5단원. 교착상태 (Deadlock)

> 대상: 전공자 · 정보처리기사 취득자 | 목적: 인프라 직무 기술 면접 대비 중요도: ★★★ 최빈출 · ★★ 자주 출제 · ★ 기본 개념

Q1. 데드락 발생의 필요조건 4가지를 설명하세요. ★★★

답안 ① **상호 배제(Mutual Exclusion)**: 자원을 한 번에 한 프로세스만 사용할 수 있습니다. ② **점유와 대기(Hold and Wait)**: 자원을 보유한 채 다른 자원을 기다립니다. ③ **비선점(No Preemption)**: 자원을 강제로 빼앗을 수 없고 자발적 반납만 가능합니다. ④ **순환 대기(Circular Wait)**: 프로세스들이 원형으로 서로의 자원을 기다립니다. 네 조건이 **모두** 성립해야 데드락이 발생하므로, 하나만 깨뜨려도 예방이 가능합니다. 이 질문은 조건 나열에 그치지 않고 '어느 조건을 어떻게 깨느냐'는 후속 질문이 거의 항상 따라옵니다.

관련 개념 필요조건(충분조건 아님), 자원 할당 그래프의 사이클, 예방 기법과의 매핑

Q2. 데드락 처리 전략 4가지(예방·회피·탐지 및 회복·무시)를 비교하세요. ★★★

답안 예방은 4가지 필요조건 중 하나를 구조적으로 제거해 데드락 자체를 불가능하게 하지만 자원 이용률이 떨어집니다. **회피**는 자원 요청 시마다 시스템이 안전 상태를 유지하는지 검사해(은행원 알고리즘) 위험한 할당을 거부하는 방식으로, 최대 자원 요구량을 미리 알아야 해서 실용성이 낮습니다. **탐지 및 회복**은 데드락을 허용하되 주기적으로 탐지해 프로세스 강제 종료나 자원 선점으로 회복합니다(DBMS의 방식). **무시(타조 알고리즘)**는 데드락이 드물다는 전제로 아무것도 안 하는 것으로, 리눅스·윈도우 등 범용 OS의 실제 선택입니다. 발생 시 사용자가 프로세스를 죽이거나 재부팅합니다.

관련 개념 은행원 알고리즘, 타조 알고리즘, DBMS 데드락 탐지, 비용-빈도 트레이드오프

Q3. 데드락 예방에서 각 필요조건을 깨는 방법을 구체적으로 설명하세요. ★★★

답안 상호 배제 부정: 자원을 공유 가능하게 만들지만(읽기 전용 파일 등) 본질적으로 배타적인 자원에는 불가능합니다. **점유와 대기 부정**: 실행 전에 필요한 모든 자원을 한꺼번에 요청하게 하거나, 새 자원 요청 시 보유 자원을 모두 반납 후 재요청하게 합니다. 자원 이용률 저하와 기아가 단점입니다. **비선점 부정**: 요청이 거부되면 보유 자원을 강제로 반납시킵니다. 상태 저장/복원이 가능한 자원(CPU, 메모리)에만 적용됩니다. **순환 대기 부정**: 모든 자원에 전역 순서를 부여하고 오름차순으로만 획득하게 합니다. 실무에서 가장 현실적이고 널리 쓰이는 방법(락 순서 규칙)입니다.

관련 개념 전량 할당, 자원 순서화(lock ordering), 선점 가능 자원

Q4. 은행원 알고리즘(Banker's Algorithm)을 설명하세요. ★★★

답안 데드락 회피 알고리즘으로, 자원 요청을 받으면 '일단 할당했다고 가정'한 뒤 시스템이 **안전 상태**인지 검사하고, 안전할 때만 실제로 할당합니다. 안전 상태란 모든 프로세스가 최대 요구량까지 자원을 받아 종료할 수 있는 실행 순서(안전 순서열)가 존재하는 상태입니다. Available(가용량), Max(최대 요구), Allocation(할당량), Need(Max−Allocation) 행

렬로 계산하며, Need ≤ Available인 프로세스를 찾아 종료시키고 자원을 회수하는 과정을 반복해 전부 종료 가능하면 안전합니다. 각 프로세스의 최대 요구량을 미리 알아야 하고 검사 비용이 커서 실제 OS에서는 쓰이지 않지만, 개념 검증 문제로 자주 출제됩니다.

관련 개념 안전 상태/불안전 상태, 안전 순서열, Need 행렬 계산 문제

Q5. 불안전 상태(unsafe state)는 곧 데드락인가요? ★★

답안 아닙니다. 안전 상태는 데드락이 절대 발생하지 않음을 보장하는 상태이고, 불안전 상태는 '데드락으로 갈 **가능성**이 있는' 상태일 뿐입니다. 프로세스들이 실제로는 최대 요구량까지 자원을 쓰지 않고 종료할 수도 있으므로 불안전 상태에서도 데드락 없이 지나갈 수 있습니다. 포함 관계로 보면 데드락 상태 ⊂ 불안전 상태입니다. 은행원 알고리즘은 불안전 상태로의 진입 자체를 차단하는 보수적 전략입니다.

관련 개념 안전/불안전/데드락 상태의 포함 관계, 보수적 회피의 이용률 손실

Q6. 데드락 탐지 후 회복 방법에는 어떤 것들이 있나요? ★★

답안 탐지는 자원 할당 그래프의 사이클 검사(자원이 단일 인스턴스일 때) 또는 탐지 알고리즘(다중 인스턴스)으로 합니다. 회복은 ① **프로세스 종료**: 데드락에 걸린 프로세스를 모두 죽이거나, 하나씩 죽이며 사이클이 풀리는지 확인합니다. 희생자는 우선순위, 수행 시간, 사용 자원량 등을 고려해 선택합니다. ② **자원 선점**: 일부 프로세스에서 자원을 빼앗아 다른 프로세스에 주고, 빼앗긴 프로세스는 안전한 지점으로 **롤백**시킵니다. 같은 프로세스만 반복 희생되는 기아를 막기 위해 롤백 횟수를 비용에 반영합니다. DBMS가 트랜잭션 하나를 골라 abort시키는 것이 대표적 실사례입니다.

관련 개념 wait-for 그래프, 희생자 선택, 롤백, 트랜잭션 abort

Q7. 실무(DB, 분산 시스템)에서 데드락을 겪는다면 어떻게 진단하고 해결하겠습니까? ★★

답안 DB: MySQL InnoDB는 데드락을 자동 탐지해 한 트랜잭션을 롤백하며, SHOW ENGINE INNODB STATUS로 최근 데드락 정보를 확인합니다. 해결은 트랜잭션들이 테이블·행을 같은 순서로 접근하도록 통일하고, 트랜잭션을 짧게 유지하며, 필요한 인덱스를 추가해 락 범위를 줄이는 것입니다. **애플리케이션**: 스레드 덤프(jstack 등)에서 락 대기 사이클을 확인하고 락 순서를 통일합니다. **분산 시스템**: 전역 탐지가 어려우므로 락에 타임아웃·TTL을 걸어 무한 대기를 차단하고 재시도(지수 백오프)로 풀어냅니다. 공통 원칙은 '락 순서 통일 + 락 시간 최소화 + 타임아웃'입니다.

관련 개념 InnoDB 데드락 탐지, 스레드 덤프, 락 타임아웃, 지수 백오프

Q8. 자원 할당 그래프에서 사이클이 있으면 항상 데드락인가요? ★★

답안 자원이 **단일 인스턴스**뿐이라면 사이클은 데드락의 필요충분조건이므로 사이클=데드락입니다. 그러나 자원에 **인스턴스가 여러 개**면 사이클이 있어도 다른 인스턴스를 가진 프로세스가 종료하며 자원을 반납해 사이클이 풀릴 수 있으므로, 사이클은 필요조건일 뿐입니다. 이 경우 데드락 여부는 탐지 알고리즘으로 판정해야 합니다. 그래프 문제로 출제될 때 인스턴스 개수를 먼저 확인하는 것이 포인트입니다.

관련 개념 자원 할당 그래프 표기(요청/할당 간선), 단일 vs 다중 인스턴스, 필요조건 vs 필요충분조건