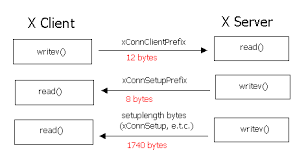
Documentation

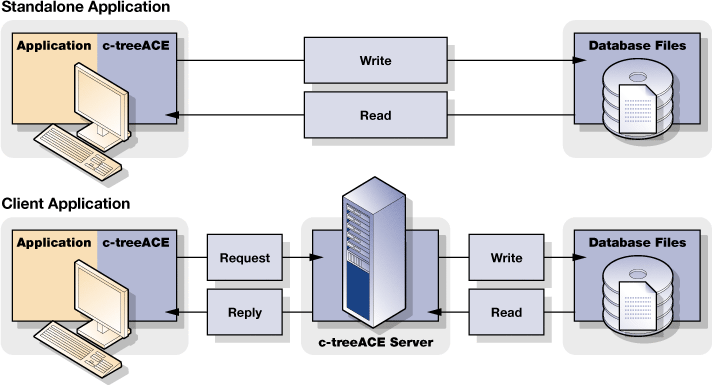
CLIENT.cpp

The client in this program took in six arguments: port, repetitions, nbufs, bufsize, serverIp, and type. It would take the nbufs and bufsize (which had to equal 1500) and then tested three different transfer scenarios.

1. multiple writes: invokes the write( ) system call for each data buffer, thus resulting in calling as many write( )s as the number of data buffers.
2. Writev: which invokes the write() system call for each data buffer. This will in turn call as many write as number of data buffers.



1. Single write: allocates an nbufs-sized array of data buffers, then it sends all of the data buffers at once.

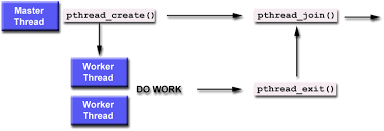


SERVER.cpp

The server program is set up to run and take in two arguments: the port number that is at least 1024 and it isn’t add 1024 to the number to find an open port. The second parameter that is supposed to be passed is the number of repetitions (that needs to match the client repetition number of the clients).

It has a thread interrupt method that acts to read all of the data that’s being passed from the client side’s 20,000 repetitions. Once it has read all of the data it writes to the socket with all of the count to confirm that all of the data is sent. Then this method calculates the data receiving time.

Within the main function it checks to see if there are enough arguments passed. If there are not an error message is thrown. If there are enough, a new socket is created and zero initialized and then the server goes into waiting until it hears from the client. Once it hears from the client it creates a pthread with the new data and merges the data back together with thread\_join().



Once this is all done the client hears back from the server and ends while the server continues listening for the client to reach back out to it.