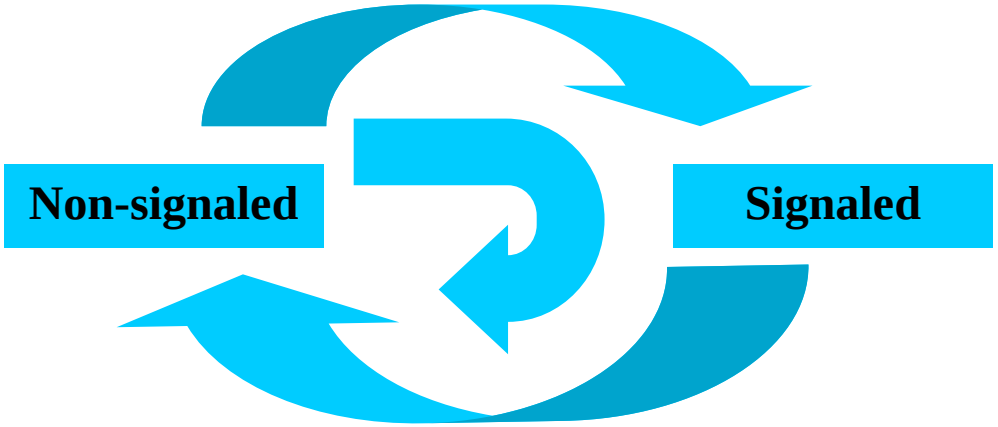
14. Mécanismes de Synchronisation entre Threads et/ou Processus

14.1 Evénements

- 14.1.1 CreateEvent
- 14.1.2 OpenEvent
- 14.1.3 SetEvent, ResetEvent
- 14.1.4 PulseEvent
- 14.1.5 Exemple

14.1 Evénements

- Plusieurs threads peuvent être déclenchés simultanément sur l'attente d'un même événement (persistant)
- Les événements sont classés en deux types :
  - Evénement « manual-reset » donc persistant
  - Événement « auto-reset » donc fugace
- Les Fonctions associées sont CreateEvent, SetEvent, ResetEvent, PulseEvent.



- Les Fonctions d'attente sont alors WaitForSingleObject ou WaitForMultipleObjects

14.1.1. CreateEvent

- La fonction, **CreateEvent,** crée un Objet Evénement nommé ou anonyme.
- Syntaxe :

> - *HANDLE CreateEvent (LPSECURITY_ATTRIBUTES lpEventAttributes, BOOL bManualReset, BOOL bInitialState, LPCTSTR lpName);*

- Arguments :

> - *LPSECURITY_ATTRIBUTES lpEventAttributes, pointeur sur les attributs de sécurité*
> - *BOOL bManualReset, si TRUE il s'agit d'un signal persistant (manual-reset), sinon fugace (auto-reset). Dans le premier ca il faut utiliser la fonction ResetEvent pour ramener l'objet Event à un état nonsignaled.*
> - *BOOL bInitialState, si TRUE l'état initial de l'Evénement est signaled, non-signaled sinon*
> - *LPCTSTR lpName, nom de l'objet événement. Il est limité à MAX_PATH caractères et peut contenir n'importe quel caractère sauf backslash. Si NULL, alors l'objet est anonyme.*

- Pb : si lpName correspond au nom d'un objet d'un autre type (sémaphore, mutex…), la fonction retournera une erreur car ces objets partagent le même espace de noms (ERROR_INVALID_HANDLE).

14.1.2 OpenEvent

- La fonction, **OpenEvent,** retourne un handle sur un Objet Evénement nommé, NULL sinon.
- Syntaxe :

  - *HANDLE OpenEvent(DWORD dwDesiredAccess, BOOL bInheritHandle, LPCTSTR lpName) ;*

- Arguments :

  - *DWORD dwDesiredAccess, flag d'accès*
  - *BOOL bInheritHandle, flag d'héritage (TRUE, les processus créés par le processus courant hériteront du Handle)*
  - *LPCTSTR lpName, pointe sur le nom de l'objet événement*

14.1.3 SetEvent, ResetEvent

- La fonction, **SetEvent**, met l'objet événement spécifié dans l'état *signaled*
- Syntaxe

  - *BOOL SetEvent(HANDLE hEvent) ;*

- Arguments

  - *HANDLE hEvent, handle sur l'événement spécifié*

- La fonction, **ResetEvent**, met l'objet événement spécifié dans l'état *non-signaled*
- Syntaxe

  - *BOOL ResetEvent(HANDLE hEvent) ;*

- Arguments

  - *HANDLE hEvent, handle sur l'événement spécifié*

14.1.4 PulseEvent

- La fonction, **PulseEvent**, permet en une seule opération de positionner un objet événement spécifié dans l'état *signaled* puis dans l'état *non-signaled* après avoir déclenché un certain nombre de threads en attente :
- Dans le cas d'un Evénement persistant, tous les threads qui peuvent être déclenchés simultanément le sont. L'événement est ensuite passé à l'état *non-signaled*
- Dans le cas d'un Evénement fugace, l'événement est d'abord passé à l'état *non-signaled et* un seul thread en attente est déclenché.
- Si aucun thread n'est en attente, l'événement est ramené à l'état *non-signaled*
- Syntaxe

  - *BOOL PulseEvent (HANDLE hEvent) ;*

- Arguments

  - *HANDLE hEvent, handle sur l'événement spécifié*

14.1.5 Exemple :

```
HANDLE hEvent;
hEvent = OpenEvent(EVENT_ALL_ACCESS, FALSE,"WaitEvent");
SetEvent(hEvent);



HANDLE hEvent;
hEvent = CreateEvent(NULL, FALSE, FALSE, "WaitEvent");
WaitForSingleObject(hEvent, INFINITE);
CloseHandle(hEvent);
```