스레드 주소공간과 컨텍스트

유튜브: https://youtu.be/pse7AOyQH7o

사이버보안트랙 1971362 이준희

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

<목차>

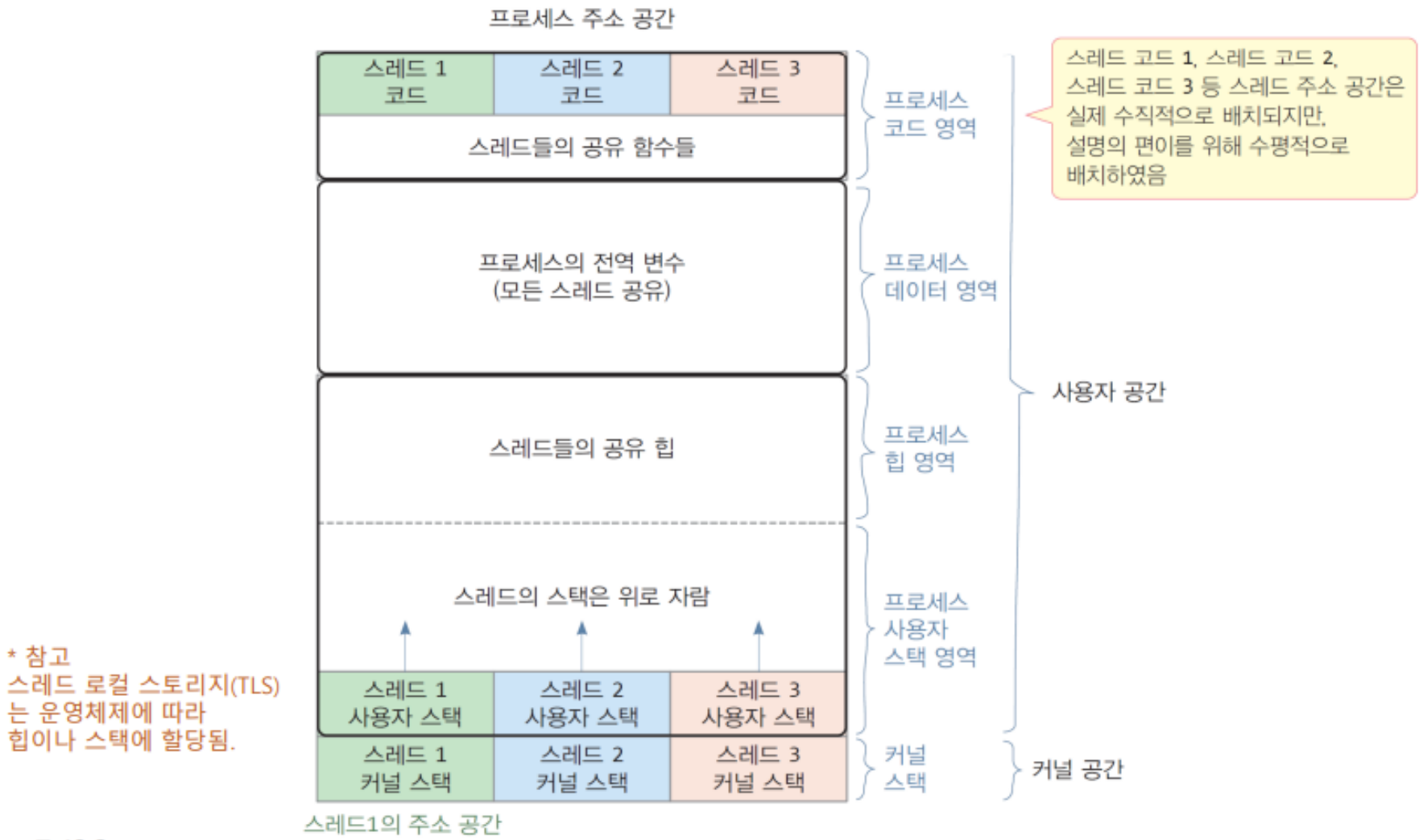
1. 스레드 주소 공간

- 스레드가 실행 중에 사용하는 메모리 공간
  - 스레드의 코드, 데이터, 힙, 스택 영역

- 프로세스의 주소 공간 내에 형성
- 스레드 주소 공간은
  - 프로세스 주소 공간 내에서 ‘사적 공간’과 ‘공유 공간’으로 구분

- 스레드 사적 공간
  - 스레드 스택
  - 스레드 로컬 스토리지 (TLS, Thread local storage)
- 스레드 사이의 공유 공간
  - 프로세스의 코드(스레드 코드 포함)
  - 프로세스의 데이터 공간
  - 프로세스의 힙 영역

스레드 주소 공간 사례



* 참고
스레드 로컬 스토리지(TLS)는 운영체제에 따라 힙이나 스택에 할당됨.

5/23

※ 스레드의 주소 공간은 프로세스 주소 공간 내에 존재한다.

스레드 주소 공간에 대한 설명(1)

- 스레드 코드 영역
  - 스레드가 실행할 작업의 함수, 프로세스의 코드 영역에 있음
  - 스레드는 프로세스의 코드 영역에 있는 다른 모든 함수 호출 가능
- 스레드 데이터 영역
  - 스레드가 사용할 수 있는 데이터 공간
  - 프로세스에 선언된 모든 전역 변수들 - 프로세스의 데이터 영역
  - 개별 스레드의 전용 변수 공간(스레드 로컬 스토리지)

스레드 주소 공간에 대한 설명(1)

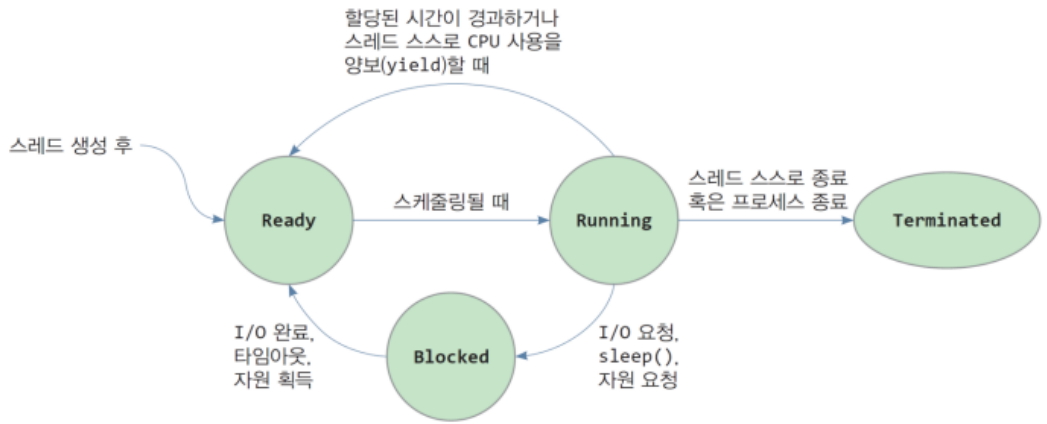
- 스레드 힙
  - 모든 스레드가 동적 할당받는 공간, 프로세스 힙 공간을 공유하여 사용
  - 스레드에서 malloc()를 호출하면 프로세스의 힙 공간에서 메모리 할당
- 스레드 스택
  - 스레드가 생성될 때,
    - 프로세스에게 할당된 스택에서 사용자 스택 할당
    - 커널 공간에 스레드마다 커널 스택 할당
  - 스레드가 시스템 호출로 커널에 진입할 때, 커널 스택 활용
  - 스레드 종료 시, 스레드가 할당 받은 사용자 스택과 커널 스택 반환

스레드 주소 공간에 대한 설명(2)

- 스레드 로컬 스토리지(TLS, Thread Local Storage)
  - 스레드마다 안전하게 다루고자 하는 데이터를 저장하기 위한 별도의 영역
    - => 프로세스의 데이터 영역은 모든 스레드의 공용 공간이므로
  - 스레드가 자신만 사용할 변수들을 선언할 수 있는 영역
  - 생성되는 영역 : 운영체제마다 다름. 대체로 힙이나 스택에 할당
  - 프로그램에서 할당받는 방법
    - 프로그래밍 언어마다 다름
- C 언어에서 스레드 로컬 스토리지 할당 사례
  - 스레드 로컬 스토리지가 선언되면, 스레드가 생성될 때마다 모든 스레드에 동일 한 크기의 공간 할당
  - ex) static __thread int sum = 5; // 스레드 로컬 스토리지에 할당될 변수 sum 선언
    - => 스레드가 생길 때마다, 각 스레드에 sum 변수 크기의 스레드 로컬 스토리지가 할당되고 초기값이 5로 설정
  - 스레드가 종료하면 스레드의 로컬 스토리지 사라짐

2. 스레드 상태

- 스레드 일생
  - 스레드는 생성, 실행, 중단, 실행, 소멸의 여러 상태를 거치면서 실행
  - 스레드 상태는 TCB에 저장
  - 스레드 상태
    - 준비 상태(Ready) - 스레드가 스케줄 되기를 기다리는 상태
    - 실행 상태(Running) - 스레드가 CPU에 의해 실행 중인 상태
    - 대기 상태(Blocked) - 스레드가 입출력을 요청하거나 sleep() 같은 시스템 호출로 인해 중단된 상태
    - 종료 상태(Terminated) - 스레드가 종료한 상태

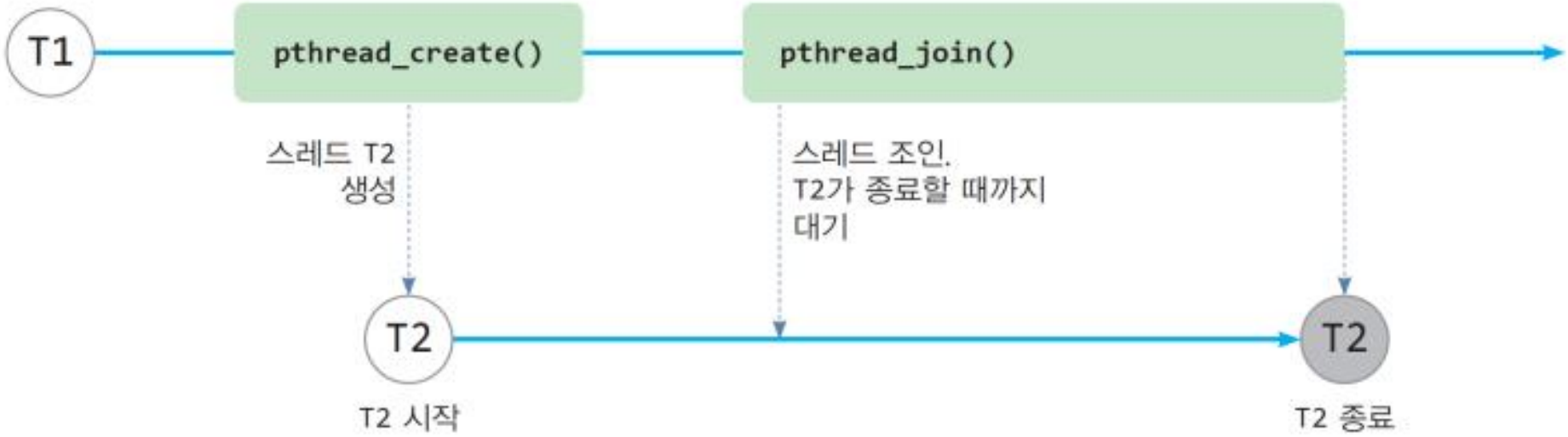


3. 스레드 운용(operation)

- 응용프로그램이 스레드에 대해 할 수 있는 운용의 종류
  - 1) 스레드 생성
    - 프로세스가 생성되면 운영체제에 의해 자동으로 main 스레드 생성
    - 스레드는 시스템 호출이나 라이브러리 함수를 호출하여 새 스레드 생성 가능
  - 2) 스레드 종료
    - 프로세스 종료와 스레드 종료의 구분 필요
    - 프로세스 종료
      - => 프로세스에 속한 아무 스레드가 exit() 시스템 호출을 부르면 프로세스 종료(모든 스레드 종료)
      - 메인 스레드의 종료(C 프로그램에서 main() 함수 종료) – 모든 스레드가 함께 종료
      - 모든 스레드가 종료하면 프로세스 종료
  - 3) 스레드 조인
    - 스레드가 다른 스레드가 종료할 때까지 대기
      - => 주로 부모 스레드가 자식 스레드의 종료 대기
  - 4) 스레드 양보
    - 스레드가 자발적으로 yield()와 같은 함수 호출을 통해 스스로 실행을 중단하고 다 른 스레드를 스케줄하도록 요청

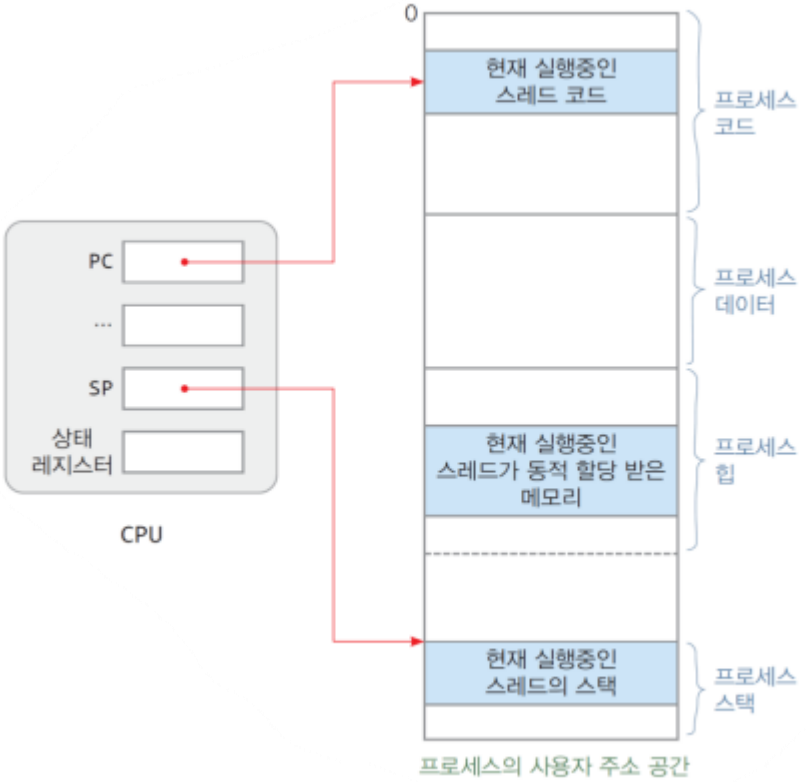
스레드 조인

- 스레드가 다른 스레드의 종료를 기다리는 행위



4. 스레드 컨텍스트

- 스레드 컨텍스트
  - 스레드가 현재 실행중인 일체의 상황
  - CPU 레지스터 값들
- 스레드 컨텍스트 정보
  - PC 레지스터
    - 실행 중인 코드 주소
  - SP 레지스터
    - 실행 중인 함수의 스택 주소
  - 상태 레지스터
    - 현재 CPU의 상태 정보
  - CPU에 기타 수십 개의 레지스터 있음
    - 데이터 레지스터 등
  - 컨텍스트 스위치 될 때 TCB에 저장
  - 스레드 컨텍스트를 저장해 두었다가 CPU에 복귀시키면, 이전에 중단된 상태에서 실행할 수 있음



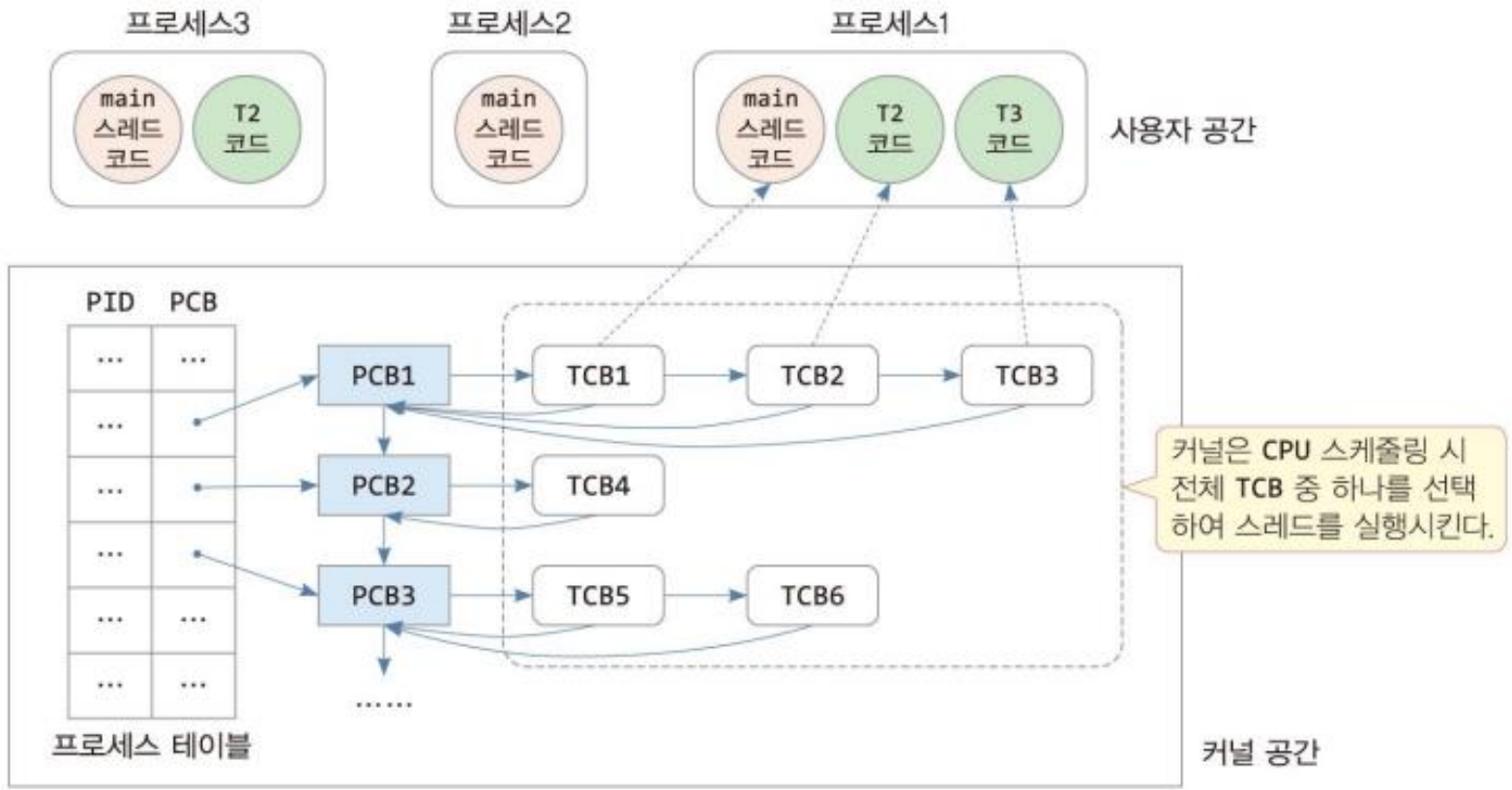
스레드 제어 블록 (Thread Control Block)

- 스레드 제어 블록, TCB(Thread Control Block)
  - 스레드를 실행 단위로 다루기 위해 스레드에 관한 정보를 담은 구조체
    - 스레드 엔터티(thread entity), 스케줄링 엔터티(scheduling entity)라고도 불림
  - 커널 영역에 만들어지고, 커널에 의해 관리
    - 스레드가 생성될 때 커널에 의해 만들어지고, 스레드가 소멸되면 사라짐

구분	요소	설명
스레드 정보	tid	스레드 ID. 스레드가 생성될 때 부여된 고유 번호
	state	스레드의 상태 정보. 실행(Running), 준비(Ready), 블록(Blocked), 종료(Terminated)
컨텍스트	PC	CPU의 PC 레지스터 값. 스레드가 스케줄될 때 실행할 명령의 주소
	SP	CPU의 SP 레지스터 값. 스레드 스택의 톱 주소
	다른 레지스터들	스레드가 중지될 때의 여러 CPU 레지스터 값들
스케줄링	우선순위	스케줄링 우선순위
	CPU 사용 시간	스레드가 생성된 이후 CPU 사용 시간
관리를 위한 포인터들	PCB 주소	스레드가 속한 프로세스 제어 블록(PCB)에 대한 주소
	다른 TCB에 대한 주소	프로세스 내 다른 TCB들을 연결하기 위한 링크
	블록 리스트/준비 리스트 등	입출력을 대기하고 있는 스레드들을 연결하는 TCB 링크, 준비 상태에 있는 스레드들을 연결하는 TCB 링크(스레드 스케줄링 시 사용) 등

스레드와 TCB, 그리고 PCB와의 관계

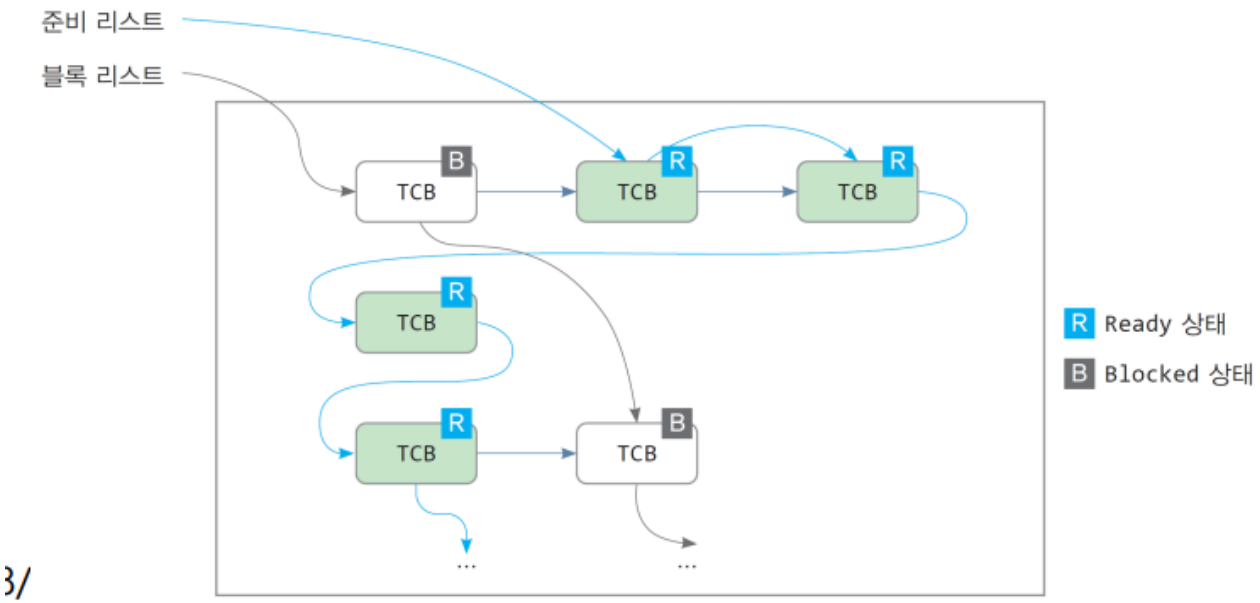
- 프로세스 : 스레드들이 생기고 활동하는 자원의 컨테이너
- TCB들은 링크드 리스트로 연결



3

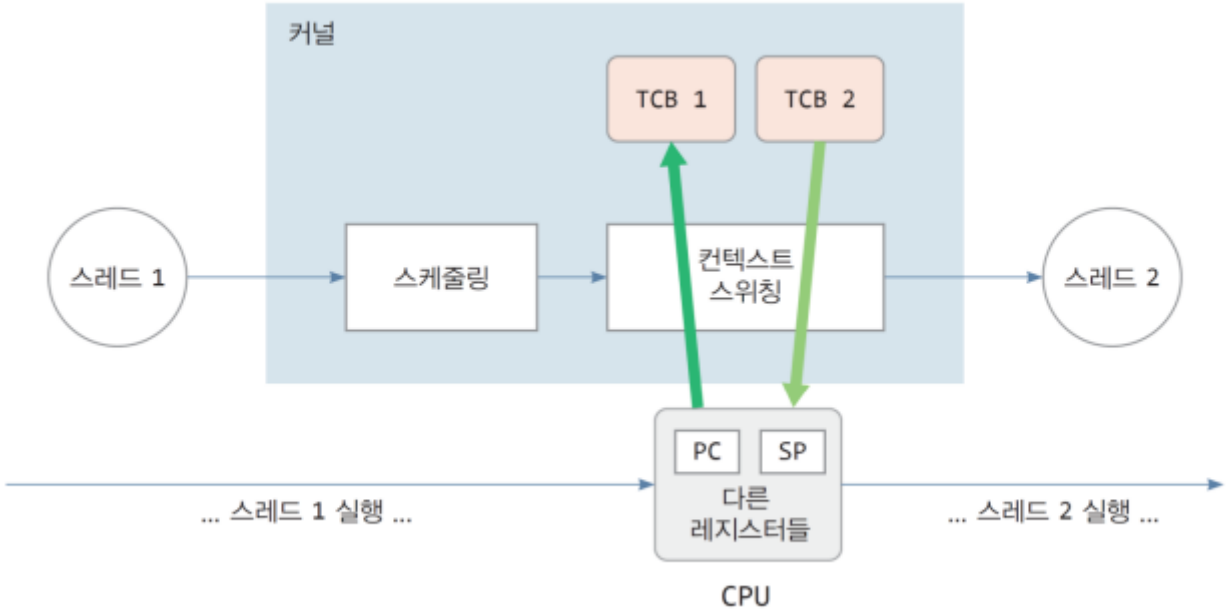
준비 리스트와 블록 리스트

- 준비 리스트
  - 준비 상태에 있는 스레드들의 TCB를 연결하는 링크드 리스트
  - 스레드 스케줄링은 준비 리스트의 TCB들 중 하나 선택
- 블록 리스트
  - 블록 상태에 있는 스레드들의 TCB를 연결하는 링크드 리스트



스레드 컨텍스트 스위칭

- 스레드 컨텍스트 스위칭(스레드 스위칭)
  - 스레드 스케줄링 후,
  - 현재 실행중인 스레드를 중단시키고, 선택된 스레드에게 CPU 할당
    - 현재 CPU 컨텍스트를 TCB에 저장하고,
    - 선택된 스레드의 TCB에서 컨텍스트를 CPU에 적재, CPU는 선택된 스레드 실행



스레드 스위칭이 발생하는 4가지 경우

- 스레드 스위칭이 발생하는 4가지 경우
  - 1. 스레드가 자발적으로 다른 스레드에게 양보
    - yield() 등의 시스템 호출(혹은 라이브러리 호출)을 통해
  - 2. 스레드가 시스템 호출을 실행하여 블록되는 경우
    - read(), sleep(), wait() 등 I/O가 발생하거나 대기할 수 밖에 없는 경우
  - 3. 스레드의 타임 슬라이스(시간 할당량)를 소진한 경우
    - 타이머 인터럽트에 의해 체크되어 진행
  - 4. I/O 장치로부터 인터럽트가 발생한 경우
    - 현재 실행중인 스레드보다 높은 순위의 스레드가 I/O 작업을 끝낸 경우 등

    - 상황에 따라 운영체제에 따라, 이들 4가지 경우 외에도 스레드 스위칭이 일어날 수도 있고, 아닐 수도 있음

스레드 스위칭이 이루어지는 위치

- 스레드 스위칭이 이루어지는 위치는 2가지
  - 1. 스레드가 시스템 호출을 하여, 커널이 시스템 호출을 처리하는 과정에서
  - 2. 인터럽트가 발생하여 인터럽트 서비스 루틴이 실행되는 도중 커널 코드에서

Q & A