Once more, from the top.

The purpose of this program is to serve as a web proxy. It will take a GET or HEAD request from wget, curl, or a browser, and then retrieve its intended data through OpenSSL, and then return it to the client. It must be able to function with multiple clients at once, and the proxy server is never supposed to die. Like always, before I explain the core components, I'll provide the starting command.

bin/myproxy listen_port forbidden_sites_file_path access_log_file_path

"myproxy" starts the proxy server.

listen_port sets the port that the proxy server will listen on.

forbidden_sites_file_path is the path to a user-created file that contains a list of hostnames or IP addresses. Should the proxy receive any request to those websites, it will refuse to connect, and will instead return a 403 Forbidden response. **In addition, as the proxy server is to constantly remain open, sending Control+C will cause the server to reload this file, in case any new urls have been added. You must close the program manually such as through killall -9 myproxy to close it.**

access_log_file_path is the path to file-which will be created if it does not exist-that stores every past request given to the proxy. It includes the time, the client's IP address, the first line of the request, the status code, and the number of bytes written to the client.

Finally, a **disclaimer. I based my code off the textbook we were assigned for our class, "UNIX Network Programming Volume 1, Third Edition: The Sockets Networking API", and as such, the server code is mostly unchanged from the examples given in Chapter 8. We were permitted to do this.**

Additionally, I used this specific page to figure out how to print the current time and date: https://www.techiedelight.com/print-current-date-and-time-in-c/ .

I also used this website to figure out how to establish my OpenSSL connection: https://aticleworld.com/ssl-server-client-using-openssl-in-c/

Finally, again, I used this website to figure out sscanf in a loop: https://www.daniweb.com/programming/software-development/threads/93264/sscanf-in-a-loop

Tests:

Test 1, a simple curl to example.com:

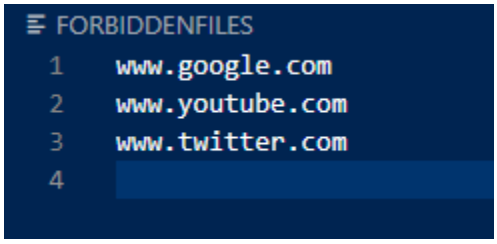

```
curl -x http://127.0.0.1:11037/ http://www.example.com
<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvet
ica Neue", Helvetica, Arial, sans-serif;

    }
    div {
        width: 600px;
        margin: 5em auto;
        padding: 2em;
        background-color: #fdfdff;
        border-radius: 0.5em;
        box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
    }
    a:link, a:visited {
        color: #38488f;
        text-decoration: none;
    }
    @media (max-width: 700px) {
        div {
            margin: 0 auto;
            width: auto;
        }
    }
    </style>
</head>

<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is for use in illustrative examples in documents. You may use this
    domain in literature without prior coordination or asking for permission.</p>
    <p><a href="https://www.iana.org/domains/example">More information...</a></p>
</div>
</body>
</html>
bash-4.2$
```

Test 2, a curl to a forbidden website, www.twitter.com:



Test 3, 2 successes (the wgets) and 2 intentional failures (the first one sends a CONNECT request through https, and the second one has an invalid port defined):

```
bash-4.2$ wget http://www.example.com -e use_proxy=yes -e http_proxy=127.0.0.1:11037 & wget -S --sp
ider http://www.examples.com -e use_proxy=yes -e http_proxy=127.0.0.1:11037 & curl -x http://127.0.
0.1:11037/ https://www.example.com & curl -I -x http://127.0.0.1:11037/ http://www.example.com:445
```

```
bash-4.2$ bin/myproxy 11037 FORBIDDENFILES finale/log
501, Not Implemented.
Could not connect to server.
```

```
2022-03-10T02:53:05.943Z 127.0.0.1 "CONNECT www.example.com:443 HTTP/1.1" 501 0
2022-03-10T02:53:05.995Z 127.0.0.1 "GET http://www.example.com/ HTTP/1.1" 200 1626
2022-03-10T02:53:05.999Z 127.0.0.1 "HEAD http://www.examples.com/ HTTP/1.1" 200 603
2022-03-10T02:53:20.952Z 127.0.0.1 "HEAD http://www.example.com:445/ HTTP/1.1" 500 0
```

index.html.23

```
<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;

    }
    div {
        width: 600px;
        margin: 5em auto;
        padding: 2em;
        background-color: #fdfdff;
        border-radius: 0.5em;
        box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
    }
    a:link, a:visited {
        color: #38488f;
        text-decoration: none;
    }
    @media (max-width: 700px) {
        div {
            margin: 0 auto;
            width: auto;
        }
    }
    </style>
</head>

<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is for use in illustrative examples in documents. You may use this
    domain in literature without prior coordination or asking for permission.</p>
    <p><a href="https://www.iana.org/domains/example">More information...</a></p>
</div>
</body>
</html>

```

Test 4, intentional error cases:

Test 4.1, invalid file path, business as usual:

```
bash-4.2$ bin/myproxy 11037 FORBIDDENFILEStheyarenotreal finale/log
Invalid forbidden_sites_file_path 2.
```

Test 4.2, incorrect number of arguments as always:

```
bash-4.2$ bin/myproxy 11037 FORBIDDENFILEStheyarenotreal finale/log Hello
Not enough/too many initial parameters.
```

Test 4.3, a port that doesn't exist.

```
bash-4.2$ bin/myproxy -1 FORBIDDENFILES finale/log
Positive port required.
```

Frank Gomez, CSE 156

Test 5: It’s a lot of calls to example.com. There’s also an https request in there, a 301, and a 400.

```
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ https://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.youtube.com/dsfasdafsdfsasdafd &
curl -x http://127.0.0.1:11037/ http://google.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
```

```
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/ &
curl -x http://127.0.0.1:11037/ http://www.example.com/
```

```
[58]   Done                    curl -x http://127.0.0.1:11037/ http://www.example.com/
[59]   Done                    curl -x http://127.0.0.1:11037/ http://www.example.com/
[60]   Done                    curl -x http://127.0.0.1:11037/ http://www.example.com/
[61]   Done                    curl -x http://127.0.0.1:11037/ http://www.example.com/
[62]   Done                    curl -x http://127.0.0.1:11037/ http://www.example.com/
[63]   Done                    curl -x http://127.0.0.1:11037/ http://www.example.com/
[64]   Done                    curl -x http://127.0.0.1:11037/ http://www.example.com/
[65]   Done                    curl -x http://127.0.0.1:11037/ http://www.example.com/
[66]-  Done                    curl -x http://127.0.0.1:11037/ http://www.example.com/
[67]+  Done                    curl -x http://127.0.0.1:11037/ http://www.example.com/
bash-4.2$
```

```
bash-4.2$ bin/myproxy 11037 FORBIDDENFILES finale/log
501, Not Implemented.
Forbidden site attempted access.
```

```
2022-03-10T03:06:46.309Z 127.0.0.1 "CONNECT www.example.com:443 HTTP/1.1" 501 0
2022-03-10T03:06:46.345Z 127.0.0.1 "GET http://www.youtube.com/dsfasdafsdfsasdafd HTTP/1.1" 403 0
2022-03-10T03:06:46.349Z 127.0.0.1 "GET http://www.example.com/ HTTP/1.1" 200 2107
2022-03-10T03:06:46.396Z 127.0.0.1 "GET http://www.example.com/ HTTP/1.1" 200 2123
2022-03-10T03:06:46.413Z 127.0.0.1 "GET http://google.com/ HTTP/1.1" 301 709
2022-03-10T03:06:46.418Z 127.0.0.1 "GET http://www.example.com/ HTTP/1.1" 200 2123
2022-03-10T03:06:46.419Z 127.0.0.1 "GET http://www.example.com/ HTTP/1.1" 200 2107
2022-03-10T03:06:46.419Z 127.0.0.1 "GET http://www.example.com/ HTTP/1.1" 200 2107
2022-03-10T03:06:46.420Z 127.0.0.1 "GET http://www.example.com/ HTTP/1.1" 200 2122
2022-03-10T03:06:46.420Z 127.0.0.1 "GET http://www.example.com/ HTTP/1.1" 200 2107
2022-03-10T03:06:46.442Z 127.0.0.1 "GET http://www.example.com/ HTTP/1.1" 200 2123
2022-03-10T03:06:46.459Z 127.0.0.1 "GET http://www.example.com/ HTTP/1.1" 200 2107
2022-03-10T03:06:46.459Z 127.0.0.1 "GET http://www.example.com/ HTTP/1.1" 200 2123
2022-03-10T03:06:46.478Z 127.0.0.1 "GET http://www.example.com/ HTTP/1.1" 200 2107
2022-03-10T03:06:46.483Z 127.0.0.1 "GET http://www.example.com/ HTTP/1.1" 200 2106
2022-03-10T03:06:46.493Z 127.0.0.1 "GET http://www.example.com/ HTTP/1.1" 200 2107
2022-03-10T03:06:46.500Z 127.0.0.1 "GET http://www.example.com/ HTTP/1.1" 200 2123
2022-03-10T03:06:46.501Z 127.0.0.1 "GET http://www.example.com/ HTTP/1.1" 200 2122
2022-03-10T03:06:46.502Z 127.0.0.1 "GET http://www.example.com/ HTTP/1.1" 200 2123
2022-03-10T03:06:46.503Z 127.0.0.1 "GET http://www.example.com/ HTTP/1.1" 200 2107
2022-03-10T03:06:46.510Z 127.0.0.1 "GET http://www.example.com/ HTTP/1.1" 200 2107
2022-03-10T03:06:46.523Z 127.0.0.1 "GET http://www.example.com/ HTTP/1.1" 200 2123
2022-03-10T03:06:46.524Z 127.0.0.1 "GET http://www.example.com/ HTTP/1.1" 200 2123
2022-03-10T03:06:46.533Z 127.0.0.1 "GET http://www.example.com/ HTTP/1.1" 200 2123
```

And that marks the end of networks.