22.12.18

📌 프로세스 동기화

- 여러 프로세스에서 접근이 가능한 공유 자원(Shared Resource)에 대한 경쟁 조건(Race Condition)이 발생하는 곳인 임계 영역(Critical Section)에 어떻게 대처할 것인가
- 임계 영역에 담긴 자원의 일관성을 위한 3가지 조건
  - 상호 배제
  - 진행
    - 임계 영역은 계속해서 사용되어야 한다
    - 경직된 동기화(LockStepSynchronization)의 원인이 될 수 있음
  - 한정 대기
    - 임계 영역 진입에는 횟수 한정이 있어야 한다.
    - 무한정 대기 시 Starvation, DeadLock 발생 요인이 될 수 있음

  → 결국 임계 영역에 대한 적절한 Lock이 필요함
- 2가지의 임계 영역에 대한 Lock 구현 방식
  - Mutex(Mutual Exclutsion)
    - 임계 구역에 대한 프로세스 간의 시간이 겹치지 않도록 하는 방식
    - 2개 프로세스 간의 경쟁을 기반으로 함
    - Lock을 소유한 프로세스만이 임계 영역에 접근 가능
    - 프로세스 단위의 동기화 방식
  - Semaphore
    - 시그널 매커니즘(임계 영역 전 wait(), 임계 영역 후 signal())을 통해, 프로세스 간 임계 영역을 사용하도록 한 방식
    - 2개 이상의 프로세스 간 경쟁을 기반으로 함
    - 시스템 범위에서 커널이 소유한 세마포어를 통해 임계 영역에 접근 가능
    - 기다리는 방식 - Busy Waiting(초기) → Block-Wakeup(현재)

📌 PRG

문제 상황

- Post 요청을 한 웹 브라우저에서 새로고침 시, Post 요청이 재발송될 수 있다
- 예시
  - Client - Post /order?bread=2
  - Server - 요청 처리 후 200 ok 반환
  - 이 때 클라이언트는 서버에서 요청 처리되었음을 200 ok를 통해 인지할 수는 있으나, 현재 위치한 url 자체는 /order?bread=2 이다.
  - 따라서 새로고침을 할 경우 다시 Post /order?bread=2 요청을 서버로 송신하게 된다. 중복 요청의 문제가 발생.

해결 방안

- PRG(Post - Redirection - Get)를 사용
- 예시
  - Client - Post /order?bread=2
  - Server - 요청 처리 후 결과 페이지로 이동을 요청하는 302 Found URL 반환 (헤더에 Location 필드 추가)
  - Client는 302 Found를 받고, 헤더로 받은 Location으로 이동. 이 때 GET 요청을 통해 이동한다.
  - 이제 브라우저의 url은 결과 페이지의 url이므로, 새로 고침 시에도 재전송과 같은 문제가 발생하지 않는다.