```
/**********************************************************************
**
**  NTThrds.c
**   George Shepherd 1/24/93
**
**  Compiled using Microsoft C for Windows NT
**
**  A program to demonstrate threads in Windows NT. In addition to
**   the main thread, this process spawns two other threads-
**      A keyboard input thread
**      A console output thread
**
**  The keyboard input thread reads keystrokes into a shared buffer
**   until either the return key is pressed or the buffer is full.
**   At that point, the input thread signals an event to indicate
**   that input is done. The output thread, which has been waiting
**   on the event, sees that it is OK to print the string to
**   the console. If a blank line is entered, then the input
**   thread raises the "end input" event, which notifies the main
**   thread via an event signal that the process should end.
*/

#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>

#define MAXL_STR 80

/* Shared buffer... */
char str[MAXL_STR];

/* Event Handles... */
HANDLE hInputDone, hOutputDone, hEndInputEvent;

/* The Input Thread... */
DWORD inputThread ( LPVOID lpThreadParameter ) {

  char ch;
  int  nCount = 0;

  while( 1 ) {
     ch = _getch();

     /* The [Enter] key was hit OR buffer is full... */
     if( ch == 13 || nCount >= MAXL_STR - 1 ) {
      if( str[0] == 0 ) {  /* A blank line means end process      */

         /* Set the End Input Event so that the main thread,      */
         /*  which has been waiting on this event, knows to end   */
         /*  the process.                                         */
         SetEvent( hEndInputEvent );
         ExitThread( 0 );
      }

      /* Signal the input event so the output thread will know    */
      /*  it's time to print the string.                          */
      SetEvent( hInputDone );

      /* Wait till the output event is signaled so the input      */
      /*  thread can start taking characters again.               */
      WaitForSingleObject( hOutputDone, INFINITE );

      /* Output is finished. Clear the string and begin reading   */
      /*  from the keyboard again...                              */
      memset( str, '\0', sizeof(str) );
      nCount = 0;
    } /* if */
    else
      /* Add the character to the string... */
      str[ nCount++ ] = ch;

  } /* while */
  return 0;
} /* inputThread */

/* The output thread... */
DWORD outputThread( LPVOID lpThreadParameter ){
  while( 1 ) {
    puts( "Output thread waiting for signal from input thread" );

    /* Wait for the input done event. It signals that the string   */
    /*  can be shown...                                            */
    WaitForSingleObject( hInputDone, INFINITE );

    /* Input is done. Print the string...                          */
    puts( str );
    puts( " " );

    /* Set the output event so that the input thread knows it      */
    /*  may start reading a new string...                           */
    SetEvent( hOutputDone );
  } /* while */
  return 0;
}

/* The main thread... */
int main() {
  HANDLE hInputThread = NULL,
         hOutputThread = NULL;
  DWORD  dwInputThreadID, dwOutputThreadID;

  /* Initialize the buffer... */
  memset( str, '\0', sizeof(str) );

  /* Create the event flags... */
  hInputDone = CreateEvent( NULL,   // Not worried about security here
                            FALSE,  // Let Windows automatically
                                    //  reset the event once the
                                    //  waiting thread resumes...
                            FALSE,  // Initial state is off, or
                                    //  "not signaled"...
                            NULL ); // There needn't be a name for
                                    //  this event...

  hOutputDone = CreateEvent( NULL,
                             FALSE,
                             FALSE,
                             NULL );

  hEndInputEvent = CreateEvent( NULL,
                                FALSE,
                                FALSE,
                                NULL );

  /* Instructions for the user... */
  puts("Enter keystrokes- they will be displayed when you hit [Enter]" );
  puts(" A blank line ends the process\n" );

  /* Start the input thread...   */
  hInputThread = CreateThread( NULL,             // Not worried about
                                                 //  security here
                               0,                // Use the default
                                                 //  stack size
                               inputThread,      // Start of code
                               NULL,             // No parameters needed
                               0,                // Start thread
                                                 //  immediately
                               &dwInputThreadID ); // Put the thread ID
                                                 //  here

  /* Start the output thread...  */
  hOutputThread = CreateThread( NULL,
                          0,
                          outputThread,
                          NULL,
                          0,
                          &dwOutputThreadID );

  /* Wait on the "end input" event. It will be raised by the input */
  /*  thread when an empty line is entered...                      */
  WaitForSingleObject( hEndInputEvent, INFINITE );

  puts( "End of input..." );

  /* Clean up... */
  TerminateThread( hOutputThread, 0 );
  CloseHandle( hInputThread );
  CloseHandle( hOutputThread );
  CloseHandle( hInputDone );
  CloseHandle( hOutputDone );
  CloseHandle( hEndInputEvent );

  return 0;
}
```