Unit 3: Processes and Threads

3.6. Win32 Thread Creation

Win32 Thread Creation

```
HANDLE CreateThread (
    LPSECURITY_ATTRIBUTES lpsa,
    DWORD cbStack,
    LPTHREAD_START_ROUTINE lpStartAddr,
    LPVOID lpvThreadParm,
    DWORD fdwCreate,
    LPDWORD lpIDThread)
```

cbStack == 0: thread‘s stack size defaults to primary thread‘s size

- lpstartAddr points to function declared as

  `DWORD WINAPI ThreadFunc(LPVOID)`
- lpvThreadParm is 32-bit argument
- LPIDThread points to DWORD that receives thread ID
  non-NULL pointer !

Win32 Thread Termination

```
VOID ExitThread( DWORD devExitCode )
```

- When the last thread in a process terminates, the process itself terminates
  (TerminateThread() does not execute final SEH)
- Thread continues to exist until last handle is closed (CloseHandle())

```
BOOL GetExitCodeThread (
      HANDLE hThread, LPDWORD lpdwExitCode)
```

- Returns exit code or STILL_ACTIVE

Suspending and Resuming Threads

- Each thread has suspend count
- Can only execute if suspend count == 0
- Thread can be created in suspended state

```
DWORD ResumeThread (HANDLE hThread)
DWORD SuspendThread(HANDLE hThread)
```

- Both functions return suspend count or 0xFFFFFFFF on failure

Synchronization & Remote Threads

- WaitForSingleObject() and WaitForMultipleObjects() with thread handles as arguments perform thread synchronization
  - Waits for thread to become signaled
  - ExitThread(), TerminateThread(), ExitProcess() set thread objects to signaled state
- CreateRemoteThread() allows creation of thread in another process
  - Not implemented in Windows 9x
- C library is not thread-safe; use libcmt.lib instead
  - #define _MT before any include
  - Use _beginthreadex/_endthreadex instead of Create/ExitThread

Example: multithreaded sort

```
typedef struct _THREADARG {
        DWORD iTh;
        LPRECORD LowRec;
        LPRECORD HighRec;
} THREADARG, *PTHREADARG;
```

```
/* Create the sorting threads. */
LowRecNo = 0;
for (iTh = 0; iTh < NPr; iTh++) {
        ThArg [iTh].iTh = iTh;
        ThreadHandle [iTh] = (HANDLE)_beginthreadex (
                                NULL, 0, ThSort, &ThArg [iTh],
                                CREATE_SUSPENDED, &ThId);
}
/* Resume all the initially suspened threads. */
for (iTh = 0; iTh < NPr; iTh++)
        ResumeThread (ThreadHandle [iTh]);
/* Wait for the sort-merge threads to complete. */
WaitForSingleObject (ThreadHandle [0], INFINITE);
for (iTh = 0; iTh < NPr; iTh++)
        CloseHandle (ThreadHandle [iTh]);
```

Example: the sort function

```
DWORD WINAPI ThSort (PTHREADARG pThArg) {
    DWORD GrpSize = 2, RecsInGrp, MyNumber, TwoToI = 1;
            /* TwoToI = 2**i, where i is the merge step number. */
    LPRECORD First;
    /* do all the work */
    /* Either exit the thread or wait for the adjoining thread. */
    while ((MyNumber % GrpSize) == 0 && RecsInGrp < nRec) {
                                    /* Merge with the adjacent sorted array. */
            WaitForSingleObject (ThreadHandle [MyNumber + TwoToI], INFINITE);
            MergeArrays (First, First + RecsInGrp);
            RecsInGrp *= 2; GrpSize *= 2; TwoToI *=2;
    }
    _endthreadex (0);
    return 0;           /* Suppress a warning message. */
}
```