Laborator 9

Thread-uri - Windows

Sisteme de Operare

21 - 27 aprilie 2016

Windows Threads API

- Operații cu thread-uri
  - CreateThread
  - ThreadProc
  - WaitForSingleObject
  - ExitThread
  - GetCurrentThread
- Thread Local Storage
  - TlsAlloc
  - TlsFree
  - TlsGetValue
  - TlsSetValue

Mecanisme de sincronizare

- Mutex (POSIX, Win32)
- Semafor (POSIX, Win32)
- Secțiune critică (Win32)
- Variabilă de condiție (POSIX)
- Barieră (POSIX)
- Eveniment (Win32)
- Operații atomice cu variabile partajate (Win32)
- Thread pooling (Win32)

Mutex (Win32)

- Creare mutex
  - `HANDLE CreateMutex(LPSECURITY_ATTRIBUTES lpMutexAttributes, BOOL bInitialOwner, LPCTSTR lpName)`
- Deschidere mutex deja creat
  - `HANDLE OpenMutex(DWORD dwDesiredAccess, BOOL bInheritHandle, LPCTSTR lpName)`
- Așteptare/acaparare mutex
  - Funcțiile din familia `WaitForSingleObject`
- Eliberare mutex
  - `BOOL ReleaseMutex(HANDLE hMutex)`

Semafor (Win32)

- Creare semafor
  - `HANDLE CreateSemaphore(LPSECURITY_ATTRIBUTES sematttr, LONG initial_count, LONG maximum_count, LPCTSTR name)`
- Deschidere semafor deja existent
  - `HANDLE OpenSemaphore(DWORD dwDesiredAccess, BOOL bInheritHandle, LPCTSTR name)`
- Așteptare/decrementare semafor
  - Funcțiile din familia `WaitForSingleObject`
- Incrementare, cu `lReleaseCount`
  - `BOOL ReleaseSemaphore(HANDLE hSemaphore, LONG lReleaseCount, LPLONG lpPreviousCount)`

Secțiune critică

- `CRITICAL_SECTION`
- Sincronizare DOAR între firele de execuție ale **aceluiași proces**
- Inițializare/Distrugere secțiune critică
  - `void InitializeCriticalSection(LPCRITICAL_SECTION pcrit_sect)`
  - `void DeleteCriticalSection(LPCRITICAL_SECTION pcrit_sect)`
- Intrare în secțiune critică
  - `void EnterCriticalSection(LPCRITICAL_SECTION lpCriticalSection)`
  - `BOOL TryEnterCriticalSection(LPCRITICAL_SECTION lpCriticalSection)`
- Ieșire din secțiune critică
  - `void LeaveCriticalSection(LPCRITICAL_SECTION lpCriticalSection)`

Eveniment

- Două tipuri: manual-reset, auto-reset
- Creare eveniment
  - `HANDLE WINAPI CreateEvent(LPSECURITY_ATTRIBUTES lpEventAttributes, BOOL bManualReset, BOOL bInitialState, LPCTSTR lpName);`
- Semnalizare eveniment
  - `BOOL WINAPI SetEvent(HANDLE hEvent);`
  - `BOOL WINAPI PulseEvent(HANDLE hEvent);`
  - `BOOL WINAPI ResetEvent(HANDLE hEvent);`
- Așteptarea unui eveniment
  - Funcțiile din familia `WaitForSingleObject`

Operații atomice cu variabile partajate

- Incrementare/Decrementare variabilă
  - `LONG InterlockedIncrement(LONG volatile *lpAddend)`
  - `LONG InterlockedDecrement(LONG volatile *lpDecend)`
- Atribuire atomică
  - `LONG InterlockedExchange(LONG volatile *Target, LONG Value)`
  - `LONG InterlockedExchangeAdd(LPLONG volatile Addend, LONG Value)`
  - `PVOID InterlockedExchangePointer(PVOID volatile *Target, PVOID Value)`
- Atribuire atomică condiționată
  - `LONG InterlockedCompareExchange(LONG volatile *dest, LONG exchange, LONG comp)`
  - `PVOID InterlockedCompareExchangePointer(PVOID volatile *dest, PVOID exchange, PVOID comp)`

Thread pooling

- Fiecare task primește un thread din pool
- Eliminare overhead creare/terminare fire de execuție
- Task-urile pot fi:
  - Executate **imediat**
  - Executate **mai târziu** (operații de așteptare + funcție callback asociată)
    - Așteptarea terminării unei operații I/O asincrone
    - Așteptarea expirării unui TimerQueue
    - Funcții de așteptare înregistrate