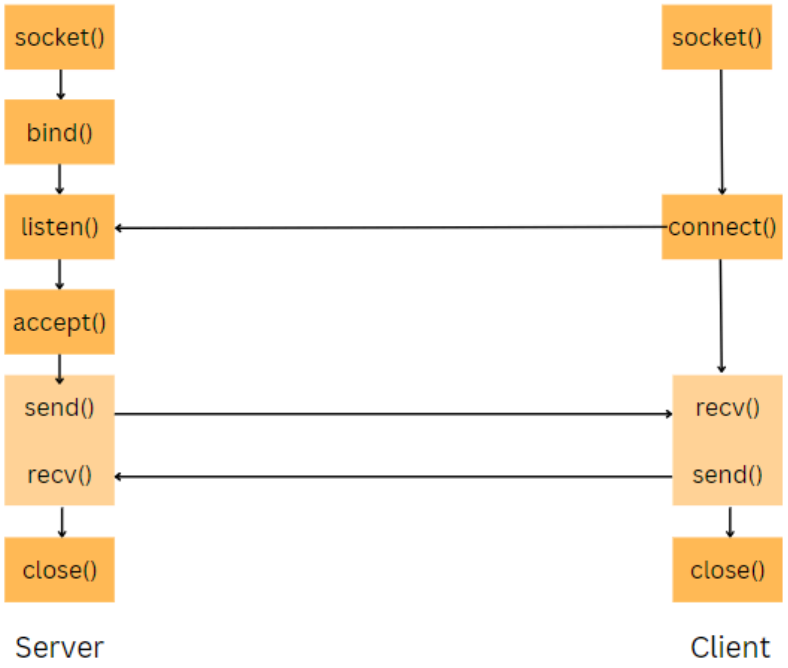
CHAT APPLICATION – CODE FLOW

Socket Programming in C –



1. Socket () – Client and server programs invoke this to create sockets. The function takes 3 parameters – the domain (IPv4/ IPv6), the type (stream/datagram socket) and the protocol.

2. Bind() – The server program invokes it to bind the created socket to its IP address and a Port number.

3. Listen() – The server invokes this method to put the socket in a passive listening mode to catch incoming client connections and places it on a waiting queue.

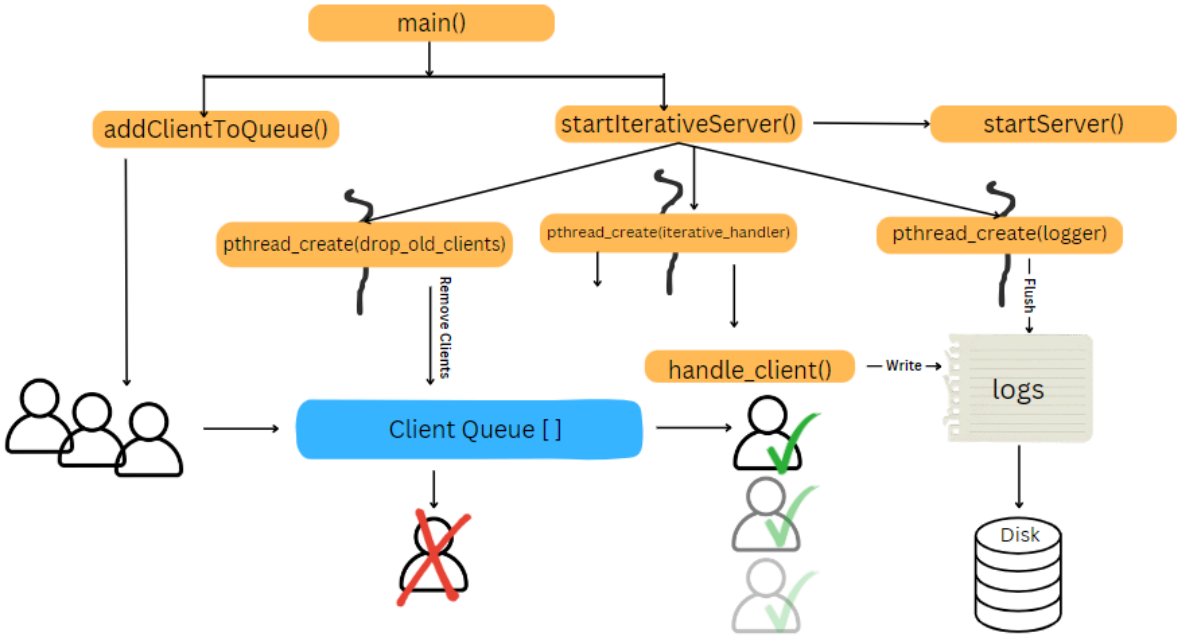
4. Connect() – The client invokes this to connect to the server socket.

5. Accept() – The server accepts a new client connection from the waiting queue.

6. Send() & Recv() – These methods are used by the client and server to send and receive messages.

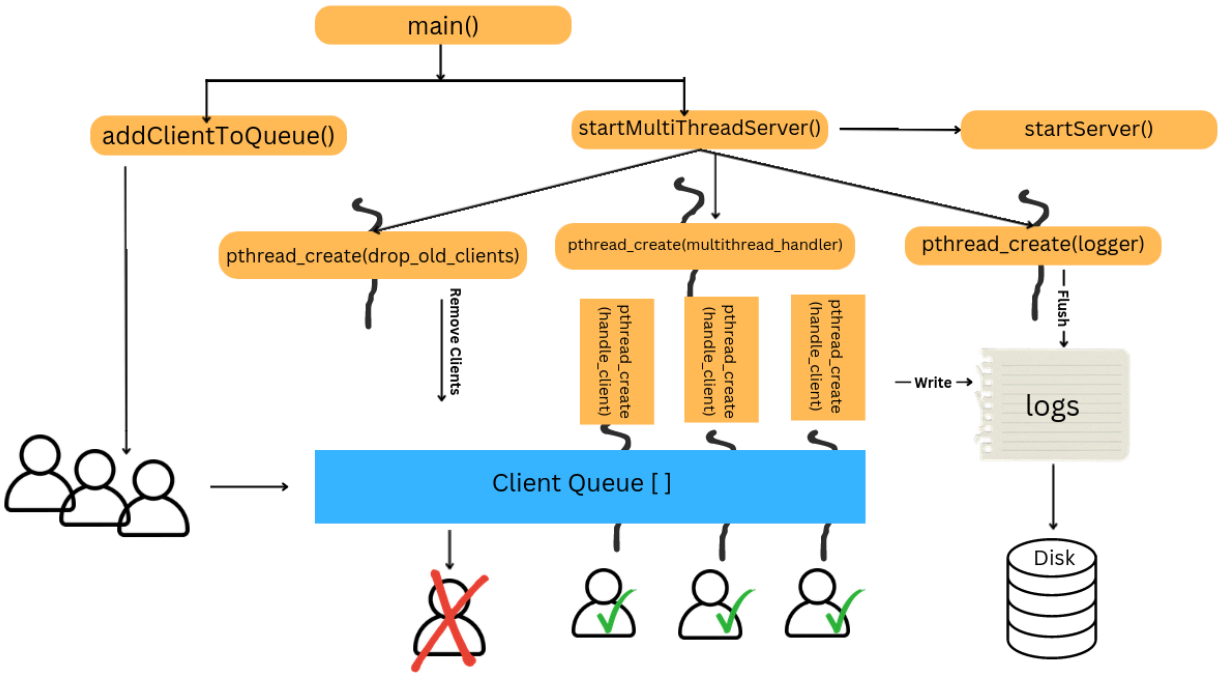
7. Close() – This is invoked by both the client and server to close the connection and release the occupied port numbers.

Approach 1 : Iterative Server



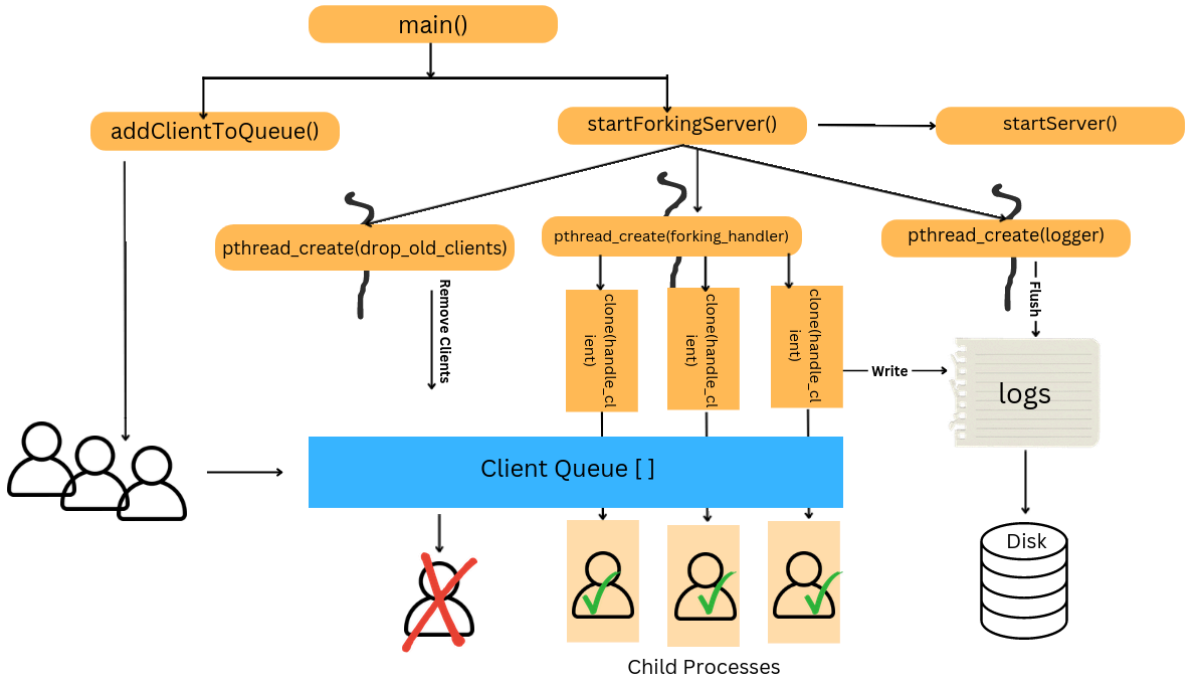
In this approach each client is served iteratively, one after the other. The handle_client() method is called for every new client from the queue, after the connection is closed with the previous client.

Approach 2 : Multithreading Server



In this approach, a new thread that executes the handle_client() method is created for every new client arriving at the queue. The number of concurrent clients served depends on the number of CPU cores.

Approach 3 : Forking Server



In this approach, a new process that executes the handle_client() method is created for every new client arriving at the queue using the clone() system call . The number of concurrent clients served depends on the number of CPU cores.

Functions and Description

void logger(void* arg)	DESCRIPTION -This method writes logs periodically from the log buffer to the log files. It is invoked by a separate thread that runs in parallel. PARAMETERS – None (This method is invoked by the pthread_create() and hence requires the parameter to be of type void*)
int handle_client(void *arg)	DESCRIPTION -This method sends and receives messages from a client using the write and read methods.

	PARAMETERS - void pointer typecasted to point a ClientInfo object.
void *iterative_client_handler(void *arg)	DESCRIPTION - This method serves clients one by one, taking a ClientInfo object from the client queue and passing it to the handle_client() method that serves it. PARAMETERS – None (This method is invoked by the pthread_create() and hence requires the parameter to be of type void*)
void *multithread_client_handler(void *arg)	DESCRIPTION - This method handles multiple clients at a time by creating a separate thread for every new client. The number of concurrent threads is based on the number of cores. PARAMETERS – None (This method is invoked by the pthread_create() and hence requires the parameter to be of type void*)
void *forking_client_handler(void *arg)	DESCRIPTION - This method handles multiple clients at a time by creating a separate process for every new client. The number of concurrent processes is based on the number of cores. PARAMETERS - void* typecasted to int* that holds the ID of the parent process
void *drop_old_connections(void *arg)	DESCRIPTION - This method is run by a separate thread parallely to drop timed out clients from the clients_queue. PARAMETERS – None (This method is invoked by the pthread_create() and hence requires the parameter to be of type void*)
void startServer()	DESCRIPTION - This method initializes the struct Server object and assigns a name for the Server PARAMETERS - None
void addClientToQueue(int new_socket, struct sockaddr_in address)	DESCRIPTION - This method adds incoming client connections to the client_queue before it is served by the server. Otherwise closes the client connection if the client_queue is full PARAMETERS – int new_socket - A number representing the new socket between client and server struct sockaddr_in address - sockaddr_in holding the details of the newly created socket

void startMultiThreadServer()	DESCRIPTION - This method creates 3 threads - 1 thread to handle client requests using multithreading, 1 thread to drop timed out requests, 1 thread for logger PARAMETERS - None
void startForkingServer()	DESCRIPTION - This method creates 3 threads - 1 thread to handle client requests using forking, 1 thread to drop timed out requests, 1 thread for logger PARAMETERS - None
void startIterativeServer()	DESCRIPTION - This method creates 3 threads - 1 thread to handle client requests using sequentially, 1 thread to drop timed out requests, 1 thread for logger PARAMETERS - None