윈도우는 선점형 멀티스레딩 (Preemptive Multi-Threading)

현재 프로세서(ex:CPU)를 사용하는 스레드를 임의로 멈추고,

다른 스레드로 사용권을 넘길 수 있음.

#프로세서가 정확한 표현이지만 프로세스와 헷갈리지 않기 위해 CPU라고 칭함.

멀티 스레딩 시분할 방식으로 딩시에 돌아가게끔 보이게 함.

멀티 코어는 실제로 동시에 여러 스레드가 실행 됨.

스케줄링 Time-Slice (Quantum)

CPU 시간을 할당하는 순서 제어.

프로세스

사용하는 자원들이 속하는 프로그램 사이의 경계.

프로세스 간 통신하려면 다른 수단을 써야함 IPC Inter-Process Communication.

프로세스 요소

{

ID

로컬 시스템에서 고유한 ID.

핸들과는 다름. 프로세스 커널 객체를 참조하는 핸들값은 프로세스에 대해 상대적임.

보안 문맥

SID 로그인된 유저에 대한 계정 ID와 소속 그룹 ID 리스트, 액세스 토큰을 통해 보안 객체 접근 권한 여부를 운영체제에게 허가 받음.

스레드는 기본적으로 프로세스의 액세스 토큰을 사용.

가상 주소 공간

선형. 32비트는 4기가, 64비트는 이론적으로 16엑사 하드웨어 특성상 8테라 바이트까지.

페이지 단위로 물리적 메모리와 매핑.

가상이기 때문에 개별 프로세스는 각각 메모리를 저만큼 사용할 수 있다고 여김.

커널 영역과 유저 영역.

프로그램 인스턴스

실행파일이 가상 주소 공간에 매핑되어 메모리에 로드된 상태.

핸들 테이블 배열

유저 애플리케이션은 커널 주소 공간에 직접 접근할수 없고 API로 생성된 커널 객체의 포인터가 핸들 테이블에 보관되어 그 인덱스인 핸들값을 참조함.

메인 스레드

WinMain, main함수 내의 코드를 실행할 주체.

메인 스레드가 종료되면 프로세스도 종료됨.

}

스레드 요소

{

ID, 보안문맥

스레드 문맥 Context

CONTEXT 구조체. 멤버는 CPU내의 레지스터들

문맥 전환 Context Switch. 이 정보들을 저장했다가 복원.

스레드 스택 Stack

기본 1MB. 생성시 설정.

커널 모드, 유저 모드로 두개.

TLS thread local storage

ex:MFC에서 그 스레드가 생성한 핸들과 관련된 정보들.

//스레드 풀 사용 등에서 주의해야 함.

스레드 메시지 큐

순차성 보장, 지연 처리, 스레드 간 매개변수 전달. 등 동기화 해결.

}

스레드 종료

{

엔트리 함수가 return되는 것이 정상적.

ExitThread

해당 스레드가 스스로 종료하는 함수.

C에선 큰 문제가 없지만

C++에선 엔트리 함수 내에서 선언한 지역 변수의 클린업(소멸자 함수 호출)이 안됨.

TerminateThread

다른 스레드가 해당 스레드의 핸들을 넘겨서 종료하는 함수.

해당 스레드는 유저 모드 코드를 실행할 수 없게 됨. 연결된 DLL은 종료 통지를 못받음.

크리티컬 섹션, 힙의 락 해제 안됨. 위험.

의도하지 않은 종료.

메모리 접근 관련 예외 같은 경우 프로세스 자체가 다운. (엔트리 함수 종료)

이에 대비하기 위해 구조적 예외 처리 SEH Structured Exception Handling와 try...catch 결합

코드 생성 - 예외 처리 가능 - SEH

}