Mutex

1 Khái niệm

Mutex là một đối tượng đồng bộ hóa nhận trạng thái TRUE khi không có một tiểu trình nào sở hữu nó, và nhận trạng thái FALSE nếu có một tiểu trình có quyền sở hữu mutex.

2 Tình huống sử dụng

Tại một thời điểm chỉ có một tiểu trình được quyền sở hữu đối tượng mutex, tiểu trình có quyền sở hữu sẽ được quyền truy xuất các tài nguyên mà đối tượng mutex bảo vệ. Do vậy, đối tượng mutex cũng được sử dụng khi cần bảo đảm tại một thời điểm chỉ có một tiểu trình bất kỳ truy xuất đến tài nguyên dùng chung.

Khác với miền găng, ta có thể sử dụng đối tượng mutex để đồng bộ hóa các tiểu trình **trong những tiến trình khác nhau**.

Ví dụ: Giả sử có nhiều tiểu trình trong các tiến trình khác nhau cùng truy xuất đến một cơ sở dữ liệu, các tiểu trình cần được cài đặt cơ chế đồng bộ hóa với đối tượng mutex để bảo đảm mỗi lúc chỉ có một tiểu trình được cập nhật nội dung cơ sở dữ liệu.

3 Các bước thiết lập và sử dụng Mutex

	Sử dụng các hàm API	Sử dụng lớp do MFC hỗ trợ
1. Tạo lập đối tượng Mutex	Tiểu trình gọi hàm CreateMutex() để tạo lập và trả về handle của đối tượng Mutex, hay gọi hàm OpenMutex() để lấy handle của một đối tượng Mutex đã tồn tại khi biết tên đối tượng đó.	CMutex < đối tượng M >
2. Giành quyền sở hữu đối tượng Mutex	Trong các tiểu trình, tại vị trí bắt đầu vào đoạn lệnh truy xuất đến tài nguyên dùng chung cần được bảo vệ, gọi **wait function** để thử giành lấy quyền sở hữu đối tượng Mutex. Nếu Mutex đang có trạng thái signal, thì tiểu trình được sở hữu đối tượng Mutex, và có thể truy xuất tài nguyên dùng chung đang được Mutex bảo vệ, trước khi trả về , **wait function** đặt trạng thái Mutex về nonsignal để ngăn các tiểu trình khác truy xuất tài nguyên được bảo vệ. Ngược lại, nếu Mutex đang có trạng thái	< đối tượng M >.Lock()

	nonsignal, nghĩa là đã có một tiểu trình khác chiếm giữ Mutex, thì hàm **wait function** không thể trả về và tiểu trình gọi nó sẽ chuyển sang trạng thái chờ.	
3. Giải phóng đối tượng Mutex	Trong các tiểu trình , tại vị trí kết thúc đoạn lệnh truy xuất đến tài nguyên dùng chung đang được bảo vệ , gọi hàm ReleaseMutex() để từ bỏ quyền sở hữu đối tượng Mutex , và đặt trạng thái signal cho đối tượng Mutex, tạo cơ hội cho các tiểu trình khác đang chờ có thể truy xuất tài nguyên chung được Mutex bảo vệ.	< đối tượng M >.UnLock()
4. Kết thúc sử dụng đối tượng Mutex	Trong các tiểu trình, khi không còn nhu cầu truy xuất tài nguyên chung được bảo vệ bởi một đối tượng Mutex, gọi hàm CloseHandle() để kết thúc sử dụng đối tượng Mutex tương ứng.	

Lưu ý: các bước 1, 4 thường được thực hiện bởi tiểu trình chính , còn các bước 2, 3 thường do các tiểu trình cần thực hiện việc đồng bộ hoá.

4 So sánh Mutex và CriticalSection

Về ý nghĩa sử dụng, Mutex gần tương tự với CriticalSection. Tuy nhiên, cần chú ý những điểm khác biệt sau:

CcriticalSection	CMutex
Chỉ dùng trong cùng một tiến trình. (Do đó không có tên).	Dùng được cho việc đồng bộ giữa các tiến trình. (Được phép dùng tên).
Không thể chỉ định dwTimeout khi Lock. (Do đó không thể dùng với hàm API của Windows WaitFor***Object).	Được dùng dwTimeout. (Có thể dùng hàm API của Windows WaitFor***Object)

5 Các hàm liên quan

```
HANDLE CreateMutex(
        LPSECURITY_ATTRIBUTE  lpMutexAttributes,
        BOOL                  bInitalOwner,
        LPCTSTR               lpName
);
```

Tạo và trả về handle của đối tượng Mutex. Ý nghĩa các tham số:

- ***lpMutexAttributes***: cho biết thuộc tính bảo mật của đối tượng Mutex được tạo ra. Nếu tham số này là NULL thì đối tượng Mutex được tạo ra sẽ có thuộc tính bảo mật mặc định.
- ***bInitalOwner***: nếu mang giá trị TRUE thì tiểu trình gọi hàm sẽ sở hữu đối tượng Mutex ngay khi vừa tạo ra.
- ***lpName***: khi muốn chia sẽ đối tượng Mutex cho các tiểu trình giữa các tiến trình khác nhau để cùng bảo vệ 1 tài nguyên chung nào đó thì ta cần đặt 1 tên cho đối tượng mutex. Nếu không muốn ta có thể đặt giá trị của tham số này là NULL. Khi giá trị khác NULL, có thể có các trường hợp xảy ra:
  - Trường hợp 1: tên hợp lệ không có kí tự ' \ ' và độ dài trong khoảng MAX_PATH.
  - Trường hợp 2: tên trùng với một đối tượng Mutex đã tồn tại thì đối tượng Mutex đã tồn tại phải được thiết lập với cờ MUTEX_ALL_ACCESS truy nhập. Và khi này tham số thứ hai sẽ không có được xem xét.
  - Trường hợp 3: tên trùng với một tên định danh của các đối tượng event, semaphore, waitable timer, job , hay file-mapping đã tồn tại thì hàm này sẽ thất bại .

```
HANDLE OpenMutex(
            DWORD     dwDesireAccess,
            BOOL      bInheritHandle,
            LPCTSTR   lpName
);
```

Gọi 1 đối tượng Mutex đã tồn tại và hàm sẽ trả về handle của đối tượng mutex đó. Khi 1 tiến trình tạo 1 đối tượng mutex với 1 tên xác định thì tiến trình khác sẽ gọi hàm này với tên mutex để nhận được handle của đối tượng mutex.

- ***dwDesireAccess***: chỉ định chế độ truy xuất MUTEX_ALL_ACCESS hay SYNCHRONIZE.
- ***bInheritHandle***: cho biết giá trị handle trả về có được kế thừa hay không. Nếu là TRUE thì các tiến trình tạo bởi lệnh CreateProcess có thể kế thừa handle này, ngược lại thì không kế thừa được.
- ***lpName***: tên định danh của đối tượng Mutex đã tồn tại.

```
BOOL ReleaseMutex(
        HANDLE    hMutex
);
```

Loại bỏ quyền sở hữu đối tượng Mutex và đặt trạng thái TRUE cho đối tượng Mutex để các tiểu trình khác có thể sở hữu được .Tham số truyền vào là handle của đối tượng Mutex.