프로세스 관리

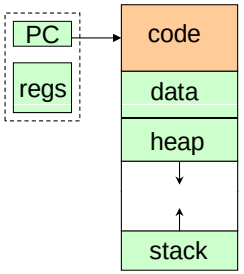
운영체제

연세대학교

1. 프로세스 개념

연세대학교

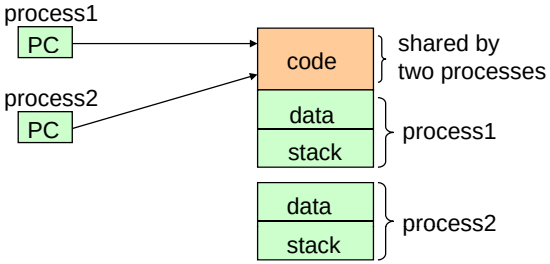
- 프로그램(program)
  - 실행 파일 형태로 보조기억장치에 저장됨
- 프로세스(process)
  - 메모리에 적재되어 실행 중인 프로그램
- 프로세스의 구성
  - 메모리 사용
    - 텍스트 섹션: 프로그램 코드
    - 데이터 섹션: 전역 변수 저장
    - 스택(stack): 지역변수, 함수인수, 복귀주소 등 저장 (임시데이터)
    - 힙(heap): 실행시간 동안 동적 할당하는 메모리 공간
  - 현재 상태
    - 현재의 PC와 레지스터 값, 프로세스 상태 등 포함



프로그램과 프로세스

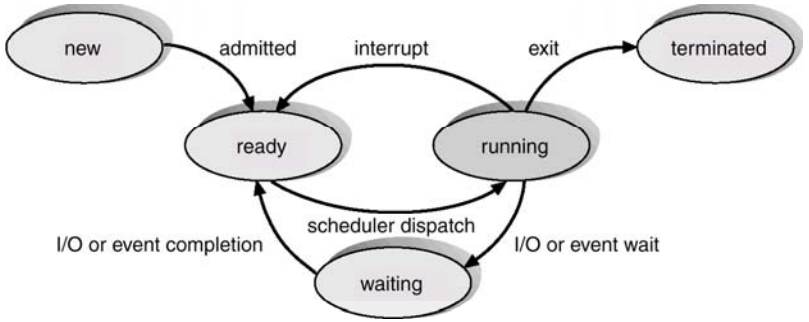
연세대학교

- 프로그램과 프로세스
  - 프로그램: 수동적 개체
  - 프로세스: 능동적 개체
- 여러 프로세스가 같은 프로그램을 실행할 수 있음
  - 텍스트 섹션은 공유함
  - 데이터, 스택 섹션은 별도로 사용함



프로세스의 상태

연세대학교

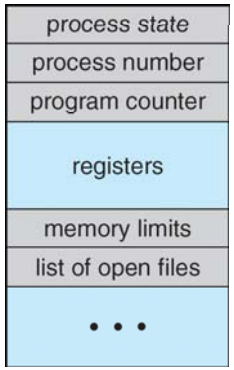


- 생성(new): 프로세스가 생성 중임
- 실행(running): 프로세스의 명령어들이 CPU에서 실행 중임
- 대기(waiting): 프로세스가 어떤 사건을 기다림
- 준비(ready): 실행할 수 있는 상태로서 CPU를 할당 받기를 기다림
- 종료(terminated): 실행이 종료됨

프로세스 제어 블록(PCB)

연세대학교

- 프로세스 제어 블록(Process Control Block: PCB)
  - 운영체제에서 각 프로세스를 관리하기 위해 유지되는 자료구조
- PCB에 유지되는 정보
  - 프로세스 상태
  - 프로그램 카운터
  - CPU 레지스터
  - CPU 스케쥴링 정보
  - 메모리 관리 정보
  - 회계 정보:
    - CPU시간, 시간제한 등
  - 입출력 상태 정보:
    - 할당된 입출력장치 목록
    - open 파일 목록 등



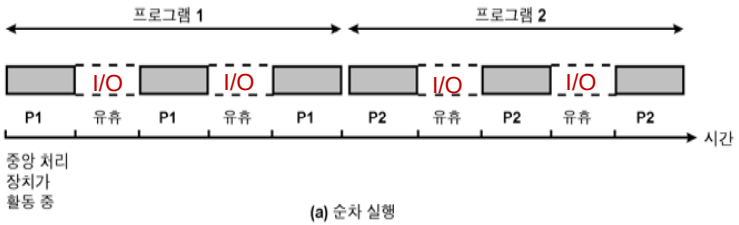
2. 프로세스 스케줄링

연세대학교

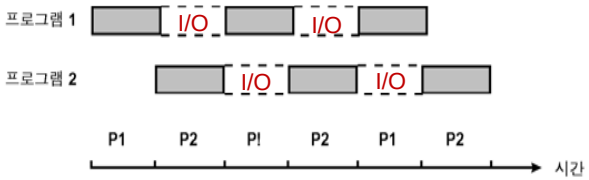
- 다중 프로그래밍
  - 여러 프로세스를 동시에 메모리에 적재하여, 여러 프로세스가 CPU를 번갈아서 사용하도록 하는 것
  - 한 프로세스가 입출력을 하여 CPU를 내어놓으면 다른 프로세스가 CPU를 사용함 (다음 슬라이드 참조)
  - CPU의 이용률을 극대화 하기 위함.
- 시분할 시스템
  - CPU를 시간적으로 분할하여 각 프로세스가 번갈아서 사용할 수 있도록 하는 것.
  - 사용자가 대화형으로 작업을 원활하게 할 수 있도록 하기 위함
- 다중 프로그램, 시분할 시스템은 CPU 스케줄링이 필요함
  - CPU 스케쥴링 : 다음에 CPU를 사용할 프로세스를 선택하는 것
    - ready 상태의 프로세스들 중에서 선택함

다중 프로그래밍에서의 실행

연세대학교



(a) 순차 실행



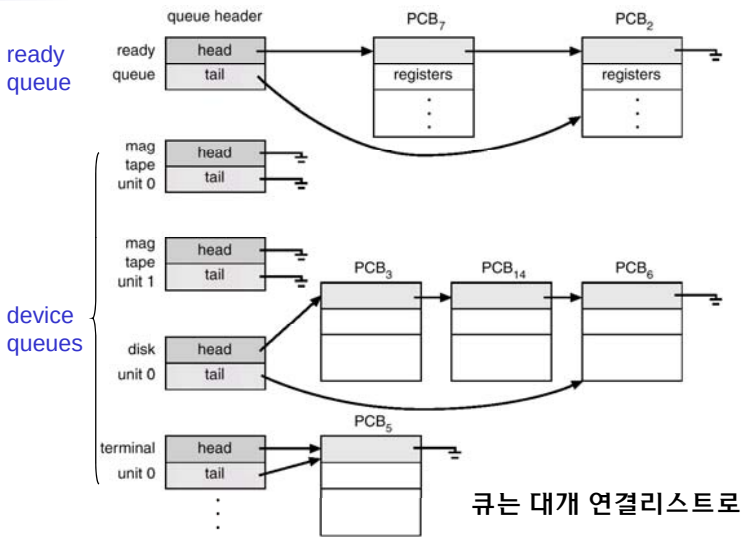
(b) 다중 프로그래밍에서의 실행

스케줄링 큐

연세대학교

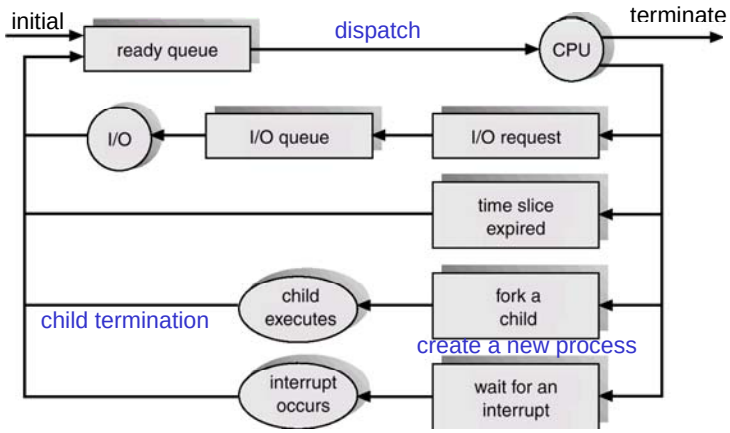
- 프로세스 스케줄링 큐
  - 작업 큐: 전체 프로세스 집합
  - 준비(ready) 큐: 실행되기를 기다리는 프로세스 집합
  - 장치(device) 큐: 입출력 장치를 기다리는 프로세스 집합
    - 각 장치마다 큐가 존재함
  - 대기(wait) 큐: 사건발생을 기다리는 프로세스 집합
    - 각 사건마다 큐가 존재함
- CPU 스케줄링은 ready 큐에 있는 프로세스 집합에서 프로세스를 선택함

여러 가지 스케줄링 큐

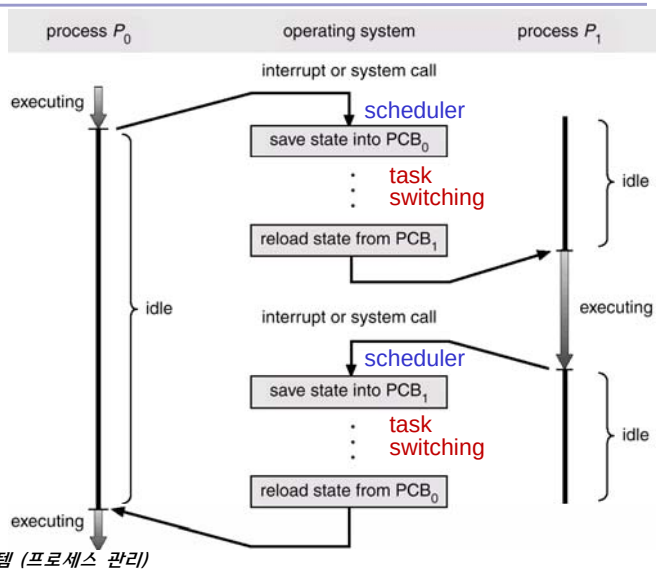


큐는 대개 연결리스트로 구현

프로세스 실행 중 발생할 수 있는 사건



CPU의 프로세스 전환 (context 전환)



컨텍스트 전환

- 프로세스의 컨텍스트(context)
  - CPU 레지스터 값, 메모리 관리 정보 등 프로그램 실행 동안 CPU 내에 서 유지되는 정보
- Context 전환
  - 스케줄러에 의해서 다른 프로세스로 스위칭 되면 현재 프로세스의 context를 PCB에 보관한 다음에
  - 새 프로세스의 context를 PCB에서 CPU로 적재한다.
- Context 전환 시간은 운영체제의 overhead이다.
  - 하드웨어 지원을 통하여 context 전환 시간을 빠르게 할 수 있다.
  - (예) 여러 개의 레지스터 집합, context 전환용 특수 명령어
    쓰레드(thread) 사용

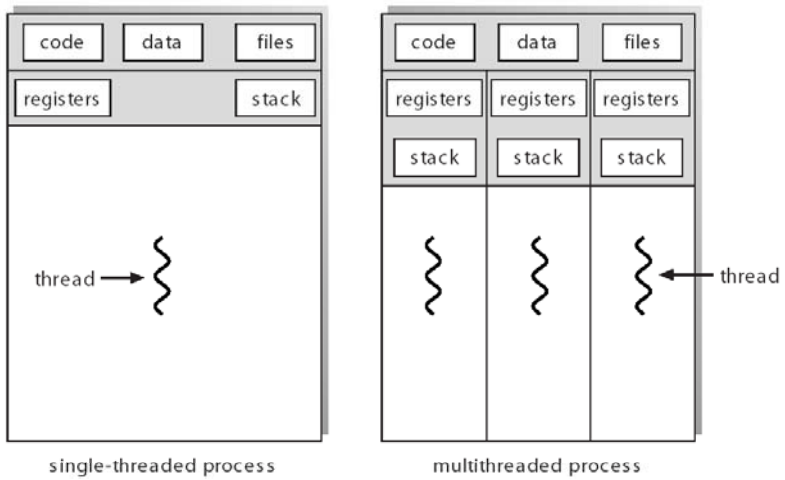
3. 쓰레드(Thread)

연세대학교

- 쓰레드(thread)
  - CPU의 기본 작업 단위
  - 하나의 프로세스는 여러 개의 thread로 구성할 수 있음
    → 다중 쓰레드 프로세스
- 다중 프로세스와 다중 쓰레드
  - 다중 프로세스
    - 프로세스들은 독립된 주소공간을 갖는다.
  - 다중 쓰레드
    - 한 프로세스에서 병행 수행되는 쓰레드들은 주소공간을 공유 → 코드, 데이터, 입출력 자원 등을 공유
    - 쓰레드 ID, program counter(PC), 레지스터 저장 공간과 스택 공간은 개별적으로 보유함

단일 쓰레드와 다중 쓰레드 프로세스

연세대학교



쓰레드와 프로세스

연세대학교

- 쓰레드와 프로세스
  - 쓰레드 → 경량(lightweight) 프로세스
  - (전통적) 프로세스 → 중량(heavyweight) 프로세스
- 전통적인 프로세스는 단일 쓰레드를 갖는 프로세스로 생각할 수 있다.
- 한 응용 프로그램이 여러 가지 일을 동시에 수행하는 경우에
  - 다중 프로세스 방식:
    - 프로세스 생성 비용이 크고 자원 공유가 어려움
  - 다중 쓰레드 방식
    - 쓰레드 생성 비용이 작고, 자원 공유가 쉽다.

4. CPU 스케줄링

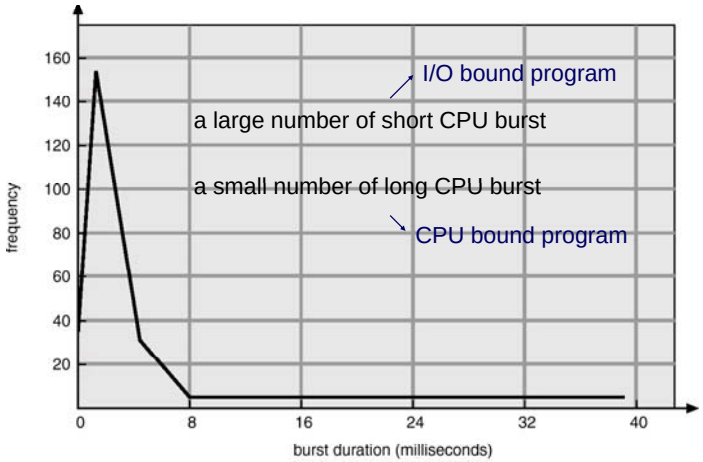
연세대학교

- 기본 개념
  - CPU 사용률 극대화
- CPU와 I/O burst 사이클
  - 프로세스의 실행은 CPU 실행과 I/O 대기의 두 과정을 반복함

CPU burst I/O burst

- CPU burst의 분포
  - 계산 중심의 적은 수의 프로세스가 긴 CPU burst를 가짐
  - 입출력 중심의 많은 수의 프로세스는 짧은 CPU burst를 가짐
  - CPU 성능이 발달할 수록 입출력 중심의 프로세스가 되는 경향
  - (다음 장 그림 참고)

CPU-burst 시간의 분포



CPU 스케줄러

- CPU 스케줄러
  - 준비 큐에 있는 프로세스들 중 하나를 선택하여 선택된 프로세스에게 CPU를 할당함
- 스케줄링이 발생할 수 있는 경우
  - 프로세스가 다음과 같이 전환될 때
    1. 실행 상태 → 대기 상태로 전환 (필수)
    2. 실행 상태 → 준비 상태로 전환 (선택)
    3. 대기 상태 → 준비 상태로 전환 (선택)
    4. → 종료 상태로 전환 (필수)

선점과 비선점 스케줄링

- 비선점 스케줄링 (Nonpreemptive Scheduling)
  - 1,4의 경우에만 스케줄링
  - 현재 프로세스가 I/O 대기 또는 종료로 인해서 CPU를 내어놓는 경우에만 스케줄링 가능
  - 특정 프로세스가 CPU를 오랫동안 독점할 수도 있음
- 선점 스케줄링 (Preemptive Scheduling)
  - 2,3의 경우에도 스케줄링 가능
  - 어떤 조건이 만족되면 현재 실행 중인 프로세스의 의사와 관계없이 현재 프로세스의 실행을 중단하고 CPU를 다른 프로세스에게 할당함
  - 대부분의 운영체제에서 채택
  - 여러 프로세스가 데이터를 공유하는 경우에 문제가 발생할 수 있으며 이 문제를 해결하는 방법이 필요함 (경쟁 조건)

스케줄링 알고리즘

- FCFS(First-Come First-Served) 스케줄링
  - 요청한 순서로 스케줄링
  - CPU burst가 짧은 프로세스들이 CPU burst가 긴 프로세스 뒤에 있는 경우에 짧은 프로세스들은 대기 시간이 길어짐
- SJF(Shortest-Job-First) 스케줄링
  - 다음 CPU-burst가 가장 짧은 프로세스에게 스케줄링
  - 문제점: 프로세스의 다음 CPU-burst 시간을 사전에 알 수 없다.
  - 해결방법: 과거의 CPU-burst 시간들을 사용하여 다음 CPU-burst 시간을 예측한다.
  - SJF 스케줄링 방법은 최소 평균 대기 시간을 제공함
  - 선점 SJF, 비선점 SJF 스케줄링 가능

스케줄링 알고리즘(2)

연세대학교

- 우선순위 스케줄링
  - 우선순위가 가장 높은 프로세스에게 스케줄링
  - SJF는 우선순위 스케줄링의 한 예
    - next CPU-burst가 작을 수록 우선순위가 높음
  - 우선 순위는 내부요인(운영체제가 판단)와 외부 요인(관리자/사용자가 지정)에 의해서 정해짐
  - 선점/비선점 우선순위 스케줄링 가능
  - 기아(starvation) 현상이 발생할 수 있음 → Aging 기법을 사용하여 해결 (대기 중인 프로세스의 우선순위를 점진적으로 증가시킴)
- Round-Robin(RR) 스케줄링
  - 일정 시간(time quanum)이 경과할 때마다 현재 프로세스 중단하고 다음 프로세스를 스케줄링 → 선점 스케줄링
  - 시분할 시스템에서 사용, time quantum 값 선택이 중요
  - 응답시간이 작다

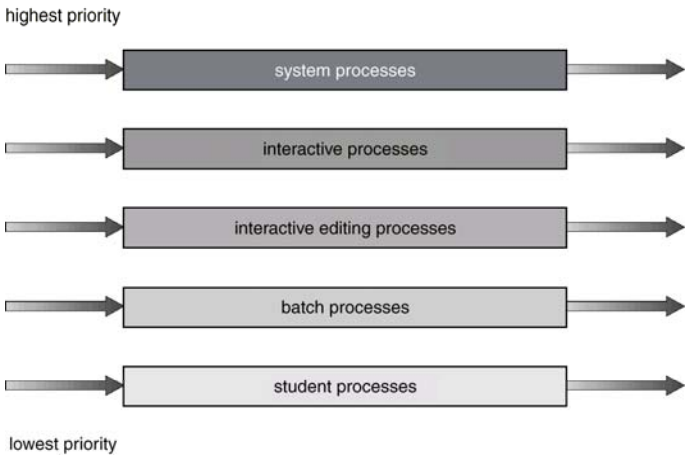
스케줄링 알고리즘(3)

연세대학교

- 다중 레벨 큐 스케줄링
  - 여러 개의 준비 큐를 사용
  - 프로세스 특성에 따라서 특정 큐를 할당 (다음 장 그림 참조)
  - 각 큐마다 독자적인 스케줄링 알고리즘 사용 (RR, FCFS 등)
  - 큐 간에 우선 순위 부여
    - 상위 순위 큐가 하위 순위 큐에 우선함, 또는
    - 상위와 하위 큐 간에 CPU 시간 사용 비율 지정 (예) 8 : 2

스케줄링 알고리즘(4)

연세대학교



스케줄링 알고리즘(5)

연세대학교

- 다중 레벨 피드백 큐 스케줄링
  - 다중 레벨 큐 스케줄링 방법에서 프로세스가 큐 간에 이동할 수 있도록 수정한 방법
  - 대화식 프로세스는 상위 큐에 할당되고 , 계산중심 프로세스는 하위 큐에 할당되도록 프로세스를 큐 간에 이동시킴
    - 상위 큐는 RR 스케줄링 방법 사용
    - 하위 큐는 FCFS 스케줄링 방법 사용