

```

// Vahe Karamian - Windows 32 Multithreaded Programming
// Filename: multithreadedHelloWorldc.cpp

// Multithreaded Hello World Program C Runtime Library Compatible

#include <windows.h>
#include <process.h> // For _beginthreadex( )
#include <stdio.h>

// Prototype for Hello application worker thread procedure
DWORD WINAPI WorkerThreadProc( LPVOID lpThreadParameter );

// Main program entry point
int main( void )
{
    HANDLE hWorkerThread;
    DWORD dwWorkerSomeData = 3; // The answer to everything
    DWORD dwWorkerThreadId;
    DWORD dwPrimaryThreadId;

    // Determine and save the primary thread's identifier
    dwPrimaryThreadId = GetCurrentThreadId( );

    printf( "[%08x] Multithreaded hello application started.\n", dwPrimaryThreadId );

    // Launch the worker thread, using C Runtime Library compatible
    hWorkerThread = (HANDLE)
        _beginthreadex( NULL, // Default security attributes
            0, // Default stack size
            WorkerThreadProc, // Thread procedure to start at
            (void *) dwWorkerSomeData, // Argument: work for 3 seconds
            &dwWorkerThreadId // Out: thread ID of worker
        );

    if( hWorkerThread != NULL )
    {
        printf( "[%08x] Worker thread ID = 0x%08x.\n", dwPrimaryThreadId, dwWorkerThreadId );

        // Give the worker thread time to do its thing
        printf( "I am going to sleep for 6 seconds - starting Worker Thread!\n" );
        Sleep( 6000 );
        printf( "I am back - terminating Worker Thread!\n" );

        // Read the worker thread's exit code and cleanup
        DWORD dwThreadExitCode;

        GetExitCodeThread( hWorkerThread, &dwThreadExitCode );

        printf( "[%08x] Worker thread exit code = 0x%08x.\n", dwPrimaryThreadId, dwThreadExitCode );

        // This function takes a single parameter, which is the exit code for the current
        // thread, this function does not return
        _endthreadex( dwThreadExitCode );
    }
    else
    {
        printf( "[%08x] Failed to create worker thread: error 0x%x\n", dwPrimaryThreadId, GetLastError( ) );
    }

    printf( "[%08x] Multithreaded hello application exiting.\n", dwPrimaryThreadId );

    return( 0 );
}

// Thread entry procedure for the worker thread
DWORD WINAPI WorkerThreadProc( LPVOID lpThreadParameter )
{
    // For this thread, lpThreadParameter is used to indicate the number
    // of seconds this thread should execute for.
    DWORD dwNumSeconds = (DWORD) lpThreadParameter;
    DWORD dwThreadId = GetCurrentThreadId( );

    while( dwNumSeconds-- > 0 )
    {
        Sleep( 1000 );

        printf( "[%08x] Hello, multithreaded world!\n", dwThreadId );
    }
}

```

```
// Return from thread procedure and specify an exit code equal to our thread ID.  
return( dwThreadId );  
}
```