인덕대학교

프로세스 관리
- 스레드

컴퓨터소프트웨어학과

김병국 교수

학습목표

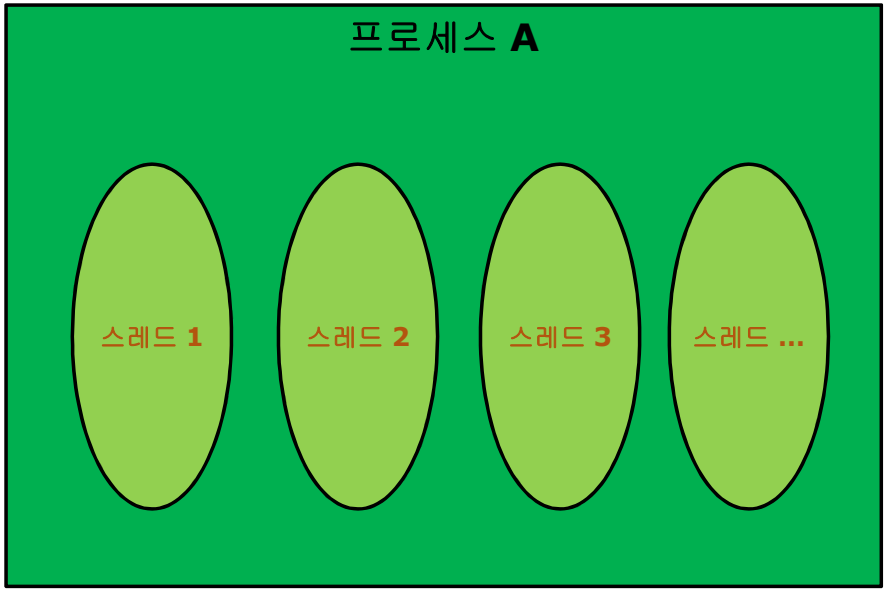
- 스레드의 개념을 안다.
- 스레드 제어 블록에 대한 기능을 안다.
- 멀티 스레드, 멀티 태스킹, 멀티 프로세싱을 구분할 수 있다.
- 멀티 스레드를 위한 모델을 익힌다.

목차

1. 스레드의 개념 (1/2)

❑ 스레드의 정의

- CPU 스케줄러가 CPU에 전달하는 일들 중 하나
- 스레드: 프로세스의 최소 단위
  - 하나의 프로세스에는 여러 개의 스레드를 구성할 수 있음



1. 스레드의 개념 (2/2)

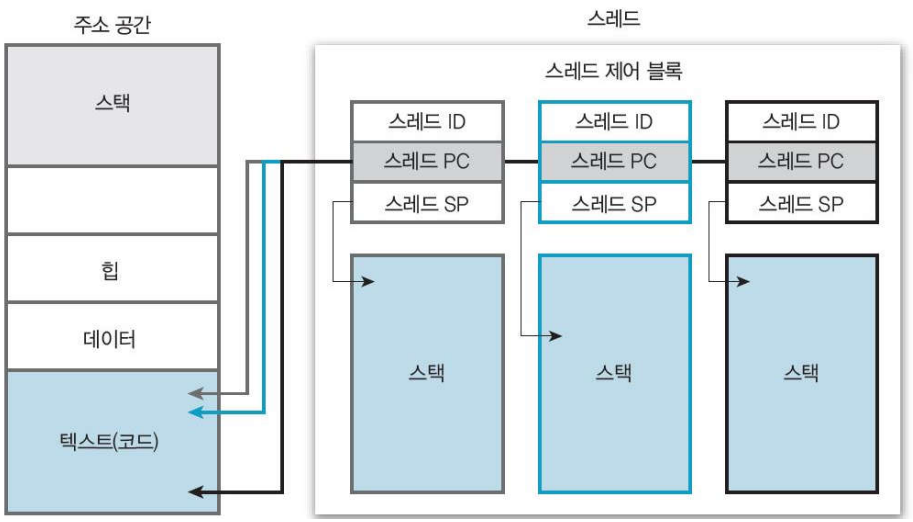
❑ 멀티 태스크와 멀티 스레드의 차이

- 멀티 태스크(Multi-tasks)
  - 하나의 업무수행을 위해 여러 개의 프로세스들로 구성 시키는 것
  - 구현을 위해 프로세스간의 통신기법(IPC: Inter-Process Communication 기술이 필요)

- 멀티 스레드(Multi-threads)
  - 하나의 프로세스에 여래 개의 스레드로 구성 시키는 것
  - 하나의 프로세스를 공유하기 때문에, 모든 스레드는 전역 메모리 영역 등을 공유함

※ 결과적으로 동일한 업무를 수행하더라도,
구현 및 내부 동작의 차이로 이해하면 됨

2. 스레드 제어 블록



❑ TCB(Thread Control Block)

- 스레드의 정보를 저장
- PCB는 TCB리스트를 포함
- 프로세스와는 달리 스레드 간의 보호는 없음
  - 스레드 간의 모든 자원은 공유됨

- TCB의 주 내용
  - 실행 상태:
    - 레지스터, 프로그램 카운터, 스택 포인터
  - 스케줄링 정보:
    - 상태(실행, 준비, 대기 등),
    - 우선순위 등
  - 계정 정보
  - 스케줄링 큐를 위한 다양한 포인터
  - PCB를 위한 주소 포인터

3. 멀티 스레드/태스킹/프로세싱 (1/5)

❑ 멀티 스레드

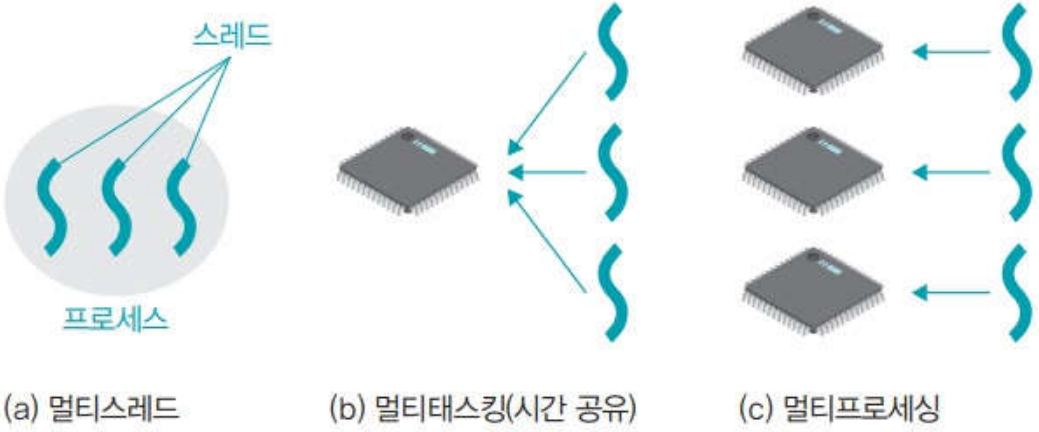
- 프로세스 내 작업을 여러 개의 스레드로 분할함으로써 작업의 부담을 줄이는 프로세스 운영 기법

❑ 멀티 태스킹

- 운영체제가 CPU에 작업을 줄 때 시간을 잘게 나누어 배분하는 기법

❑ 멀티 프로세싱

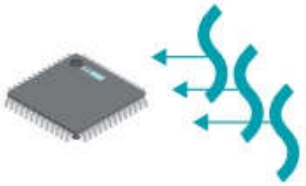
- CPU를 여러 개 사용하여 여러 개의 스레드를 동시에 처리하는 작업 환경



3. 멀티 스레드/태스킹/프로세싱 (2/5)

❑ CPU 멀티 스레드

- **파이프라인 기법**을 이용
- 동시에 여러 스레드를 처리하도록 만든 **병렬 처리 기법**
  - 멀티 스레드 :
    - 운영체제가 **소프트웨어적**으로 프로세스를 작은 단위의 스레드로 분할
  - CPU 멀티스레드 :
    - **하드웨어적인 방법**
    - 하나의 CPU에서 여러 스레드를 동시에 처리



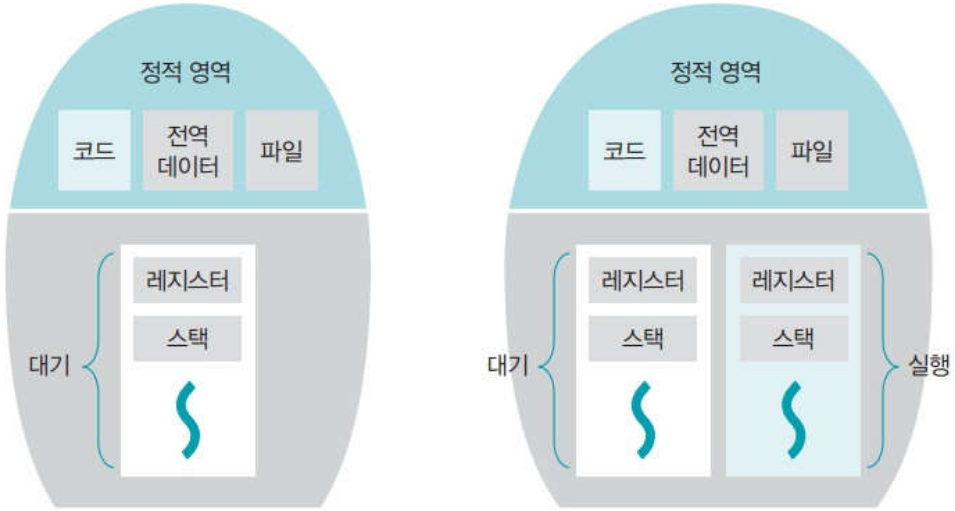
(d) CPU 멀티스레드

3. 멀티 스레드/태스킹/프로세싱 (3/5)

인덕대학교

❑ 멀티스레드의 장점

- 응답성 향상
- 자원 공유
- 다중 CPU 지원
- 효율성 향상



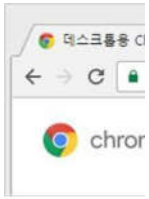
(a) 단일 스레드 (b) 멀티스레드

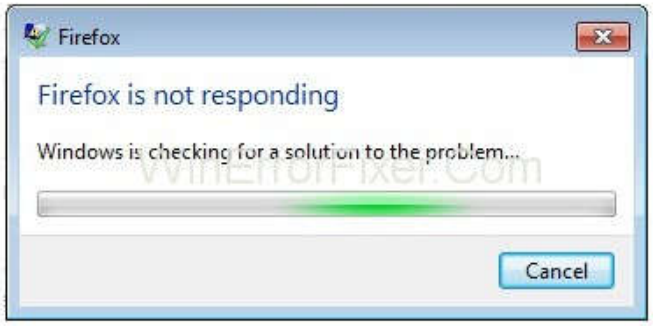
그림 3-34 단일 스레드와 멀티스레드의 구조

3. 멀티 스레드/태스킹/프로세싱 (4/5)

❑멀티스레드의 단점

- 자원 공유로 인한 오류정보 또한 공유되어 전체 프로세스에 영향을 줌





3. 멀티 스레드/태스킹/프로세싱 (5/5)

※ 하이퍼 스레드딩(Hyper-Threading Technology)

- 인텔이 멀티스레딩을 위해 구현한 기술
- 물리적인 코어 하나를 가상의 논리적인 코어 두 개로 할당하는 기술
- 운영체제는 다중 코어(멀티 프로세서)로 인식하여, 다중의 프로세스를 동작시키기 위한 기법으로 운영
  - 결과: 프로세스의 대기시간 단축 및 성능 향상 발생

4. 멀티 스레드 모델 (1/4)

❑종류

- 사용자 스레드
  - 사용자 프로그램(응용프로그램 등)에 의해 구현된 일반적인 스레드
- 커널 스레드
  - 커널이 내부적인 동작을 위해 사용되는 스레드
  - 커널이 프로세스를 위한 스레드를 직접 생성 및 관리
- 멀티레벨 스레드
  - 사용자 스레드와 커널 스레드의 혼합

4. 멀티 스레드 모델 (2/4)

❑ 사용자 스레드

- 사용자 프로세스 내에 여러 개의 스레드가 커널의 스레드 하나와 연결(1 to N 모델)
- **라이브러리가 직접 스케줄링 →** 문맥 교환이 필요 없음
- 커널 스레드가 입출력 작업을 위해 대기 상태에 들어가면 모든 사용자 스레드가 같이 대기하게 됨
- 한 프로세스의 타임 슬라이스를 여러 스레드가 공유하기 때문에 여러 개의 CPU를 동시에 사용할 수 없음

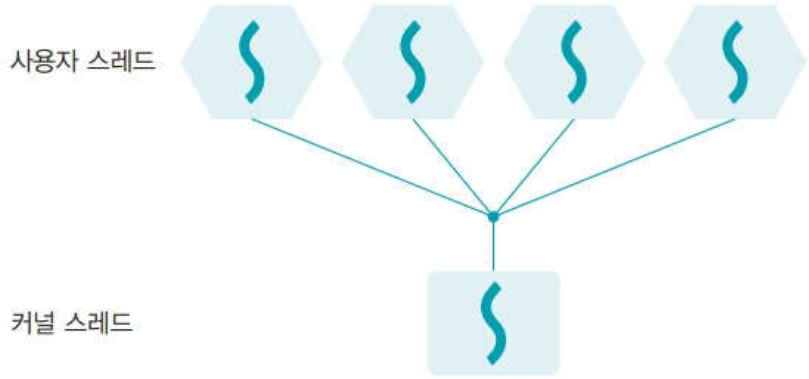


그림 3-36 **사용자 레벨 스레드(1 to N 모델)**

4. 멀티 스레드 모델 (3/4)

❑ 커널 스레드

- 하나의 사용자 스레드가 하나의 커널 스레드와 연결(1 to 1 모델)
- 하나의 스레드가 대기 상태에 있어도 다른 스레드는 작업을 계속할 수 있음
  - 독립적으로 스케줄링이 되므로 특정 스레드가 대기 상태에 들어가도 다른 스레드는 작업을 계속할 수 있음
- 커널 레벨에서 모든 작업을 지원하기 때문에 **멀티 CPU를 사용할 수 있음**
- 커널의 기능을 사용하므로 보안에 강하고 안정적으로 작동
- 문맥 교환 시 오버헤드 발생 → 느리게 작동

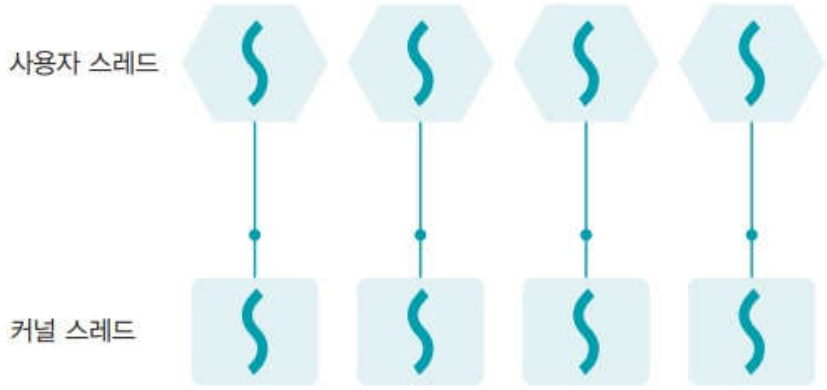


그림 3-37 **커널 레벨 스레드(1 to 1 모델)**

4. 멀티 스레드 모델 (4/4)

❑ 멀티레벨 스레드

- 사용자 스레드와 커널 스레드를 혼합한 방식(M to N 모델)
- 커널 스레드가 대기 상태에 들어가면 **다른 커널 스레드가 대신 작업**을 하여 사용자 스레드보다 유연하게 작업을 처리할 수 있음
- 커널 스레드를 같이 사용하기 때문에 여전히 문맥 교환 시 오버헤드가 있어 사용자 스레드만큼 빠르지 않음

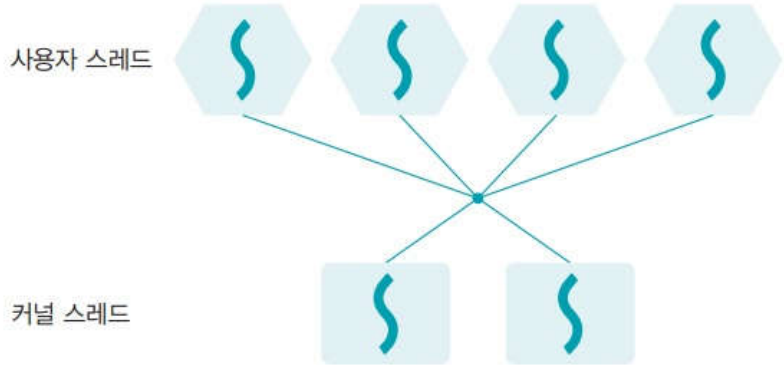


그림 3-38 **멀티레벨 스레드(M to N 모델)**

수고하셨습니다.