# Design Document: Multi Threaded HTTP SERVER

**Scott Fischer id: 1610331**

Goal: Modularize the Server Further to Use Multiple Threads in order to service many clients at once

## Main Data Types

- httpObj{

```
  //Create a struct to parse and sort HTTP request into sections
  char method[6];         //Holds Request Type: PUT or HEAD or GET
  char filename[30];      //Holds what the file we are trying to act on
  char httpversion[9];    //Holds that bit HTTP/#
  ssize_t contentlength;  //length of data we are given or pulling
  int statuscode;         //200,201,400,404,405,500
  uint8_t buffer[BUFFER_SIZE];
  Node* list;
  int local_offset
```

  };

- This holds the data for the server thread to manipulate. This is not shared passed the server thread
- ThreadArg{

```
  int log_offset;
  int log_flag;
  int ID;                 //ID for each server
  int* queue;             //This holds the socket number of each unhandled
request currently given to us
  sem_t request_sem;      //This stops busy waiting with the connector
  sem_t thread_sem;       //This stops the connector from assigning sockets to
busy threads
  sem_t log_sem;          //This is to add atomicity to incrementing offset

  Node* server_queue;     //Holds Server Thread that are currently waiting
  int* socket_list;       //This is where the servers pull the sockets from
  sem_t *server_sems;     //This is the semaphore array that holds sems for
each server

  int log_file;
```

  }

    - This would be the object that is shared and passed between the threads and carries the data from thread to thread

### Main()

Assumptions: We are at least passed the port number. Could also be passed N thread number, and log flag Purpose: Create the socket, and send the requests to the acceptor

- Server Setup Using Starter Code
- Parse Arguements Using getopt()
- Create Writer Thread and Connection Thread
    - writer id: 0, connection id: 1
- Use Arguments to Create Server Threads
    - these will be labeled from 2 to n+1
- Create Object to hold port and queue
- Infinite While Loop to Listen for socket
    - pass client socket fd

**Acceptor Thread:**

Assumptions: A socket file descriptor is passed to it Purpose: To Decide Which Server Thread will be Given the Request

- Pull from the queue and check which thread is open
- Push to open thread
    - This would be pushing to the Linked List server_queue from our data type

**Server Thread:**

Assumptions: Passed Client Socket Description:

- Runs Single Threaded HTTP SERVER CODE
- This is almost a copy and paste of the code from assignment one, however it excludes all the server setup, and adds some calls to semaphores and pushing to a queue for logging

**Writer Thread:**

Assumptions: This thread will be passed the message object
Description: To take Linked Lists and write them to the Log file

- Pulls from a Write Queue and writes to the Log
    - This queue will be a Linked List that is first in first out
    - The Linked List will have data of buffer strings that we got when we read in the data, and now we write that data as hex values to the log file