15. Mécanismes de communication entre processus et/ou threads

15.1 Tubes anonymes

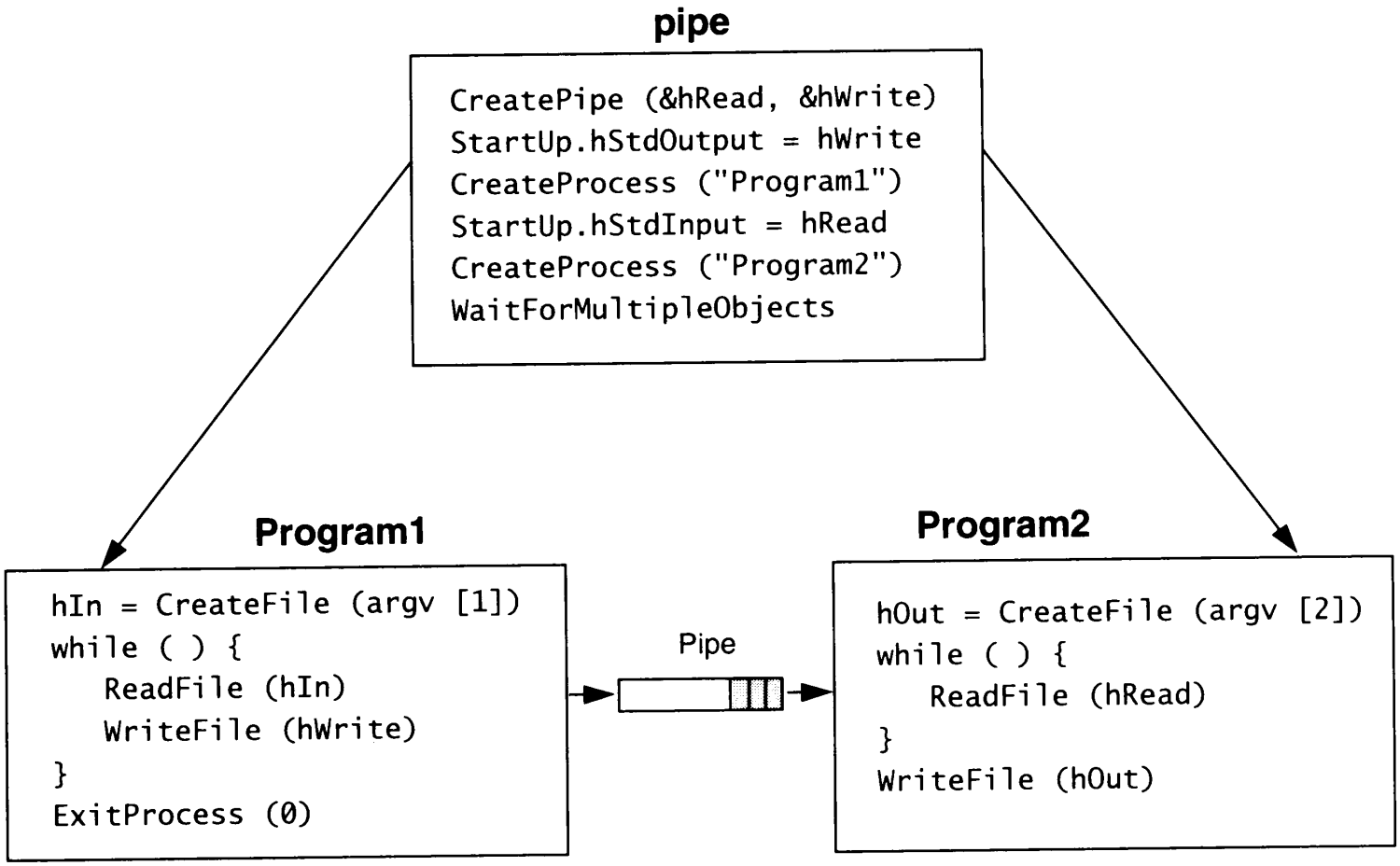
- La fonction, **CreatePipe**, crée un tube anonyme et retourne deux handles pour la lecture et l'écriture dans le tube
- Syntaxe :

> - *BOOL CreatePipe(PHANDLE hReadPipe, PHANDLE hWritePipe, LPSECURITY_ATTRIBUTES lpPipeAttributes, DWORD nSize);*

- Arguments :

> - *PHANDLE hReadPipe, Handle de lecture dans le tube*
> - *PHANDLE hWritePipe, Handle d'écriture dans le tube*
> - *LPSECURITY_ATTRIBUTES lpPipeAttributes, pointeur sur les attributs de sécurité*
> - *DWORD nSize, taille du buffer du tube en octets*

15.1.1 Tubes anonymes : Communication entre deux Processus



15.2 Tubes Nommés

- Les tubes nommés sont orientés messages, ainsi le processus lecteur peut lire des messages de taille variable, correspondante à la taille du message envoyé par le processus écrivain
- Les tubes nommés sont bidirectionnels
- Il peut y avoir plusieurs instances d'un tube nommé. Par exemple, plusieurs clients peuvent communiquer avec un serveur en utilisant le même tube et le serveur peut répondre aux clients en utilisant la même instance
- Un tube nommé peut se trouver indifféremment sur une même machine ou sur un réseau

15.2.1 Côté serveur :

- La fonction, **CreateNamedPipe** , crée un tube nommé et retourne un handle sur le tube.
- Syntaxe :

> - *HANDLE CreateNamedPipe(LPCTSTR lpName, DWORD dwOpenMode, DWORD dwPipeMode, DWORD nMaxInstances, DWORD nOutBufferSize, DWORD nInBufferSize, DWORD nDefaultTimeOut, LPSECURITY_ATTRIBUTES lpSecurityAttributes);*

- Arguments :

> - *LPCTSTR lpName, pointeur sur le nom du tube nommé*
> - *DWORD dwOpenMode, flags gérant le mode d'accès et autres (exemple : PIPE_ACCESS_DUPLEX (bidirectionnel) PIPE_ACCESS_INBOUND (unidirectionnel client - serveur) PIPE_ACCESS_OUTBOUND (unidirectionnel serveur – client) ...)*
> - *DWORD dwPipeMode, flags gérant le type, le mode de lecture et d'attente du pipe (exemple : PIPE_TYPE_BYTE ou PIPE_TYPE_MESSAGE, PIPE_READMODE_BYTE ou PIPE_READMODE_MESSAGE, PIPE_WAIT ou PIPE_NOWAIT)*

- *DWORD nMaxInstances, nombre maximal d'instances qui peuvent être crées pour ce tube*
- *DWORD nOutBufferSize, taille du buffer de sortie*
- *DWORD nInBufferSize, taille du buffer en entrée*
- *DWORD nDefaultTimeOut, Timeout en millisecondes*
- *LPSECURITY_ATTRIBUTES lpSecurityAttributes, pointeur sur les attributs de sécurité*

15.2.2 Côté Client :

- Le clients peuvent ouvrir une instance du tube avec CreateFile (déjà vu), puis utiliser ReadFile et WriteFile (selon le type de tube créé) et CloseHandle dans une boucle d'attente active (les fonctions d'attentes Wait.. ne sont pas utilisables pour un tube).
- D'autres fonctions de plus haut niveau sont utilisables pour gérer cette attente active : PeekNamedPipe, TransactNamedPipe, CallNamedPipe, ConnectNamedPipe
- La fonction, **PeekNamedPipe**, lit des données d'un tube nommé ou anonyme vers un buffer sans les effacer du tube. Retourne une valeur non nulle en cas de succès.
- Rem : Le client est connecté en permanence sur une instance du tube
- Syntaxe :

  - *BOOL PeekNamedPipe(HANDLE hNamedPipe, LPVOID lpBuffer, DWORD nBufferSize, LPDWORD lpBytesRead, LPDWORD lpTotalBytesAvail, LPDWORD lpBytesLeftThisMessage);*

- Arguments :

- *HANDLE hNamedPipe, handle du tube*
- *LPVOID lpBuffer, pointeur sur le buffer des données lues*
- *DWORD nBufferSize, taille en octets de ce buffer*
- *LPDWORD lpBytesRead, pointeur sur le nombre d'octets lus*
- *LPDWORD lpTotalBytesAvail, pointeur sur le nombre d'octets disponibles*
- *LPDWORD lpBytesLeftThisMessage pointeur sur les octets non lus de ce message*

♦ La fonction, **TransactNamedPipe**, écrit un message et lit un message dans un tube spécifié. Retourne une valeur nulle en cas d'échec.

♦ Rem : Le client est connecté en permanence sur une instance du tube

♦ Syntaxe :

- *BOOL TransactNamedPipe(HANDLE hNamedPipe, LPVOID lpInBuffer, DWORD nInBufferSize, LPVOID lpOutBuffer, DWORD nOutBufferSize, LPDWORD lpBytesRead, LPOVERLAPPED lpOverlapped);*

♦ Arguments :

- *HANDLE hNamedPipe, handle sur le tube spécifié*

- *LPVOID lpInBuffer, pointeur sur le buffer des données écrites dans le tube*
- *DWORD nInBufferSize, taille du buffer des données écrites*
- *LPVOID lpOutBuffer, pointeur sur le buffer des données lues dans le tube*
- *DWORD nOutBufferSize, taille du buffer des données lues*
- *LPDWORD lpBytesRead, taille en octets des données lues dans le tube*
- *LPOVERLAPPED lpOverlapped, adresse de la structure d'overlapped (NULL par défaut)*

♦ La fonction, **CallNamedPipe**, se connecte sur un tube de type message (attend si aucune instance du tube n'est disponible), lit et écrit dans l'instance du tube et la ferme.

♦ Rem : Le client n'est pas connecté en permanence sur une instance du tube

♦ Syntaxe :

- *BOOL CallNamedPipe(LPCTSTR lpNamedPipeName, LPVOID lpInBuffer, DWORD nInBufferSize, LPVOID lpOutBuffer, DWORD nOutBufferSize, LPDWORD lpBytesRead, DWORD nTimeOut);*

♦ Arguments :

- *LPCTSTR lpNamedPipeName, pointeur sur le nom du tube nommé*
- *LPVOID lpInBuffer, pointeur sur le buffer*
- *LPVOID lpInBuffer, pointeur sur le buffer des données écrites dans le tube*
- *DWORD nInBufferSize, taille du buffer des données écrites*
- *LPVOID lpOutBuffer, pointeur sur le buffer des données lues dans le tube*
- *DWORD nOutBufferSize, taille du buffer des données lues*
- *LPDWORD lpBytesRead, taille en octets des données lues dans le tube*
- *DWORD nTimeOut, timeout en millisecondes*

♦ La fonction, **ConnectNamedPipe**, permet à un processus serveur d'attendre la connexion d'un processus client sur une instance du tube nommé. Retourne une valeur non nulle en cas de succès.

♦ Rem : Le client peut se connecter par CreateFile (permanente) ou CallNamedPipe (temporaire)

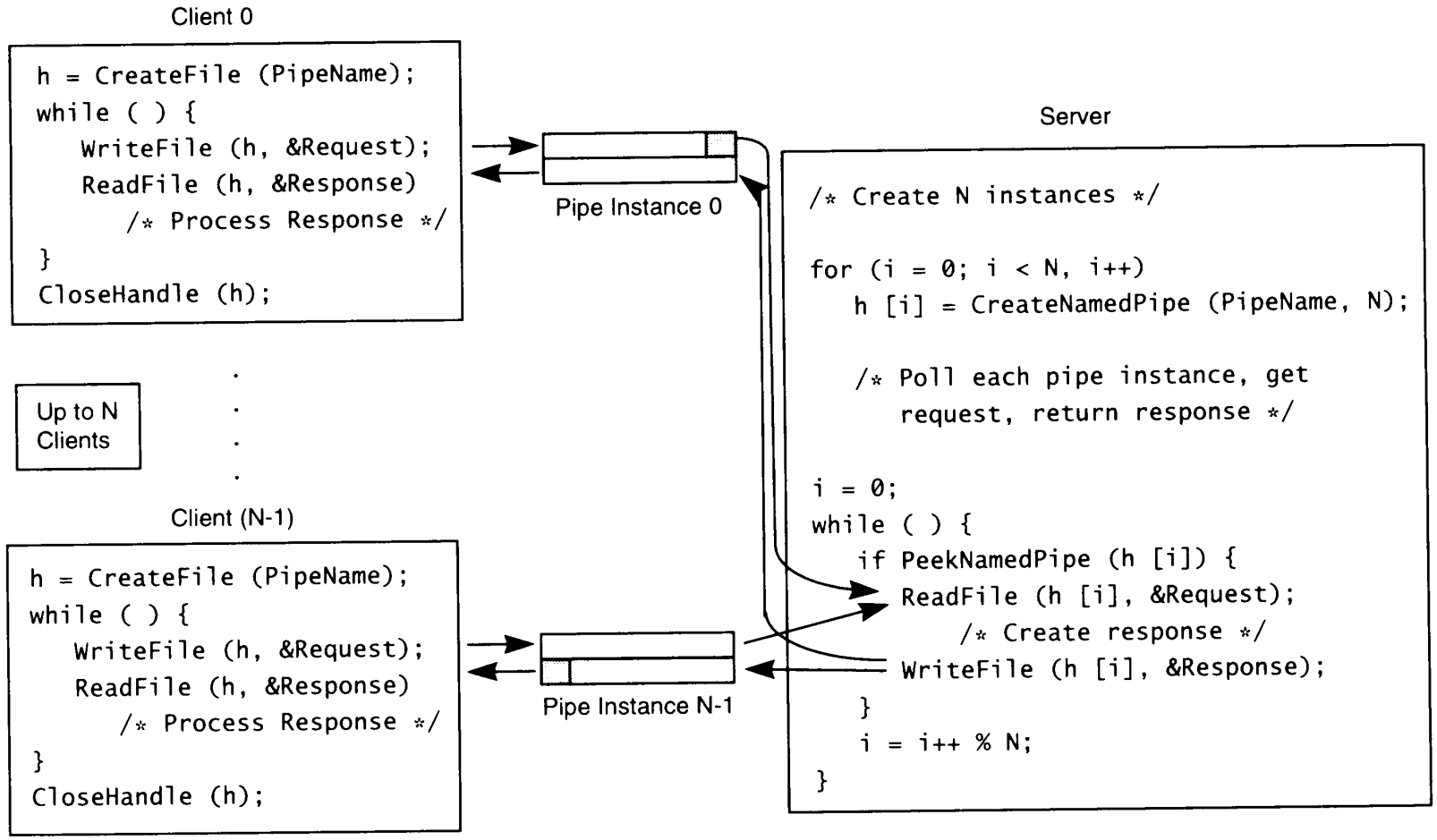
♦ Syntaxe :

- *BOOL ConnectNamedPipe(HANDLE hNamedPipe, LPOVERLAPPED lpOverlapped);*

♦ Arguments :

- *HANDLE hNamedPipe, Handle*
- *LPOVERLAPPED lpOverlapped, adresse de la structure d'overlapped (NULL par défaut)*

15.2.3 Tubes Nommés : exemple de base



15.3 Mailslots

- Le Mailslot est unidirectionnel
- Les Mailslots sont des mécanismes de broadcast
- Un Mailslot peut avoir plusieurs lecteurs et plusieurs écrivains
- Les Mailslots peuvent se trouver sur le réseau
- Chaque serveur (lecteur) crée un handle Mailslot (CreateMailslot)
- Le serveur peut alors utiliser ReadFile pour lire les messages de son Mailslot
- Le client (écrivain) peut ouvrir un Mailslot avec CreateFile et y écrire des messages avec WriteFile. L'ouverture échouera, si le Mailslot n'a pas été créé par un serveur.

15.3.1 CreateMailslot

- La fonction, **CreateMailslot**, crée un Mailslot nommé et retourne un Handle utilisable par le serveur pour opérer sur le Mailslot (INVALID_HANDLE_VALUE sinon).
- Rem : un Mailslot est local à l'ordinateur qui le crée
- Syntaxe :

  - *HANDLE CreateMailslot(LPCTSTR lpName, DWORD nMaxMessageSize, DWORD lReadTimeout, LPSECURITY_ATTRIBUTES lpSecurityAttributes) ;*

- Arguments :

  - *LPCTSTR lpName, pointeur sur le nom du Mailslot*
  - *DWORD nMaxMessageSize, taille maximale des messages*
  - *DWORD lReadTimeout, Timeout en millisecondes*
  - *LPSECURITY_ATTRIBUTES lpSecurityAttributes, pointeur sur les attributs de sécurité du Mailslot (NULL par défaut)*

15.3.2 Mailslot : exemple de base