq.edu.au/viewvc/comp3301-pracs/2015/assignment1.md?view=log.