리눅스 프로그래밍

프로세스와 스레드

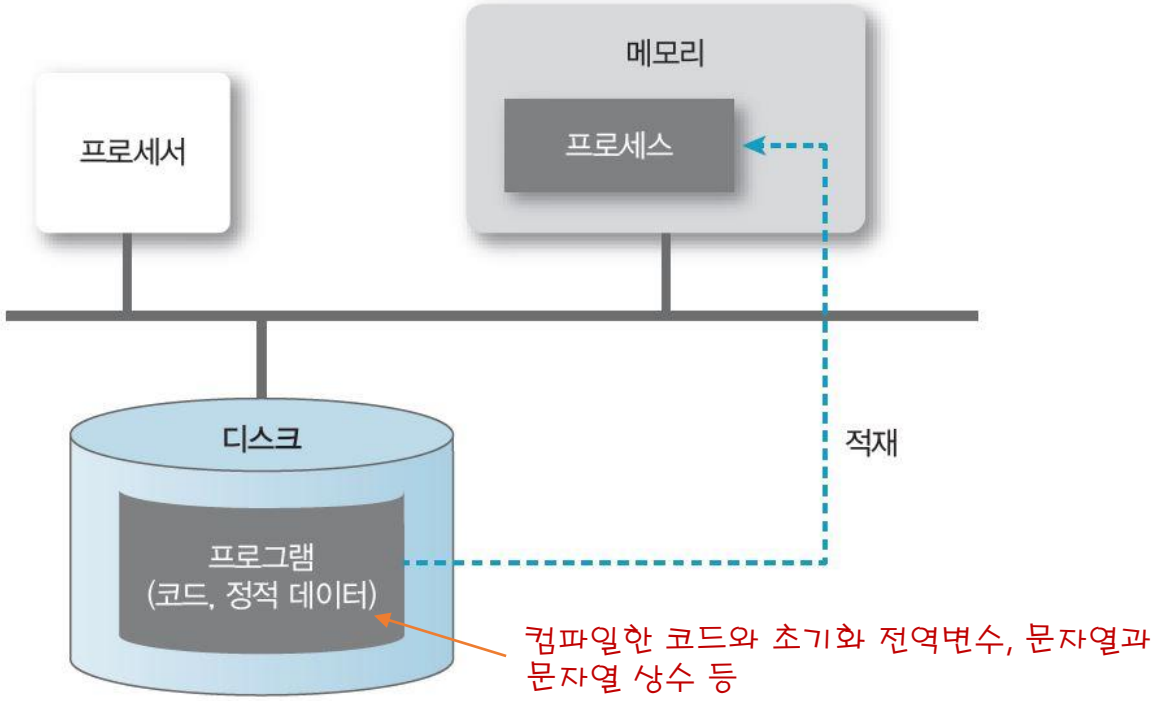
인공지능소프트웨어과
이혜정 교수

* 본 교안의 일부 그림과 텍스트는 '운영체제 – 그림으로 배우는 구조와 원리' (개정3판, 한빛미디어)으로 발췌하였음

목차

프로세스process의 정의

- **프로세스process**
  - 실행 중인 프로그램 가장 일반적인 정의
  - 프로세서에 할당하여 실행할 수 있는 개체 디스패치dispatch가 가능한 대상
- **프로그램과 프로세스**
  - 프로그램이 메모리로 적재되면 프로세스가 됨



프로세스의 일반적인 메모리 구조

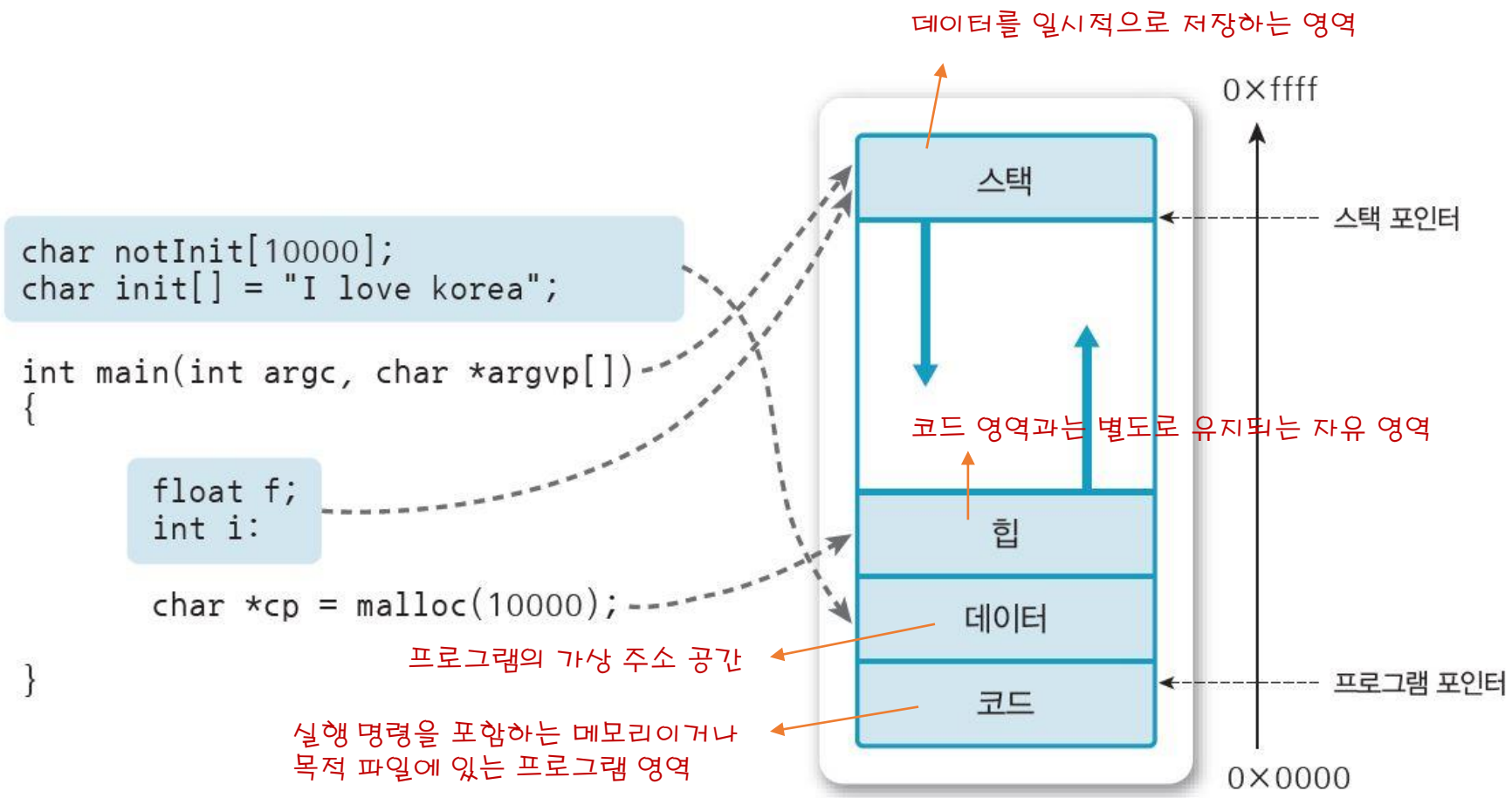


그림 3-2 프로세스의 일반적인 메모리 구조(사용자 관점의 프로세스)

프로세스의 상태 변화

준비 상태에서 대기하고 있는 프로세스 중 하나가 스케줄링되어 CPU를 할당받아 실행 상태로 전이되는 과정

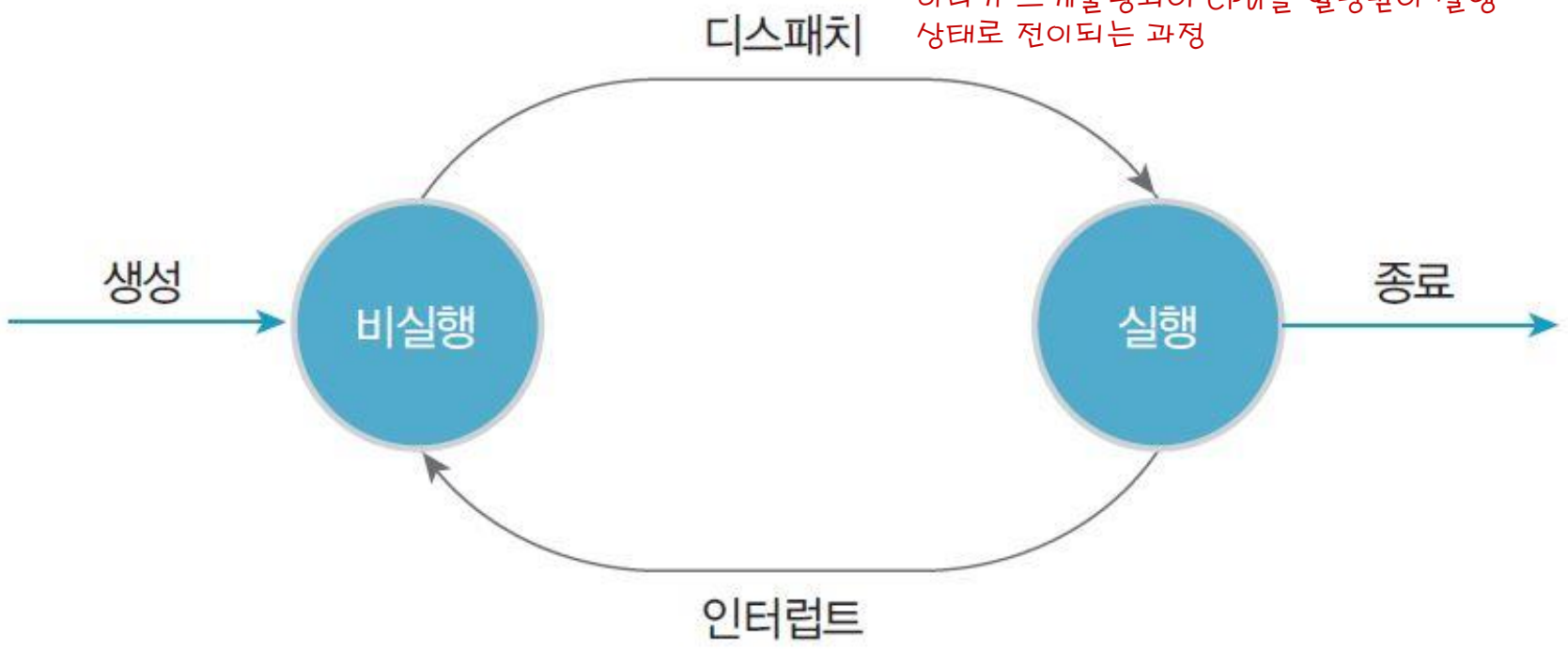
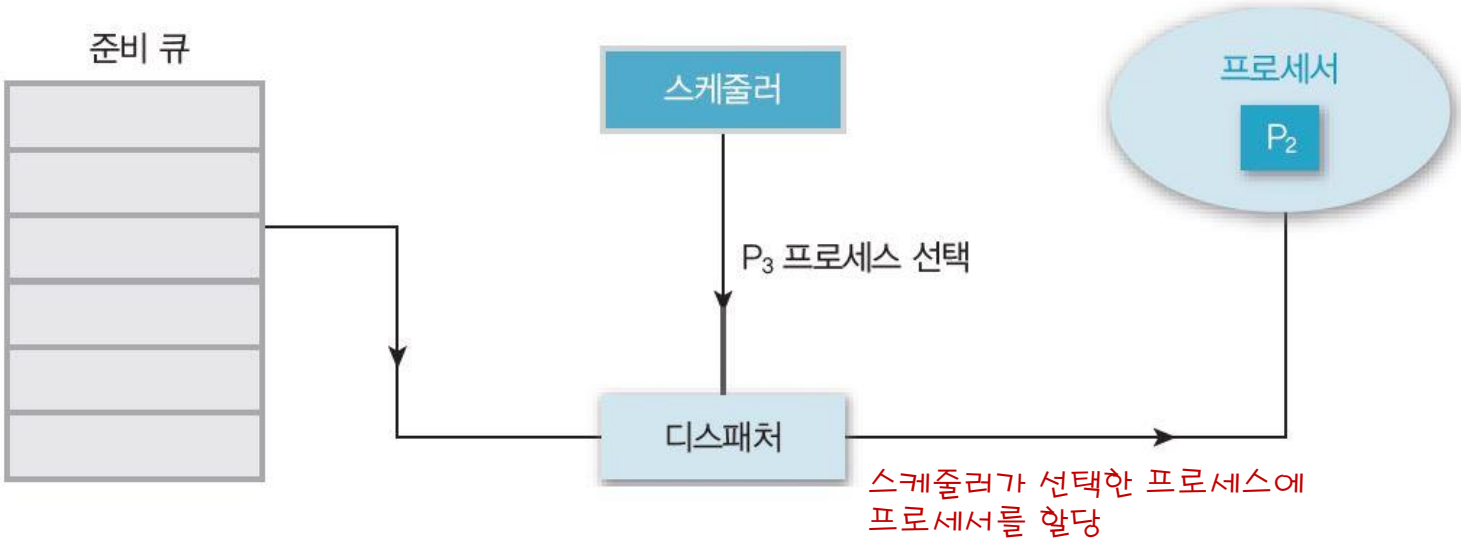


그림 3-4 프로세스의 상태

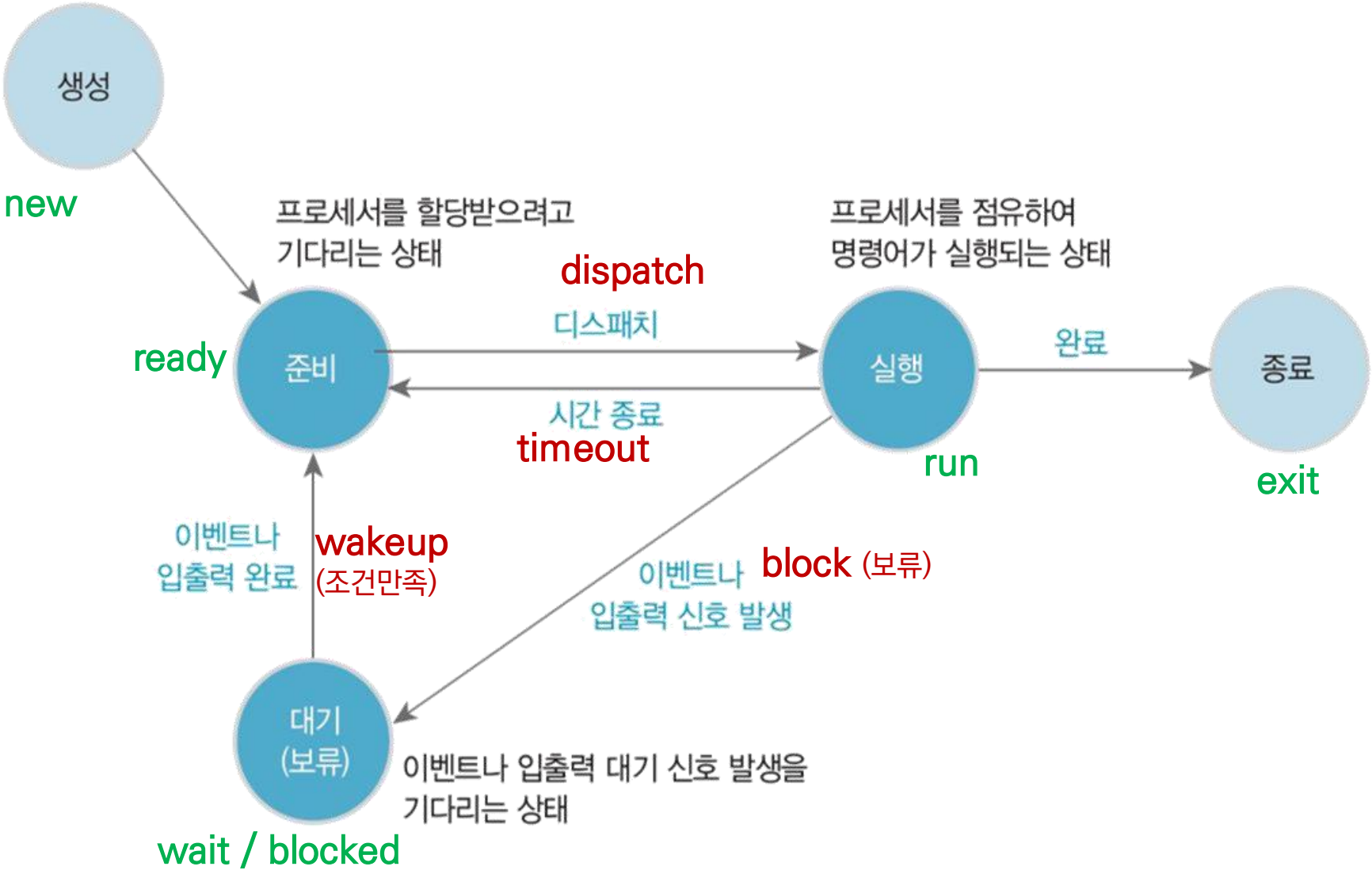
프로세스의 상태 변화

- **프로세스의 상태 변화는 운영체제가 프로세서 스케줄러 이용하여 관리**
  - 프로세스 스케줄러
    - 선정한 작업의 상태를 변화시키며 프로세스의 생성에서 종료까지 과정 수행



- **프로세스 생성 시 필요한 세부 작업 순서**
  1. 새로운 프로세스에 프로세스 식별자 할당
  2. 프로세스의 모든 구성 요소를 포함할 수 있는 주소 공간과 프로세스 제어 블록 공간 할당
  3. 프로세스 제어 블록 초기화
     - 프로세스 상태, 프로그램 카운터 등 초기화, 자원 요청, 프로세스 제어 정보(우선순위) 등
  4. 준비 큐에 삽입

프로세스의 상태 변화


생성
new
프로세서를 할당받으려고
기다리는 상태
dispatch
디스패치
프로세서를 점유하여
명령어가 실행되는 상태
ready
준비
실행
완료
종료
시간 종료
timeout
run
exit
이벤트나
입출력 완료
wakeup
(조건만족)
이벤트나
입출력 신호 발생
block (보류)
대기
(보류)
이벤트나 입출력 대기 신호 발생을
기다리는 상태
wait / blocked

프로세스의 문맥교환 Context Switching

- **실행 중인 프로세스의 제어를 다른 프로세스에 넘겨 실행 상태가 되도록 하는 것**
  - 프로세스 문맥 교환이 일어나면 프로세서의 레지스터에 있던 내용 저장

- **문맥 교환 발생 상태 변화**
  - 준비 → 실행
  - 실행 → 준비
  - 실행 → 대기

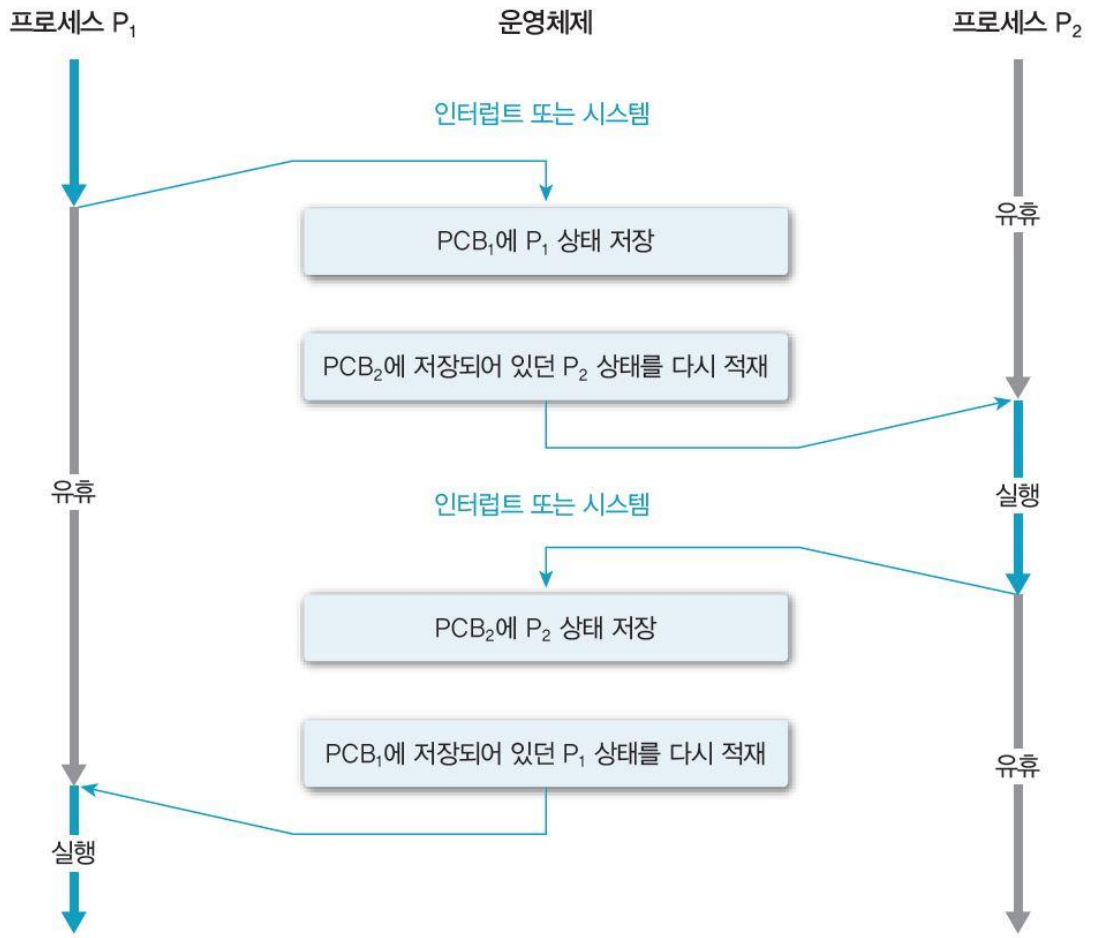


그림 3-8 프로세스 문맥 교환 예

프로세스의 상태 정보

- **프로세스 제어 블록, PCB** Process Control Block
  - 운영체제가 프로세스 제어 시 필요한 프로세스 상태 정보 저장
  - 프로세스가 생성되면 메모리에 PCB 생성, 프로세스가 실행 종료하면 해당 PCB도 삭제

프로세스 식별자	각 프로세스의 고유 식별자(숫자, 색인 항목)
프로세스 상태	생성, 준비, 실행, 대기, 중단 등 상태 표시
프로그램 카운터	프로세스를 실행하는 다음 명령의 주소 표시
레지스터 저장 영역	누산기, 인덱스 레지스터, 스택 포인터, 범용 레지스터, 조건 코드 등 정보로, 컴퓨터 구조에 따라 수나 형태가 다르다. 인터럽트가 발생하면 프로그램 카운터와 함께 저장하여 재실행할 때 원래대로 복귀할 수 있게 한다.
프로세서 스케줄링 정보	프로세스의 우선순위, 스케줄링 큐의 포인터, 기타 스케줄 매개변수
계정 정보	프로세서 사용 시간, 실제 사용 시간, 사용 상한 시간, 계정 번호, 작업이나 프로세스 번호 등
입출력 상태 정보 메모리 관리 정보 ⋮	• 특별한 입출력 요구 프로세스에 할당된 입출력장치, 열린 파일 리스트 등 • 운영체제가 사용하는 메모리 시스템에 따른 상한 · 하한 레지스터 (경계 레지스터), 페이지 테이블이나 세그먼트 테이블 값 등

스레드 thread 의 개념

- **프로세스의 특성인 자원과 제어에서 제어만 분리한 실행 단위**
  - 프로세스의 직접 실행 정보를 제외한 나머지 프로세스 관리 정보 공유
  - 프로세스 하나는 스레드 한 개 이상으로 나눌 수 있음
  - 다른 프로시저 호출, 다른 실행 기록 (별도 스택 필요)

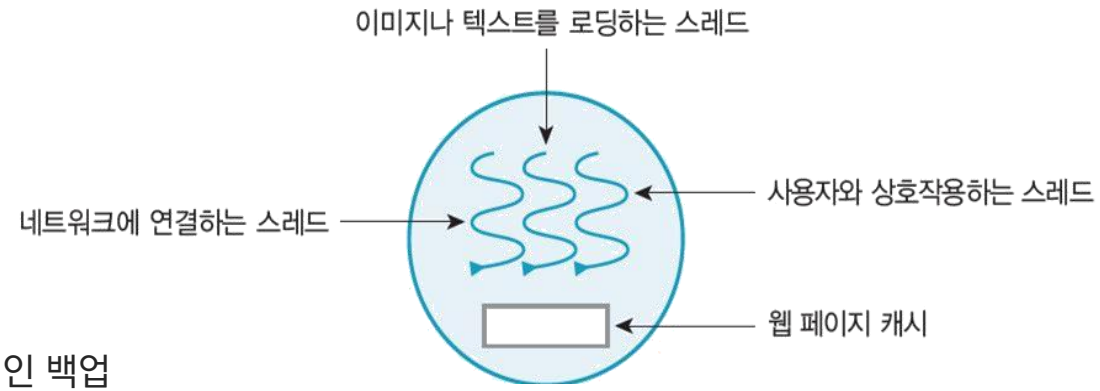
(a) 단일 스레드의 프로세스 (b) 다중 스레드의 프로세스

- **스레드 병렬 수행**
  - 프로세스 하나에 포함된 스레드들은 공동의 목적 달성을 위해 병렬 수행
  - 여러 개의 프로세서에서 프로세스가 하나인 프로그램의 여러 부분을 동시 실행

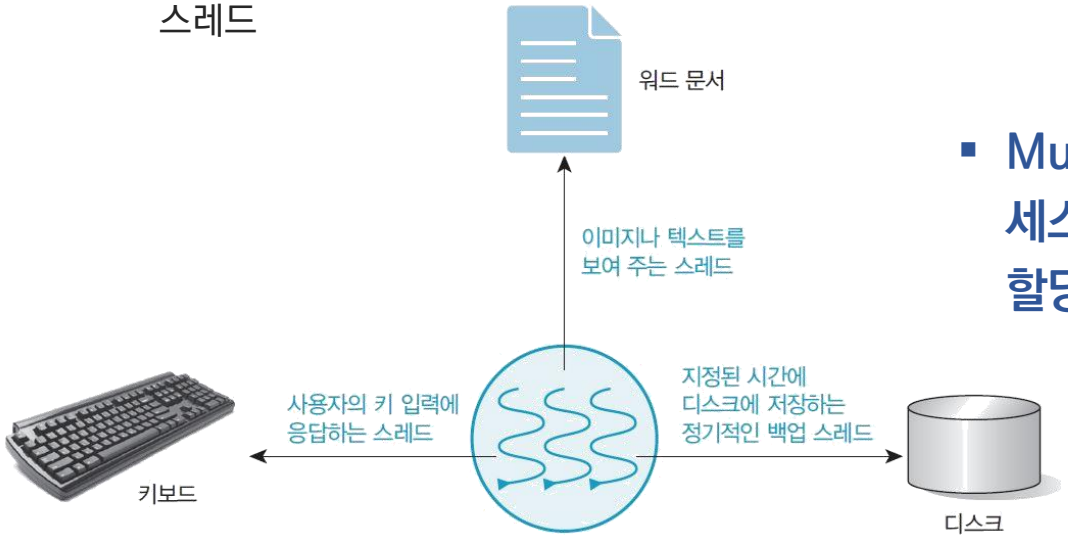
스레드의 사용

- 이전 명령 완료 전 다음 명령 신속하게 준비
  - Thread 1 : 사용자 데이터를 읽어 메뉴 표시
  - Thread 2 : 사용자 명령 실행

- 실행 중 스레드가 대기 상태가 되면 제어를 다른 스레드로 옮기는 상태 변화 이용
  - 웹브라우저



- 비동기적 요소 구현
  - 워드 편집기
    - 지정된 시간에 디스크에 저장하는 정기적인 백업 스레드



- Multi-Processing 시스템에서 하나의 프로세스 내 여러 스레드를 각각의 Processor에 할당하여 병렬로 처리

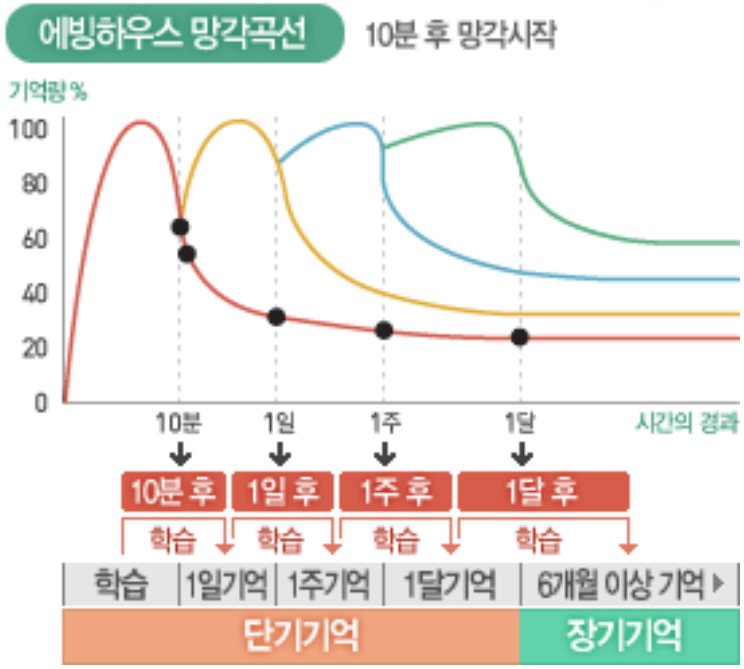
스레드의 상태 변화

- **프로세스처럼 준비, 실행, 대기(보류), 종료 상태가 있음**
  - 프로세스 생성 시 스레드도 함께 생성
    - 자원 초기화 필요 없음 (프로세스의 자원 공유 – 스택과 레지스터 제공)
      → 오버헤드가 적음
  - 한 프로세스 내 스레드는 순차적으로 실행
    - 스레드 한 개가 대기 상태로 변할 때 전체 프로세스를 대기 상태로 바꾸지 않음
      → 한 스레드가 대기 상태가 되면 다른 스레드를 실행할 수 있음

스레드 사용의 이점

- **사용자 응답성 증가**
- **다중 처리(Multi-Processing)로 성능과 효율 향상**
- **프로세스의 자원과 메모리 공유 가능**
  - 한 스레드가 전역 변수 변경 시, 다른 스레드가 접근하여 변경 결과 확인 가능
  - 한 스레드가 읽기 전용으로 파일을 열면, 다른 스레드도 이 파일을 읽을 수 있음
- **경제성 좋음**
  - 프로세스보다 생성, 종료, 문맥 교환이 빠름

망각 곡선 되살리기


에빙하우스 망각곡선
10분 후 망각시작
기억량 %
100
80
60
40
20
0
10분
1일
1주
1달
시간의 경과
10분 후
1일 후
1주 후
1달 후
학습
학습
학습
학습
학습
1일기억
1주기억
1달기억
6개월 이상 기억
단기기억
장기기억