

Forum:[Hack Forums](#)

Thread: [How to \(correctly\) do multithreading in c/c++?](#)

Lynx - January 3, 2022, 1:09 pm

I have been working on a rather high end project which involves socket connections and handling multiple clients at the same time.

I have some doubts and questions about the best way to implement this.

Starting/stopping threads is a rather intensive task to perform. Currently, for every client that connects I am starting two different threads for receiving and sending data. This works fine as of now but I have not tested it with a lot of connections yet.

I am currently using pthread to make threads. (Ignore the fact that I'm not checking for the return value of pthread_create as its a mock)

```
Code {
    Client      *self_ptr;
    pthread_t   recv_worker;
    pthread_t   ping_worker;
    socket_fd = fd;

    // ...

    pthread_create(&recv_worker, NULL, &RecvThread, (void *)self_ptr);
    pthread_detach(recv_worker);
    pthread_create(&ping_worker, NULL, &PingThread, (void *)self_ptr);
    pthread_detach(ping_worker);
}
```

I will high likely switch to std::thread anyways since I've been told its a wrapper around pthread among others meaning its more widely supported.

Let's say I will have 500 clients, will this still work?

In this example threads will only be created/destroyed on client connects/disconnects which doesn't happen so often so I am assuming (and hoping) this isn't that big of a deal.

There is another part of my code where threads will get made and destroyed every second or even faster.

I've read about workers which would constantly be checking a queue, waiting for a "request" to be handled which has to be done in a separate thread, like my program requires.

What do you recommend?

yuumi - January 3, 2022, 3:47 pm

Using a thread per client to read and write data too is a very bad idea. It might work for 100 even several hundred. But if you really want large scale connectivity say in the thousands, 100.000+ then you certainly don't want a thread per client even if you're suspending / resuming them. It's just a bad

non scalable design.

Rather you should read up on async frameworks and depending on the OS your on they might have their own. I've written my own multi connection async library using Microsoft's Overlapped I/O framework. If you're using BSD socket implementations then you can use 'select'.

I think Linux also has epoll.

This is a game I've written in C. It's a multiplayer minesweeper flags game where the objective is to locate mines rather than avoid them. The front end is written in html/JavaScript and uses websockets to connect to the C backend reverseflags.me

The game also uses browser cache to store data allowing the game to be closed and reopened to continue where it left off. So you can't play with yourself using a single browser.