System Programming Final Report

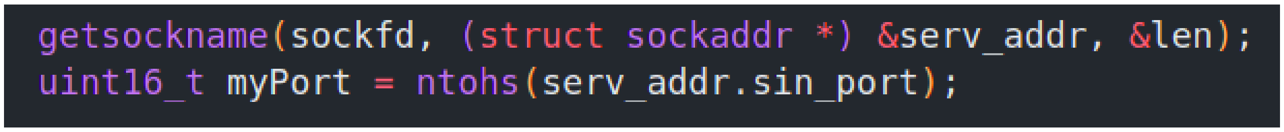
Server: Creates threads and and opens the socket for connections. It accepts it and writes socket fds to queue. Queue size is dynamically changing. Server threads waits for this queue to have elements with monitor. When there is an element, it removes the element and listens. According to the size of the message, it decides if it is client or servant. İf it is servant, it saves it is ip, port number created by servant which is unique and first and last cities the servant must take care of. If the socket is client's, server thread decides which servant must work with it. It checks the city names given by servants, and takes the corresponding servant's port number and sends the request to it. If the given request does not have city name, all servants are awaken.

Servant: Reads data from directory, and saves the part of the data which is between given ranges. Writes them to hashmap using their city name as key. Opens the socket and waits to accept coming requests. When request comes, it creates a new thread and records it pointer to join it when the program ends. And main servant threads goes back to accept state again. Servant threads calculates the number of transactions which is proper for request. When it finds, it sends the result to the server thread which is waiting for the result.

Client: Creates threads as much as request file has requests. They wait for each other, and when all of them are ready, they send the request to the server. Because the server's main thread tries to stay in accept state as much as possible, they all can be handled by the server.

If the client wants to know about city which is not in the dataset, -1 is returned and error message is printed.

Generating unique port number for servant threads are done using getsockname and ntohs functions.



```
getsockname(sockfd, (struct sockaddr *) &serv_addr, &len);
uint16_t myPort = ntohs(serv_addr.sin_port);
```

For counting, saving to queue,removing from queue, saving servant info, mutexes are used. So that race conditions are avoided.

When reading the data from the file, readdir function is used and the directory entries are sorted by name. Scandir function is easier to use because it gives the directory entries in order, it causes memory leaks. So readdir function is preferred.