1. **병행 프로세스1 개관**

병행 프로세스의 기본 개념과 상호배제와 동기화 문제 해결하기 위해 제공되는 장치 학습목표

1. **병행성과 병행 프로세스의 개념을 이해한다.**
2. **병행성 문제를 이해한다.**
3. **세마포어의 개념을 이해한다. 주요용어**

병행성

독립 프로세스 협력 프로세스 임계영역 상호배제 동기화 세마포어

* 1. **병행 프로세스의 개요**
     1. **병행성과 병행 프로세스 [51]**

# 병행성(concurrency)

여러개의 프로세스 또는 쓰레드가 동시 수행되는 시스템의 특성 병행 프로세스

동시에 수행되는 여러 개의 프로세스 또는 쓰레드 병행 프로세스가 실행되는 형태는 ()에 따라 차이가 있다.

**CPU의 개수**

하나의 CPU에서 병행 프로세스가 실행되는 경우 인터리빙 형식

각 프로세스가 짧은 시간간격으로 번갈아 가며 실행

정확히 한 순간에는 하나의 프로세스가 실행, 전체적으로 보면 여러 프로세스가 동시에 수행 여러 개의 CPU에서 병행 프로세스가 실행되는 경우

정확히 한순간에도 여러 프로세스가 실행되는 병렬처리 방식

**여러 개의 CPU, 즉 멀티프로세서(multiprocessor) 시스템에서 병행 프로세스가 실행되는 경우 메모리 구조**

**강결합(tighty-coupled) 시스템 여러 CPU가 하나의 기억장치 공유**

# 예) master-slave, SMP(Symmetric MultiProcessing; 대칭구조)

**마스터 슬레이브는 약결합 시스템이라는 의견도 있음 (RAM을 공유하지 않는 경우 대부분) 약결합(loosely-coupled) 시스템**

각 시스템은 자신의 운영체제와 기억장치를 가지고 있으며, 독립적으로 운영되고 필요할 때 통신한다. 2개 이상의 독립된 컴퓨터 시스템이 네트워크로 서로 연결

시스템 부하를 조절하기 위해 부하가 적은 시스템에 작업을 보내기도 함. 약결합 시스템 간 통신

메시지 전달

**원격 프로시저 호출 (Remote Procedure Calls: RPC)**

**예) 클러스터 구조 : Cluster Structure**

* 1. **병행 프로세스의 개요 (2)**
     1. **프로세스 간의 관계 [53]**

독립 프로세스

자신의 데이터, 상태 공유 x

수행 중인 다른 프로세스에 영향을 주지도 않고 받지도 않는 프로세스 독립 프로세스의 실행

# 결정적(deterministic)

재생가능

입력 데이터만 동일하다면 실행결과 항상 동일 다른 프로세스와 무관하게 중단/재시작 가능 협력 프로세스

다른 프로세스와 데이터 공유, 자신의 상태 공유

수행 중인 다른 프로세스와 영향을 주고받으며 동작하는 프로세스 협력 프로세스의 실행

**비결정적(nondeterministic) 재생 불가능**

입력 데이터가 동일하더라도 실행결과는 달라질 수 있음

예) 두 사람이 하나의 계좌를 이용하여 돈을 입금하거나 출금하는 경우

**잔고(자원): 10,000 프로세스A: 5,000 입급 프로세스B: 15,000 출금 협력 프로세스는 자원을 공유, 상태를 공유, 일 처리의 서순이 중요하다. => 멱등성 보장X**

* 1. **병행성 문제 [54]**

협력 프로세스는 병행성과 관련한 다양한 문제가 발생할 수 있다.

* + 1. **상호배제**

**상호 배제 (mutual exclusion)**

**2개 이상의 프로세스가 동시에 임계영역 수행하지 못하도록 하는 것 임계영역 (critical section)**

2개 이상의 프로세스가 동시에 사용하면 안 되는 공유자원을 액세스하는 프로그램 코드 영역 예제 (갱신 손실)

**자원: 10,000원 프A: +5,000 입금 프B: +8,000 입금**

# 프A: 10,000 읽기 ---> +5,000 = 15,000원 프B: --> 10,000 읽기 ---> +8,000 = 18,000원

**===================================**

갱신 손실 발생 해결책

임계영역, 상호배제를 명확하게 설정

**최종 18,000원**

* + 1. **동기화**

임계영역: 계좌의 잔고를 읽고 입금액을 더한 후 더해진 액수를 계좌에 쓰는 일련의 입금과정에 해당하는 코드

상호배제: 한 프로세스가 임계영역에 해당하는 입금과정을 수행 중이라면, 다른 프로세스는 입금과정을 수행하지 못하 도록 하는 것

**프로세스 동기화(synchronization)**

2개 이상의 프로세스에 대한 "처리순서를 결정"하는 것

**예) 한 계좌에서 다른 계좌로 10, 000원을 송금한다는 가정**

프로세스 A는 출금을 담당하고 프로세스 B는 입금을 담당할 때, A가 10,000원을 출금 처리한 후 프로세스 B가 이어서

**10,000원 입금을 처리하도록 동기화 필요**

상호배제 또한 임계영역에 대한 동기화 문제

* + 1. **통신**

**프로세스 간 통신(InterProcess Communication: IPC)**

서로 다른 프로세스들이 데이터를 주고받거나 동기화하는 방법

운영체제에서 각 프로세스는 독립적인 주소 공간을 가지므로, 프로세스 간 데이터를 직접 공유할 수 없고, 특정한 IPC 기법을 사용해야 함

통신하는 방법에 다라 상호배제 및 동기화 문제를 포함하여 다양한 문제 발생할 수 있음=> 5장 IPC 주요 방식

**메시지 기반(Message-based) 파이프(Pipe)**

# 소켓(Socket)

**메시지 큐(Message Queue)**

**메모리 기반(Memory-based) 방식 공유 메모리 (Shared Memory)**

# 메모리 매핑(Memory-Mapped Files, MMF)

**동기화(Synchronization) 방식 세마포어 (Semaphore)**

* 1. **세마포어 [56]**

**뮤텍스 (Mutex)**

**이벤트 (Event)**

* + 1. **세마포어의 정의 세마포어(semaphore)**

상호배제와 동기화 문제를 해결하기 위해 다익스트라가 제안한 도구 세마포어 s

정수형 공용변수

**(1) 사용 가능한 자원의 수 또는 (2) 잠김이나 풀림 상태의 값**

처음 세마포어 s를 선언할 때 상황에 맞는 값(0 이상인 정수)으로 초기화 한다. 세마포어 s는 기본 연산 P와 V에 의해서만 사용될 수 있다.

**연산 P와 V를 통하지 않고는 세마포어 s의 값을 확인하거나 수정할 수 없다. 세마포어의 기본 연산 (primitive operation)**

기본연산 P

**down(감소) / wait(대기)**

기본연산 V

**up(증가) / signal(진행 신호)**

'두 연산 P와 V가 기본연산이다'의 의미 인터럽트되지 않고 하나의 단위로 처리된다

어떤 프로세스가 세마포어 s에 대해 연산 P나 V를 수행 중이라면 다른 어떤 프로세스도 세마포어 s에 대해 연산 P나 V를 수행할 수 없다.

예를 들어, P(s)의 경우 s>0임을 확인 후 s--를 하기 전에 다른 프로세스에 의해 세마포어 s의 값이 바뀌는 일은 발 생하지 않는다.

* + 1. **상호배제 해결**

세마포어는 임계영역에 대한 상호배제 문제를 해결할 수 있다.

한 프로세스가 임계영역을 수행 중이라면, 다른 프로세스는 접근X 임계영역을 수행 중이던 프로세스가 임계영역을 벗어나면 접근 허용O

상호 배제 해결 방법

**임계영역 시작 부분 (진입영역): 연산 P**

임계영역에 대한 수행을 해도 되는지 체크 임계영역 끝 부분(해제영역) : 연산 V

다른 프로세스가 임계영역 수행을 시작할 수 있도록 함

진입 영역에서 대기하는 프로세스들은 '적절한 시간' 내에 임계영역 수행을 시작할 수 있어야 한다. 세마포어 mutex

임계영역 수행의 시작 가능 여부 초기 값 1

임계영역을 수행할 수 있음

# 0

임계 영역을 수행할 수 없음

**진입영역을 P(mutex)로 두면 어떤 프로세스가 P(mutex)를 수행할 때**

**mutex가 1인 경우**

**mutex만 0으로 바뀌고 그 프로세스는임계영역 수행을 시작하게 된다. mutex가 0인 경우**

**현재 프로세스는 대기 상태로 전이되고 mutex의 대기 큐에 추가 된다. 이 때 mutex는 변함없이 0이다.**

**해제영역을 V(mutex)로 두면임계영역 수행을 마친 프로세스가 V(mutex)를 수행할 때**

**mutex의 대기 큐가 비어있다면**

**mutex만 1로 바뀐다.**

**mutex의 대기 큐가 비어있지 않다면**

가장 오래 기다린 프로세스를 큐에서 빼서 준비상태로 전이시킨다. 이 때 mutex는 계속 0이다.

준비상태로 전이된 프로세스가 디스패치되어 실행상태가 되면 연산 P가 완료되며, 임계영역 수행을 시작

세마포어와 뮤텍스는 모두 세마포어의 한 종류다

**4.3. 세마포어 (2) [60]**

* + 1. **동기화 해결**

세마포어는 프로세스 동기화 문제를 해결할 수 있다.

예) 프로세스 A가 특정 코드 S1을 수행한 후에 프로세스 B가 코드 S2를 수행하도록 동기화를 하려면? 코드 S2 앞에 연산 P를 두고 코드 S1 뒤에 연산 V를 두는 것

**이 때 세마포어 sync의 초기 값은 0이다.**

세마포어 변수의 초기 값을 0으로 두면, 연산 V로 시작하면 된다.

초기 값 0은 먼저 수행되어야 하는 코드 S1이 아직 수행되지 않았으므로 코드 S2를 수행할 수 없음을 의미한다.

요약

1. **병행성은 여러 개의 프로세스 또는 스레드가 동시에 수행되는 시스템의 특성을 의미하고, 동시에 수행되는 여러 개의 프로세스 또 는 스레드를 병행 프로세스라고 한다.**
2. **병행 프로세스는 프로세스 간의 관계에 따라 독립 프로세스 또는 협력 프로세스라고 한다.**
3. **협력 프로세스는 병행성과 관련하여 상호배제, 동기화, 통신과 관련한 다양한 문제가 발생할 수 있다.**
4. **임계영역은 2개 이상의 프로세스가 동시에 사용하면 안되는 공유자원을 액세스하는 프로그램 코드 영역을 의미한다.**
5. **상호배제는 2개 이상의 프로세스가 동시에 임계영역을 수행하지 못하도록 하는 것이다.**
6. **동기화는 2개 이상의 프로세스에 대한 처리순서를 결정하는 것이다.**
7. **세마포어는 상호배제와 동기화 문제를 해결하기 위한 정수형 공용변수로, 2개의 기본 연산 P와 V에 의해서만 사용될 수 있다.**