프로세스의 개념

- 운영체제는 프로세스로 실행되는 다양한 프로그램을 실행
- 프로세스(process)

- 실행 중인 프로그램 => 활성 엔티티(active entity)

- 프로그램 : 디스크의 수동 엔티티(실행 파일)

- 실행 파일이 메모리에 로드 => 프로그램 처리

- 프로세스 실행이 순차적 진행 => 프로그램 카운터 (Program counter)

- GUI 마우스 클릭, 이름을 command line에 입력 등을 통해 시작
- 하나의 프로그램은 여러 프로세스가 된다.

- 동일한 프로그램을 실행하는 여러 사용자를 고려

- 다양한 정보

- 텍스트 섹션 (Text section) : 프로그램 코드

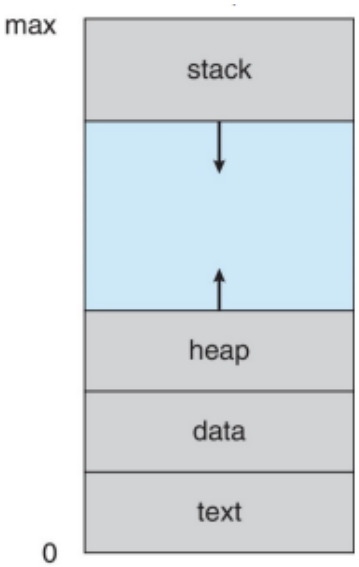
- 현재 활동 (Current activity) : 프로그램 카운터, cpu 레지스터

- 스택 (Stack) : 임시 데이터

- 함수 매개 변수, 반환 주소, 로컬 변수

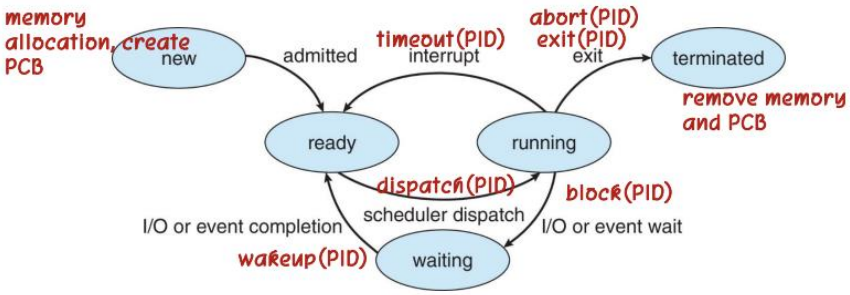
- 데이터 섹션 (Data section) : 전역 변수

- 힙 (Heap) : 실행 시간동안 동적으로 할당된 메모리



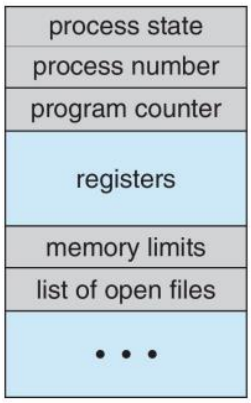
프로세스 상태

- 프로세스가 실행되면 상태가 변경된다.
  - New : 프로세스를 만드는 중이다.
  - Running : 명령이 실행 중이다.
  - Waiting : 프로세스가 일부 이벤트가 발생하기를 기다리는 중이다.
  - Ready : 프로세스가 프로세서에 할당되기를 기다리는 중이다.
  - Terminated : 프로세스가 실행을 마친다.
- 프로세스의 상태의 diagram



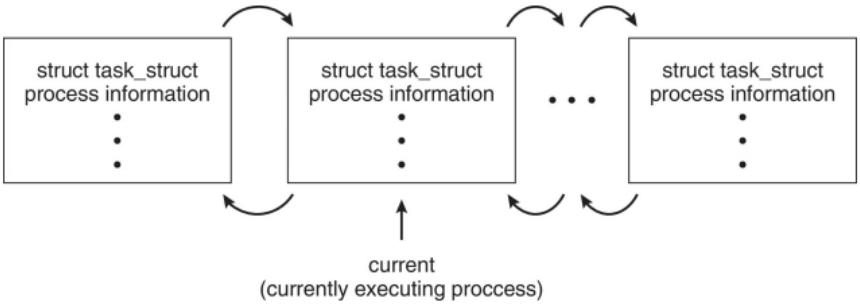
Process Control Block (PCB)

- 각 프로세스와 관련된 정보를 위한 OS 데이터 구조
  - 작업 제어 블록(task control block)이라고도 함
- PCB 정보
  - 프로세스 상태 : 실행, 대기 등
  - 프로세스 번호 : 프로세스 식별자 (PID)
  - 프로그램 카운터 : 실행할 다음 명령의 위치
  - CPU 레지스터 : 모든 프로세스 중심의 레지스터의 내용
  - CPU 스케줄링 정보 : 우선 순위, 스케줄링 큐 포인터
  - 메모리 관리 정보 : 프로세스에 할당된 메모리
  - Accounting information : 사용된 cpu, 시작 후 경과된 clock 시간, 시간 제한
  - I/O status information : 프로세스에 할당된 I/O 장치, open된 파일 목록



Process Representation

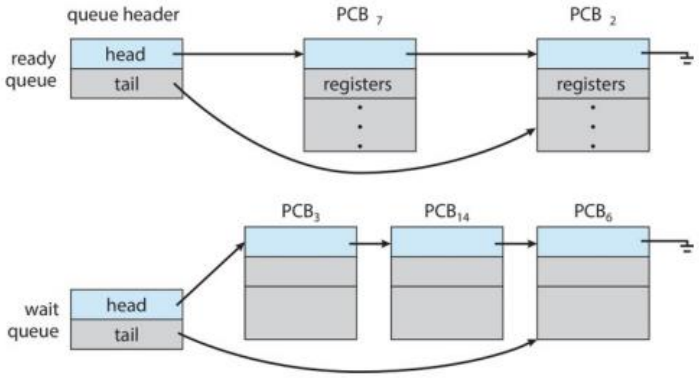
C structure



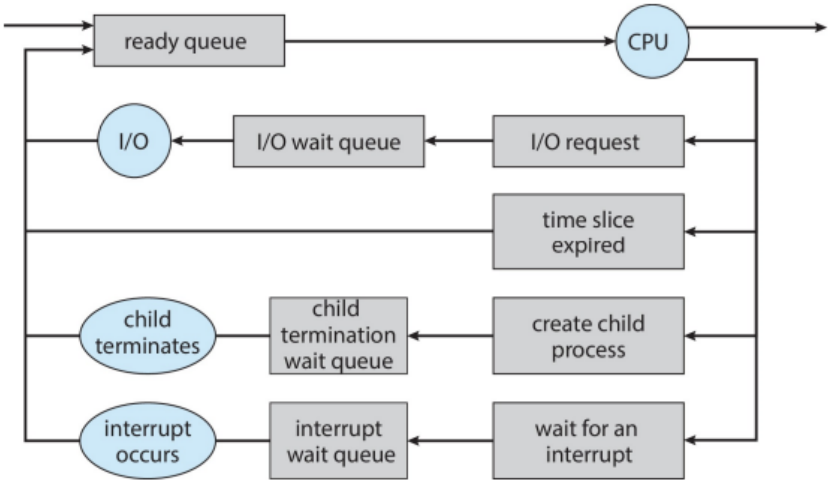
Process Scheduling

- CPU 사용을 극대화하며, 프로세스를 CPU 코어로 신속하게 전환한다.
- Process scheduler가 다음에 사용할 수 있는 프로세스 중에서 선택한다.
- 프로세스의 scheduling queues를 유지 관리한다.

- Ready queue : 메인 메모리에 상주하며 실행 준비 및 대기 중인 모든 프로세스
- Wait queue : 이벤트를 대기하는 프로세스 (예: I/O)
- 프로세스는 다양한 큐 사이로 이동한다.



Process scheduling diagram

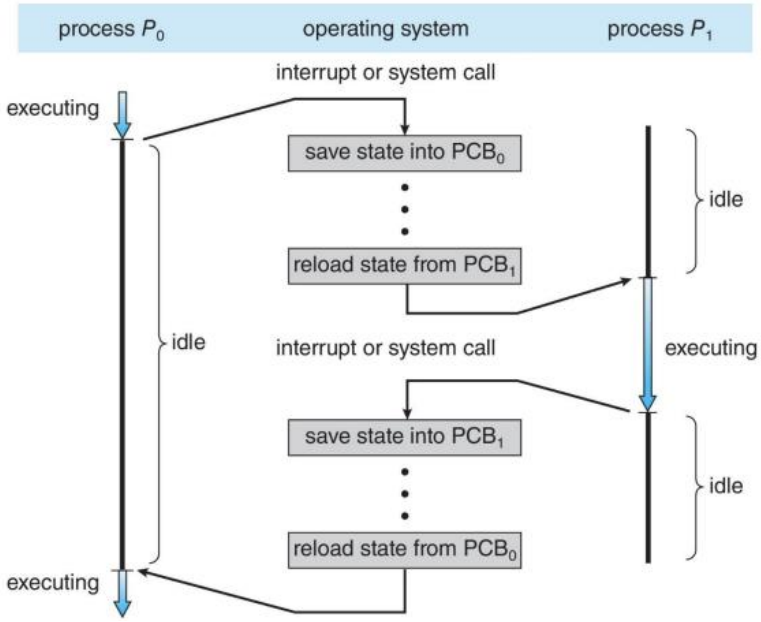


Context Switch

- Context switch는 CPU가 한 프로세스에서 다른 프로세스로 전환될 때 발생한다.
- CPU가 다른 프로세스로 전환되면
  - 시스템은 이전 프로세스의 상태를 저장해야 한다.
  - Context switch를 통해 새 프로세스를 위한 저장된 상태를 로드한다.
- PCB에 표시되는 프로세스의 컨텍스트
- Context-switch 시간이 오버헤드이므로 전환 중에는 시스템이 작동하지 않는다.
  - OS와 PCB가 복잡할수록 context switch가 길어진다.
- 하드웨어 지원에 따른 시간

- 일부 하드웨어는 한 번에 로드되는 CPU당 여러 세트의 레지스터를 제공한다.

프로세스 간 CPU 전환



Multitasking in Mobile System

- iOS
- 초기에는 한 개의 사용자 프로세스만 실행되도록 허용하고 다른 프로세스는 사용자 인터페이스 제한(화면, 터치)으로 인해 일시 중단한다.
- OS가 multi-tasking으로 실행
- iOS4부터
- 단일 포그라운드 프로세스 – 사용자 인터페이스를 통해 제어
- 다중 백그라운드 프로세스 – 메모리에서 실행. 디스플레이에서 표시되지 않으며 제한이 있다.
- 제한 사항에는 single task, short task, 이벤트 알림 수신, 오디오 재생과 같은 특정 장기 실행 작업이 포함된다.
- 현재는 multi-tasking과 display division을 사용한다.

- Android는 더 적은 제한으로 포그라운드 및 백그라운드를 실행

- 백그라운드 프로세스가 서비스를 사용하여 작업을 수행한다.
- 백그라운드 프로세스가 일시 중단된 경우에도 서비스가 계속 실행된다.
- 서비스에 사용자 인터페이스가 없고, 메모리 사용량이 적음