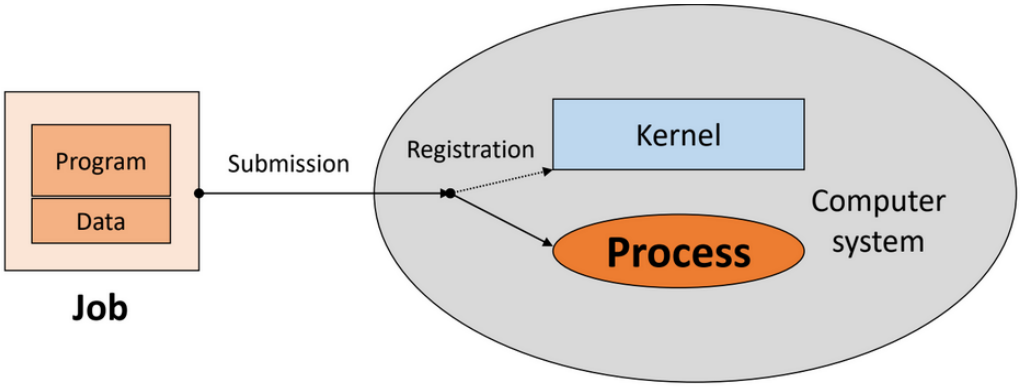
03. Process Management

프로세스



- Job/Program
  - 실행할 프로그램+ 데이터
  - 시스템에 실행 요청 전의 상태(디스크에만 있음)
- Process
  - 실행을 위해 시스템(커널)에 등록된 작업
  - 메모리에 적재됨

프로세스

- 실행 중인 프로그램
  - 커널에 등록되어 관리되는 작업 ⇒ PCB 할당 받음
  - 각종 자원 요청 및 할당 가능 ⇒ 능동적인 개체 (active entity)

[프로세스 분류]

- 역할에 따른 분류
  - 시스템(커널) 프로세스: 커널 작업 수행
  - 사용자 프로세스: 사용자 코드 수행
- 병행 수행 방법에 따른 분류
  - 독립 프로세스: 다른 프로세스와 영향 주고받기 X

- 협력 프로세스

자원

- 커널의 관리 하에 프로세스에게 할당/반납되는 수동적 개체(passive entity)
  - HW/SW 자원

PCB (Process Control Block)

- OS에서 프로세스 관리에 필요한 정보 저장한 자료 구조 (프로세스 상태 정보)
  - 프로세스 생성 시 생성, 커널 관리 영역에 저장됨
- OS별로 정보 다름, PCB 참조/갱신 속도와 OS 성능은 밀접한 연관 갖고 있음

PBC가 관리하는 정보

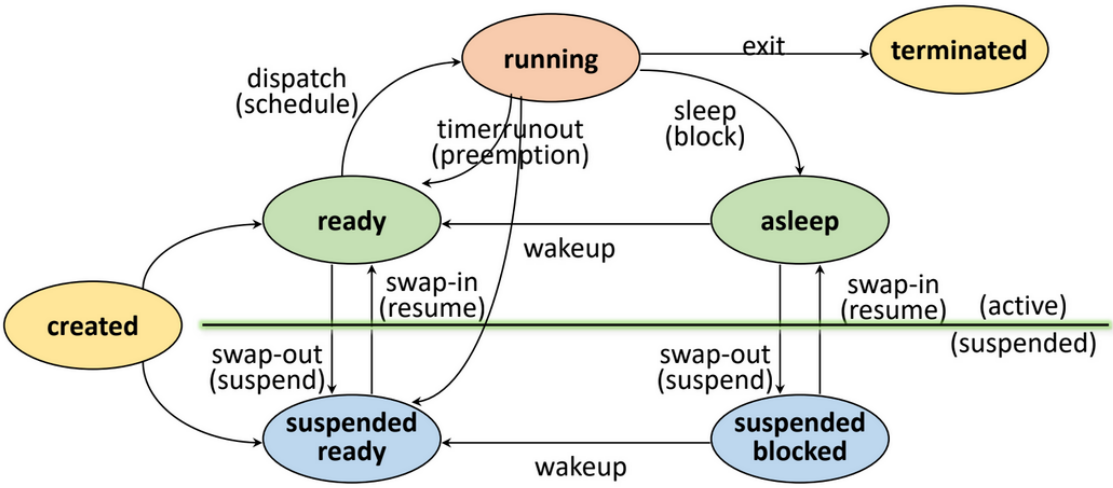
- PID: 프로세스 고유 식별 정보
- 스케줄링 정보: 프로세스 우선 순위 등
- 프로세스 상태: 자원 할당/요청 정보 등
- 메모리 관리 정보: Page table, segment table 등
- 입출력 상태 정보
- 문맥 저장 영역(context save area): 프로세스의 레지스터 상태를 저장하는 공간 등
- 계정 정보: 자원 사용 시간 등 관리

프로세스 상태 (Process States)

- 프로세스-자원 간 상호 작용에 의해 결정

상태		자원 할당 상태	
Active (swapped-in)	Running	프로세서 O	메모리 O
	Ready	프로세서 X, 기타 자원 O	
	Blocked, asleep	프로세서 X, 기타 자원 X	
Suspended (Swapped-out)	Suspended ready	프로세서 X	메모리 X
	Suspended block	프로세서 X, 기타 자원 X	

Process State Transition Diagram



Created State

- 작업이 커널에 등록된 상태: PCB 할당 및 프로세스 생성됨
- 상태 전이: 가용 메모리 공간 존재 여부에 따라
  - 메모리 할당 가능 ⇒ READY
  - 메모리 할당 대기 ⇒ SUSPENDED READY

Ready State

- 프로세서 X, 다른 모든 자원은 할당 받은 상태: CPU만 있으면 즉시 실행 가능
- 상태 전이
  - 프로세서 할당 받으면(dispatch/schedule) ⇒ RUNNING

Running State

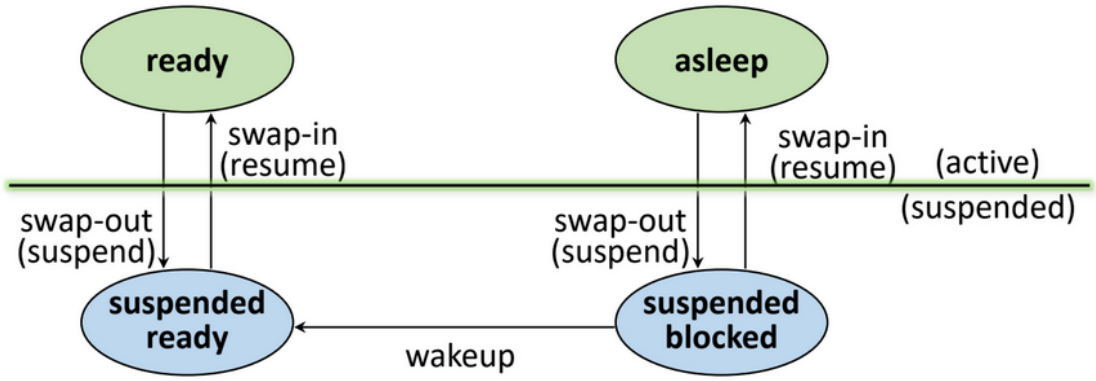
- 필요한 자원을 모두 할당받아 작업 실행하고 있는 상태
- 상태 전이
  - 프로세서 뺏긴 경우(preemption) ⇒ READY
    - ex) 프로세서 스케줄링
  - 다른 작업을 위해 잠깐 대기 상태(block/sleep) ⇒ ASLEEP
    - ex) I/O, 메모리 read 등 자원 할당 요청

Blocked/Asleep State

- 프로세서 외 다른 자원의 할당 기다리는 상태: 자원 할당은 system call에 의해 이루어짐, 이때 프로세서 X)
- 상태 전이: 곧장 running으로 복귀하는 게 아니라 ready로 이동
  - 다른 자원 대기 종료(wake up) ⇒ READY

Suspended State

- 메모리를 할당 받지 못한(빼앗긴) 상태
  - 메모리 빼앗기면, 그 순간의 memory image를 swap device(프로그램 정보 저장 위한 파일 시스템)에 보관해 작업 복귀하면 사용할 수 있도록 함
  - 커널 또는 사용자에 의해 발생

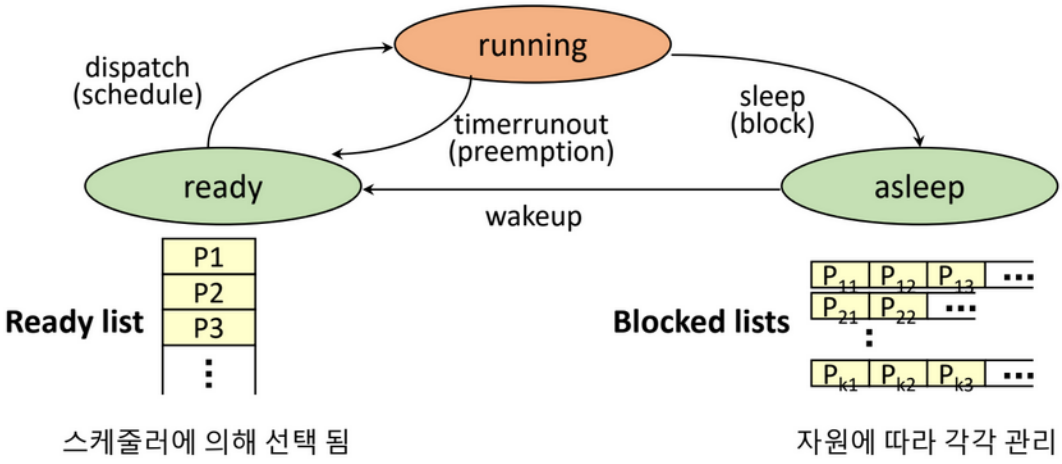


- Suspended ready: created 상태(프로세서X)에서 메모리가 없고 대기하는 상태
- Suspended blocked: blocked 상태(프로세서 X, 다른 자원 X)에서 메모리를 빼앗긴 상태

Terminated/Zombie State

- 프로세스 수행이 끝난 상태
  - 모든 자원 반납 후, 커널 내 일부 PCB 정보만 남아있음
- 커널이 PCB 정보를 수집해 이후 프로세스 관리에 사용하기 위해 잠깐 들렀다가는 state, 이후 프로세스 소멸

프로세스 관리를 위한 자료구조



- Ready Queue: ready 상태의 프로세스가 줄 서 있는 공간
- Device, I/O Queue: asleep 상태의 프로세스에 대해 자원별로 Queue 각각 관리

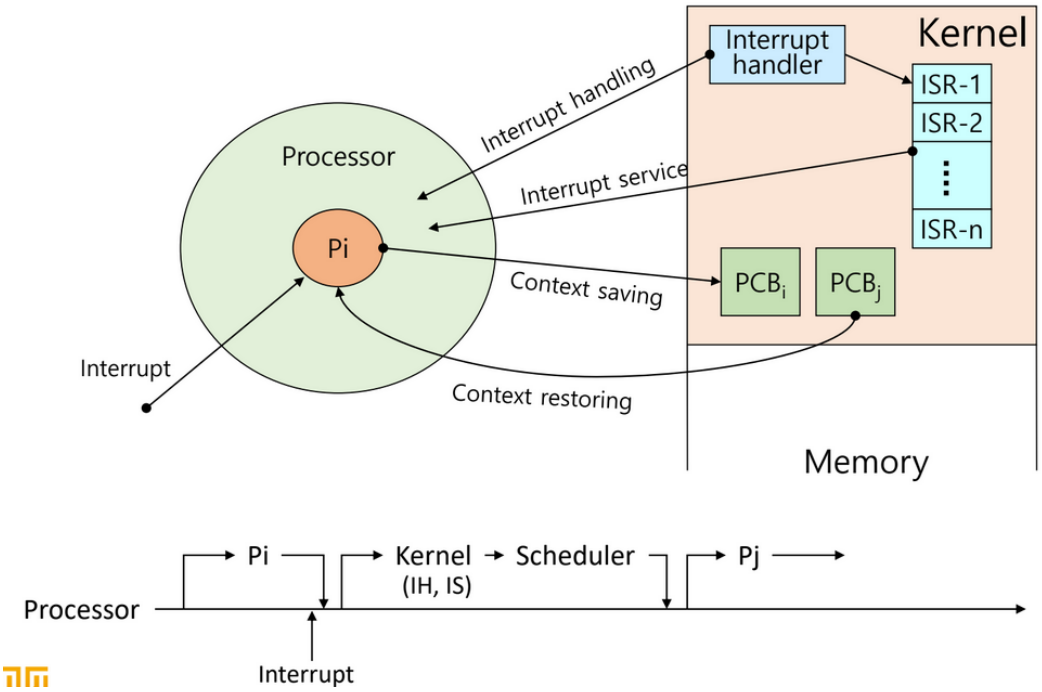
인터럽트

- 예상치 못한, 외부에서 발생한 이벤트
- 종류
  - I/O Interrupt: 입출력 장치에 의해 발생
  - Clock Interrupt: CPU에 의해 발생하는 인터럽트, 멀티태스킹 위해 사용

인터럽트 처리 과정

- 인터럽트 발생 ⇒ [커널 개입] 프로세스 중단 ⇒ 인터럽트 처리(Interrupt Handling)
- 인터럽트 처리
  - 인터럽트 발생 장소, 원인 파악
  - 인터럽트 서비스 여부 결정, 결정하면 인터럽트 서비스 루틴 호출

[인터럽트 상세 처리 과정]



1. 인터럽트 발생
2. 커널이 프로세스 중단, PCB에 Context Saving ⇒ 프로세스 흐름 저장
3. 인터럽트 처리: 인터럽트 발생 장소, 원인 파악
4. 인터럽트 서비스: 인터럽트 서비스 프로세스에 프로세서 할당
5. 인터럽트 서비스 종료 후, Ready 상태에 있던 프로세스에 프로세서 할당 (기존 서비스 아닐수도)
    - 이때 Context restoring 발생

Context Switching

- Context: 프로세스와 관련된 정보들의 집합
    - CPU register context는 CPU에 저장 (CPU 작업은 언제나 레지스터에 저장되어야 함)
    - code, data, stack, pcb 등은 메모리에 저장
- Context saving
    - 현재 프로세스의 register context를 저장 ⇒ 메모리(PCB)에 저장
- Context restoring
    - 이전에 저장한 register context를 프로세스로 복구하는 작업 ⇒ 메모리에 있던 걸 CPU로

- Context switching
    - 실행 중인 프로세스의 context를 저장하고, 다음 실행될 프로세스의 context 복구

Context Switch Overload

- context switching에 소요되는 비용
- OS 성능에 큰 영향을 주기 때문에 thread 등 사용해서 불필요한 context switching 줄이려고 함