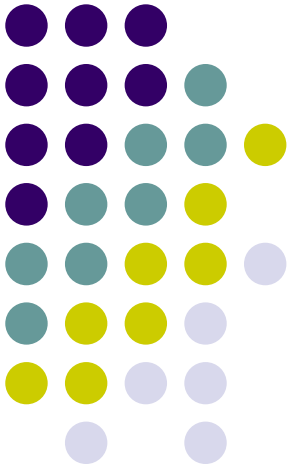
멀티 쓰레드 동기화 기법
: CriticalSection 기반

편집 김혜영



실행순서의 동기화.

메모리 접근에 대한 동기화

쓰레드 동기화 기법의 두 가지 구분.

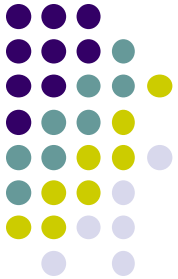
- 크리티컬 섹션 기반 동기화
- 인터락 함수 기반 동기화

유저 모드 동기화

- 뮤텍스 기반 동기화
- 세마포어 기반 동기화
- 이름있는 뮤텍스 기반 동기화
- 이벤트 기반 동기화

커널 모드 동기화

스레드 조작 – 스레드 종료 대기



- WaitForSingleObject() 함수
  - 특정 스레드가 종료할 때까지 대기

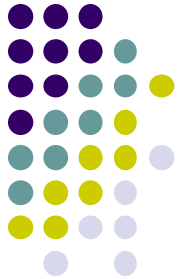
```
DWORD WaitForSingleObject (
    HANDLE hHandle,
    DWORD dwMilliseconds
) ;
```

성공: **WAIT_OBJECT_0** 또는 **WAIT_TIMEOUT**,
실패: **WAIT_FAILED**

- WaitForSingleObject() 함수 사용 예

```
HANDLE hThread = CreateThread(...);
WaitForSingleObject(hThread, INFINITE);
```

스레드 조작 – 스레드 종료 대기

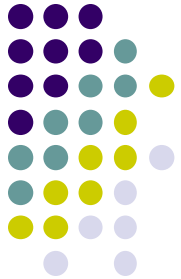


- WaitForMultipleObjects() 함수
  - 두 개 이상의 스레드가 종료할 때까지 대기

```
DWORD WaitForMultipleObjects (
   DWORD nCount,
   const HANDLE* lpHandles,
   BOOL bWaitAll,
   DWORD dwMilliseconds
) ;
```

성공: WAIT_OBJECT_0 ~ WAIT_OBJECT_0 + nCount-1
또는 WAIT_TIMEOUT,
실패: WAIT_FAILED

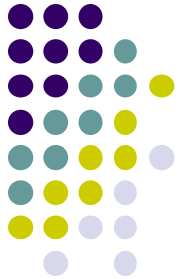
스레드 조작 – 스레드 종료 대기



- WaitForMultipleObjects() 함수 사용 예 ②

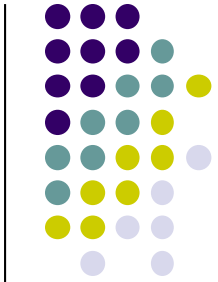
```
// 두 스레드 중 하나의 종료를 기다릴 경우
HANDLE hThread[2];
HANDLE hThread[0] = CreateThread(...);
HANDLE hThread[1] = CreateThread(...);
DWORD retval = WaitForMultipleObjects(2, hThread, FALSE, INFINITE);
switch(retval){
case WAIT_OBJECT_0:          // hThread[0] 종료
    break;
case WAIT_OBJECT_0+1:  // hThread[1] 종료
    break;
case WAIT_FAILED:            // 오류 발생
    break;
}
```

크리티컬 섹션 기반의 동기화



```
CRITICAL_SECTION   gCriticalSection;   // critical section object

InitializeCriticalSection(&gCrigicalSection);  // initialize object
. . . . .

// 임계영역 진입을 위해 크리티컬 섹션 오브젝트 획득
EnterCriticalSection(&CriticalSection);

        임   계   영   역

// 크리티컬 섹션 오브젝트 반환
LeaveCriticalSection(&CriticalSection);

 . . . . .
DeleteCriticalSection(&CriticalSection);
```

크리티컬 섹션 기반의 동기화: 예제 확인



CriticalSectionSync.cpp