운영체제 (062 분반)

Ch 04. Thread Assignment

학과: 정보컴퓨터공학부
학번: 20192443 7
이름: 김윤하
제출일: 2023.04.10

4.4 User Thread와 Kernel Thread의 특징은 다음과 같다.

〈 사용자 스레드 〉 User Thread	〈 커널 스레드 〉 Kernel Thread
- 커널 개입 없이 사용자 영역의 스레드 library 활용하는 방식.	- 커널이 관리하는 스레드. OS Scheduler의 최소 단위.
- 커널 호출이 없으므로 오버헤드가 작음.	- 하나의 Process는 적어도 Thread 하나 포함.
- 시스템 전반의 Scheduling 우선순위 지원X (대신 Context-Switching만 내부에서)	- 커널 영역에서 스레드 연산 수행 및 관리 (커널에 종속적임!)
- System Call 하면 Process 내의 모든 스레드가 Block 된다. (커널 실행하는 동안 스레드가 중단됨)	- 커널이 직접 Processor에 스케줄링, 동기화 하여 안정적으로 많은 기능을 제공함.
	- User mode <-> Kernel mode 전환이 빈번하여 성능이 저하되고 구현이 어렵다.

정리하자면, 사용자 스레드는 Kernel 호출이 없어 오버헤드가 적은 반면, 시스템 콜 시 스레드가 중단되는 단점이 있다. 또한 커널 스레드의 경우에는 User / Kernel mode 간 전환이 많아 오버헤드가 큰 반면, 안정적이고 많은 기능을 제공한다.

따라서, 디스크 I/O 와 같은 경우 System Call 을 많이 활용하는 Kernel Thread 를 사용하는 것이 실행의 측면에서 효율적이다. 반대로 Blocking 연산이 적은 상황 또는 단일 Processor System 에서는 사용자 스레드가 효율적이다. (경량화됨)

4.5

Context - Switch 오버헤드가 큰 커널 스레드 간의 전환 동작은 다음과 같다.

1. 현재 실행 중인 Thread의 PCB 정보를 저장.
   * Register 상태, Stack Pointer, Instruction Pointer 정보 등.
2. 다음 실행될 Thread의 PCB 정보를 로드.
3. Context - Switching.
   → 커널은 현재 실행 중인 스레드의 내용을 저장하고,
   다음 실행될 스레드의 내용을 복원. ⇒ 현재 Thread 중단 + 다음 Thread 실행
4. 다음 Thread 실행.

- 커널이 직접 Context - Switching을 수행하므로, 비용이 많이 든다. (OS 에서 관리)