**What is ThreadPool?**

Thread Pool is collection of Live, Reusable threads.

**Why ThreadPool?**

There are many situations where we can use ThreadPool. Consider a Client-Server application in which server has to respond to multiple client at same time. This means multitasking. Server will need a set of certain no. of threads which will do reply to client.

The big advantage of thread pool is that; it provides reusable thread. Though thread creation is very bulky process. It increases overhead. Suppose a new client send some data to server and Server has to respond. What will server do? It will create a new thread which will send data to client and then get killed or exited. Instead of creating thread on each client request we can keep collection of live thread. We will use any free thread which will send data to client. By that we will save a great overhead generated by creating threads multiple times.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CThreadPoolMgr Initialize

Description : Creates threads. Thread limit is 5

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CThreadPoolMgr::Initialize(int nThread) {

m\_nThreadCount = nThread;

int nCounter = 0;

int nThreadCount = m\_nThreadCount - 1;

while( nCounter <= nThreadCount ) {

// Create objects in heap

m\_ptrCThread[nCounter] = new CThread();

m\_ptrCThread[nCounter]->CreateWorkerThread();

m\_hThreadPool[nCounter] = m\_ptrCThread[nCounter]->GetThreadHandle();

// Increment the counter

nCounter++;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CThreadPoolMgr ShutDown

Description : Mark shutdown siganl and wait for each thread to end

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void CThreadPoolMgr::ShutDown() {

int Count = 0;

while(Count <= (m\_nThreadCount - 1)) {

m\_ptrCThread[Count]->SignalShutDownEvent();

Count++;

}

// Check if all threads ended successfully

DWORD dwWaitResult = WaitForMultipleObjects( GetThreadCount(), m\_hThreadPool, TRUE, INFINITE);

switch(dwWaitResult) {

case WAIT\_OBJECT\_0:

{

cout << "All threads are ended.\n";

// Close all handles

Count = 0;

while( Count <= (m\_nThreadCount - 1)) {

m\_ptrCThread[Count]->ReleaseHandles();

delete m\_ptrCThread[Count];

Count++;

}

break;

}

default:

cout << "Wait Error: " << GetLastError() << "\n";

}

}

Also check:

<https://stackoverflow.com/questions/15752659/thread-pooling-in-c11>

# END