Laboratorul 11

Mecanismul de sincronizare de tip eveniment

Temă laborator

Realizați 2 programe similare ca mod de funcționare cu cele realizate în laboratorul anterior (laboratorul 9 – sincronizarea a 2 fire de execuție prin intermediul unui mecanism de tip mutex) care:

1. Să se sincronizeze de această dată cu ajutorul mecanisme de tip eveniment, derularea elementelor de tip *Progress Bar* va fi condiționată de un set de evenimente.
2. La crearea firelor de execuție afișați printr-un mesaj crearea cu succes a lor sau eșecul creării.
3. Vor exista 2 evenimente diferite, de tip auto-reset, fiecare asociate cu unul din cele 2 fire de execuție anterior create.
4. Fiecare element de tip *Progress Bar* va fi controlat din cadrul firului de execuție, în mod separat și independent de restul programului.
5. Una dintre aplicații va avea un buton prin intermediul căruia se va porni mecanismul de derulare al elementelor de tip *Progress Bar*.
6. La sfârșitul programului, atunci când distrugeți fereastra eliberați resursele utilizate și alocate de dvs.

Evenimente

Evenimentele[1] reprezintă unul din mecanismele oferite de nucleul sistemului de operare Windows Embedded Compact ce oferă posibilitatea unui fir de execuție să semnalizeze unul sau mai multe fire de execuție, diferite, că un anumit eveniment a avut loc în sistem.

În mod similar mecanismului de sincronizare de tip mutex un eveniment poate sau nu să aibă un nume asociat. În momentul în care evenimentele au nume asociate acestea pot realiza sincronizarea între fire de execuție aparținând unor procese diferite; în caz contrar se pot sincroniza numai fire de execuție aparținând aceluiași proces.

[1] După cum se va prezenta ulterior evenimentele sunt utilizate într-un număr foarte mare de situații. De exemplu, evenimentele sunt utilizate de nucleul sistemului de operare pentru a informa un driver că o întrerupere a avut loc și că întreruperea trebuie tratată.

Un eveniment este activ sau semnalizat atunci când este setat (prin intermediul funcției *SetEvent*) și nu este semnalizat atunci când este resetat (prin intermediul funcției *ResetEvent*). În momentul în care un eveniment este creat, prin intermediul funcției *CreateEvent*, acesta poate fi setat sau nu, poate sau nu să aibă un nume asociat și poate fi un eveniment de tip cu reset automat sau cu reset manual.

Evenimentele cu reset manual trebuie trecute în starea inactivă, manual prin intermediul funcției *ResetEvent* imediat după ce sincronizarea este realizată prin intermediul funcțiilor *WaitForSingleObject* sau *WaitForMultipleObjects* pentru a nu permite, de exemplu, altor fire de execuție să acceseze eventualele resurse partajate. Atâta timp cât un eveniment este semnalizat toate firele de execuție ce așteaptă acest eveniment sunt **deblocate „simultan"** împreună cu toate firele de execuție care încep să aștepte acest eveniment în perioada de timp între momentul în care evenimentul este setat și momentul în care este resetat. Utilizând funcția *PulseEvent* în locul funcției *SetEvent* evenimentul va fi setat va debloca toate firele de execuție ce așteaptă să fie deblocate de acest eveniment, iar ulterior evenimentul este resetat. Dacă nici un fir de execuție nu așteaptă să fie deblocat evenimentul este resetat imediat chiar dacă nu a deblocat nici un fir de execuție.

În cazul evenimentelor cu auto-reset acestea sunt trecute în stare nesemnalizată de către nucleul SO imediat ce evenimentul determină deblocarea unei funcții de tipul *WaitForSingleObject* sau *WaitForMultipleObjects*. Daca un eveniment cu auto-reset este setat și una sau mai multe funcții asteaptă pentru a fi deblocate atunci primul fir de execuție ce așteaptă evenimentul este deblocat, iar imediat evenimentul este resetat; celelalte funcții ce asteaptă să fie deblocate vor rămâne în continuare blocate. Deci, în aceast caz (al evenimentelor cu auto-reset) numai un singur fir de execuție se va debloca și va fi executat. În cazul utilizării funcției *PulseEvent* cu un eveniment de tip auto-reset modul de comportare este identic cu cel prezentat anterior (ca în cazul utilizării funcției *SetEvent*), diferența fundamentală constă doar în situația în care nici un fir de execuție nu așteaptă să fie deblocat – în acest caz evenimentul nu va rămâne în starea setată până apare un astfel de fir de execuție ci va fi imediat resetat chiar dacă nu a deblocat nici un *thread*.

Concluzionând, putem spune că funcția *PulseEvent* setează evenimentul, iar dacă nu există nici un fir de execuție în așteptare sau nici un fir de execuție nu poate fi deblocat imediat atunci *PulseEvent* resetează evenimentul trecându-l într-o stare inactivă și nu așteaptă apariția unui fir de execuție ce poate fi deblocat de acel eveniment.

Temă:

1. Completați programul dezvoltat anterior astfel încât fiecare element de tip *Progress Bar* să fie dublat de un element de tip *Edit Box* în care să se deruleze numerele de la 1 la 10 sincron cu evoluția elementului *Progress Bar*.
2. Introduceți un buton care atunci când este apăsat pentru prima dată oprește procesul de derulare a elementelor *Progress Bar* și *Edit Box* existente pe interfața grafică. Pentru o a doua apăsare permite continuarea execuției din punctul în care a fost lăsată. Aceste funcționări repetându-se ciclic.