```
/************************************************************************
**
**  OS2Thrds.c
**   George Shepherd 1/24/93
**
**  Compiled using JPI Topspeed C for OS/2. Be sure to link with the
**   multithread library.
**
**  A program to demonstrate threads in OS/2. 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 raises a semaphore to indicate
**   that input is done. The output thread, which has been waiting
**   on the semaphore, 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" semaphore, which notifies the main
**   thread via the semaphore that the process should end.
**
*/

#define INCL_DOSPROCESS
#define INCL_DOSSEMAPHORES
#include <os2kernl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>

#define MAXL_STR 80

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

/* Semaphore Handles... */
ULONG hInputDone = 0,
     hOutputDone = 0,
     hEndInputSem = 0;

/* Stacks for the input and the output threads... */
BYTE inputThreadStack[ 2048 ];
BYTE outputThreadStack[ 2048 ];

/* The Input Thread... */
void inputThread ( void ) {
  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 the process     */

           /* Clear the End Input Semaphore so that the main thread,    */
           /*  which has been waiting on this semaphore to clear, knows */
           /*  to end the process.*/
           DosSemClear( &hEndInputSem );
           DosExit( 0,    /* End current thread... */
                    0 );  /* Return code...        */
        }

        /* Clear the input semaphore so the output thread will know */
        /*  it's time to print the string.                           */
        DosSemClear( &hInputDone );

        /* Wait till the output semaphore is cleared so the input   */
        /*  thread can start taking characters again.               */
        DosSemRequest( &hOutputDone, SEM_INDEFINITE_WAIT );

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

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

/* The output thread... */
void outputThread( void ){
  while( 1 ) {
     puts( "Output thread waiting for input semaphore to clear..." );

     /* Wait for the input done semaphore to clear. It signals      */
     /*  that the string can be shown...                             */
     DosSemRequest( &hInputDone, SEM_INDEFINITE_WAIT );
```

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

      /* Clear the output semaphore so that the input thread knows it */
      /*  may start reading a new string...                            */
      DosSemClear( &hOutputDone );
   } /* while */
   return;
}

/* The main thread... */
int main() {
   USHORT uInputThreadID, uOutputThreadID;

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

   /* Set the semaphores... */
   DosSemSet( &hInputDone );
   DosSemSet( &hOutputDone );
   DosSemSet( &hEndInputSem );

   /* 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 here. Pass the address of the function, */
   /*  a place to put the thread's ID, and a buffer that the thread  */
   /*  can use as a stack. Since the stack grows downward, pass the  */
   /*  address of the end of the buffer.                              */
   DosCreateThread( inputThread,
                    &uInputThreadID,
                    inputThreadStack + sizeof( inputThreadStack ) );

   /* Start the output thread...  */
   DosCreateThread( outputThread,
                    &uOutputThreadID,
                    outputThreadStack + sizeof( outputThreadStack ) );

   /* Wait on the "end input" semaphore. It will be cleared by the input  */
   /*  thread when a backspace character is typed...                     */
   DosSemWait( &hEndInputSem, SEM_INDEFINITE_WAIT );

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

   return 0;
}
```