

Lab 2: A Concurrent Web Server (Week 2)

Lab due: October 5th @ 11:59pm

Let's look at what we are implementing...

Program Workflow

1. Read the arguments, find your document root, bind to the specified port, and begin listening for incoming connections.
2. Accept a connection, and:
 - a. **week 1:** hand the socket off to a function that handles the remaining steps.
 - b. **week 2:** pass the socket to a new thread for concurrent processing.
3. Receive and parse a request from the client.

Program Workflow (continued)

4. Look for the path that was requested, starting from your document root (the second argument to your program). One of four things should happen: **You might want to make each of these cases a separate function!**
 - a. If the path exists and it's a regular file, formulate a response (with the Content-Type header set) and send it back to the client.
 - b. If the path exists and it's a directory that contains an index.html file, respond with that file.
 - c. **week 2: BONUS** If the path exists and it's a directory that does *NOT* contain an index.html file, respond with a directory listing.
 - d. If the path does not exist, respond with a 404 code with a basic HTML error page. The 404 HTML page can be static and very simple — it just needs to be enough for a user to see a 404 message in a real browser.

Program Workflow (continued)

4. Close the connection and continue serving other clients.

Week 2 Goals

Implement handling multiple clients concurrently using threads and BONUS of returning a directory listing.

Need brush up on threads?

See Dive into Systems Chapter: <http://diveintosystems.org/book/C14-SharedMemory/posix.html>

Run `man pthread_create`

```
int pthread_create(pthread_t *restrict thread,  
    const pthread_attr_t *_Nullable restrict attr,  
    typeof(void *(void *_Nullable)) *start_routine,  
    void *_Nullable restrict arg);
```

Well that looks confusing.... let's look at the version from the Dive Into Systems Textbook...

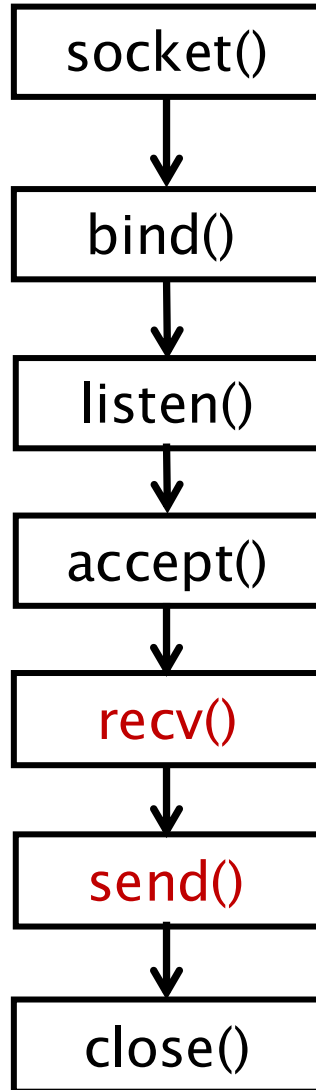
Run man pthread_create

```
int pthread_create(pthread_t *thread,  
                  const pthread_attr_t *attr,  
                  void *(*thread_function)(void *),  
                  void *thread_args);
```

1. How does pthread_create use its inputs?
 - pthread_t *thread
 - pthread_attr_t *attr
 - void *(*thread_function)(void *)
 - void *thread_args
2. What does pthread_create return on success?
3. What does pthread_create return on error?



Server



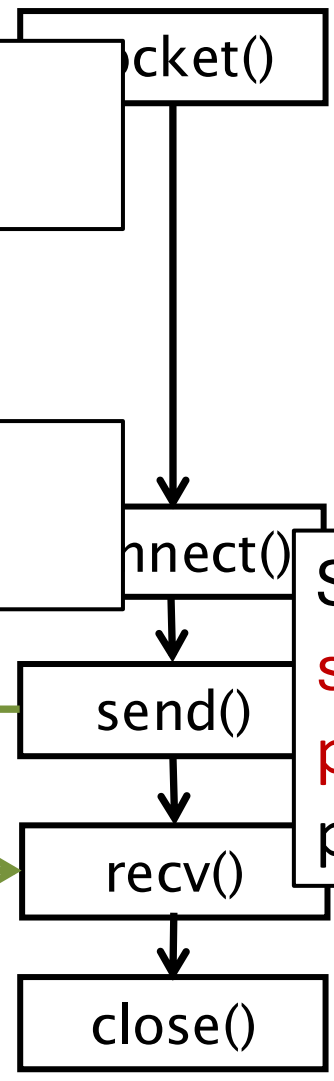
send(), recv()

socket() returns
server_sock

accept() return
per_client_sock



Client



Should you pass
server_sock or
per_client_sock to
pthread_create?



Do any of the threads need to coordinate with each other?

Run `man pthread_detach`

```
int pthread_detach(pthread_t thread);
```

1. How does `pthread_detach` use its inputs?
2. What does `pthread_create` return on success?
3. What does `pthread_detach` return on error?

Let's look at helpful example in starter code
`thread_detach_example.c....`

For bonus...look at what running generic python server does for directories with no index.html

```
cd test_documents  
python3 -mhttp.server 8080
```

In browser: <http://localhost:8080/test/>

If you are not doing the bonus: Your server should return 404 Not Found in this case!

Testing concurrency using telnet...

- In one terminal window: `valgrind ./lab2 8080 test_documents`
- In another terminal window: `telnet localhost 8080` and don't send anything. This simulates a client that is connected but doesn't do anything
- Then try to do your usual testing using telnet, the browser, or wget. It should still work even if you have the one client sitting there doing nothing
- You should test this with more than 1 “do nothing” client.

Checkpoint...

Your server should not exit after
as you handle one request!

1. Run `valgrind ./lab2 8080 test_documents`

2.. In terminal:

```
wget http://localhost:8080/
```

```
md5sum index.html
```

```
md5sum test_documents/index.html
```

3. In a browser go to:

```
http://localhost:8080/vader.jpg
```

```
http://localhost:8080/test/dir/endtoend.pdf
```

4. Did you define helper functions?