```
//==================== file = thread.c =====Pseudo Code=================
//=   - Demonstrates both Windows and POSIX threads                      =
//=======================================================================
//=  Notes:                                                             =
//=    1) This program demonstrates how threads work and shows how a global =
//=       "kill variable" can be set and then checked-for in each thread.   =
//=       When the variable is detected as set, the thread is ended.     =
//=---------------------------------------------------------------------=
//=   Kicking-off threads...                                            =
//=   Executing thread # 0 with ID = 178                                =
//=     i = 0 in thread # 0                                             =
//=   Created thread # 0, ID = 56                                       =
//=   Created thread # 1, ID = 60                                       =
//=   All threads have been kicked-off...                               =
//=   Executing thread # 1 with ID = 180                                =
//=     i = 0 in thread # 1                                             =
//=     i = 1 in thread # 0                                             =
//=     i = 1 in thread # 1                                             =
//=     i = 2 in thread # 0                                             =
//=     i = 2 in thread # 1                                             =
//=     i = 3 in thread # 0                                             =
//=     i = 3 in thread # 1                                             =
//=     i = 4 in thread # 0                                             =
//=     i = 4 in thread # 1                                             =
//=   Killing thread #0 from main()...                                  =
//=     >>> Killing thread # 0                                          =
//=     i = 5 in thread # 1                                             =
//=     i = 6 in thread # 1                                             =
//=     i = 7 in thread # 1                                             =
//=     i = 8 in thread # 1                                             =
//=     i = 9 in thread # 1                                             =
//=   Killing thread #1 from main()...                                  =
//=   Waiting for 5 seconds and then ending main()...                   =
//=     >>> Killing thread # 1                                          =
//=   End!                                                              =
//=---------------------------------------------------------------------=
//=  Build: Windows: bcc32 -WM thread.c, cl /MT thread.c wsock32.lib    =
//=         Unix: gcc thread.c -lpthreads -o thread                     =
//=---------------------------------------------------------------------=
//=  Execute: thread                                                    =
//=---------------------------------------------------------------------=
//=======================================================================
#define  WIN              // WIN for Winsock and BSD for BSD Unix

//----- Include files ---------------------------------------------------
#include <stdio.h>        // Needed for printf()
#include <stdlib.h>       // Needed for exit()
#ifdef WIN
  #include <errno.h>      // Needed for errno
  #include <stddef.h>     // Needed for _threadid
  #include <process.h>    // Needed for _beginthread(), and _endthread()
  #include <windows.h>    // Needed for Sleep()
#endif
#ifdef POSIX
  #include <pthread.h>    // Needed for pthread_create() and pthread_exit()
#endif
```

```
//----- Defines -------------------------------------------------------------
#define NUM_THREAD  2   // Number of threads (numbered 0 ... NUM_THREAD - 1)

//----- Globals -------------------------------------------------------------
int Kill[NUM_THREAD] = {0};   // Kill flags for threads by thread number

//----- Function prototypes -------------------------------------------------
#ifdef WIN
  void thread_code(void *threadno);   // Windows thread function
#endif
#ifdef BSD
  void* thread_code(void *threadno);  // POSIX thread function
#endif
void mSleep(unsigned int theSeconds); // CrossPlatform Sleep function

//===========================================================================
//=  Main program                                                           =
//===========================================================================
int main(void)
{
#ifdef WIN
  unsigned int thread_id;          // Thread id assigned by Windows
#endif
#ifdef BSD
  int errno;                       // Error number
#endif
  int i;                           // Loop counter

  // Initialize Kill[] to all zeros (boolean false)
  for (i=0; i<NUM_THREAD; i++)
    Kill[i] = 0;

  // Kick-off threads
  printf("Kicking-off threads... \n");
  for (i=0; i<NUM_THREAD; i++)
  {
#ifdef WIN
    if ((thread_id = _beginthread(thread_code,4096,(void *)i)) < 0)
    {
      printf("Unable to create thread # %d, id = %u \n", i, thread_id);
      exit(1);
    }
#endif
#ifdef BSD
    if ((errno = pthread_create(NULL, NULL, thread_code,(void *)i)) != 0)
    {
      printf("Unable to create thread %d, errno = %d \n", i, errno);
      exit(1);
    }
#endif
    printf("Created thread # %d \n", i);
  }
  printf("All threads have been kicked-off... \n");

  // Wait for 5 seconds and then kill thread #0
  mSleep(5);
  printf("Killing thread #0 from main()... \n");
```

```
  Kill[0] = 1;

  // Wait for 5 more seconds and then kill thread #1
  mSleep(5);
  printf("Killing thread #1 from main()... \n");
  Kill[1] = 1;

  // Wait for 5 more seconds and then output "End" and end
  printf("Waiting for 5 seconds and then ending main()... \n");
  mSleep(5);
  printf("End! \n");

  return(0);
}

//============================================================================
//=  This is is the thread_code function                                     =
//=   - Arguments always declared as void * and must be recast               =
//============================================================================
#ifdef WIN
  void thread_code(void *threadno)
#endif
#ifdef BSD
  void *thread_code(void *threadno)
#endif
{
  int    thread_num;    // Thread number as assigned in main()
  int    i;             // Loop counter

  // Set thread_num to threah number as assigned in main()
  thread_num = (int) threadno;

  // Loop 100 times outputing thread status.
  printf("Executing thread # %d with ID = %u \n", thread_num);
  for (i=0; i<100; i++)
  {
    printf("  i = %d in thread # %d \n", i, thread_num);

    // Wait for 1 second
    mSleep(1);

    // Check if Kill flag is set for this thread number and then _endthread()
    if (Kill[thread_num] > 0)
    {
      printf("  >>> Killing thread # %d \n", thread_num);
#ifdef WIN
         _endthread();
#endif
#ifdef BSD
      pthread_exit(NULL);
#endif
    }
  }

  // End threads
#ifdef WIN
         _endthread();
```

```
#endif
#ifdef BSD
      pthread_exit(NULL);
#endif
}

//============================================================================
//=  Function to sleep for numseconds                                        =
//=   - Windows and Unix have different native sleep function                =
//============================================================================
void mSleep(unsigned int numseconds)
{
#ifdef WIN
       Sleep(numseconds * 1000);
#endif
#ifdef BSD
       sleep(numseconds);
#endif
}
```