운 영 체 제

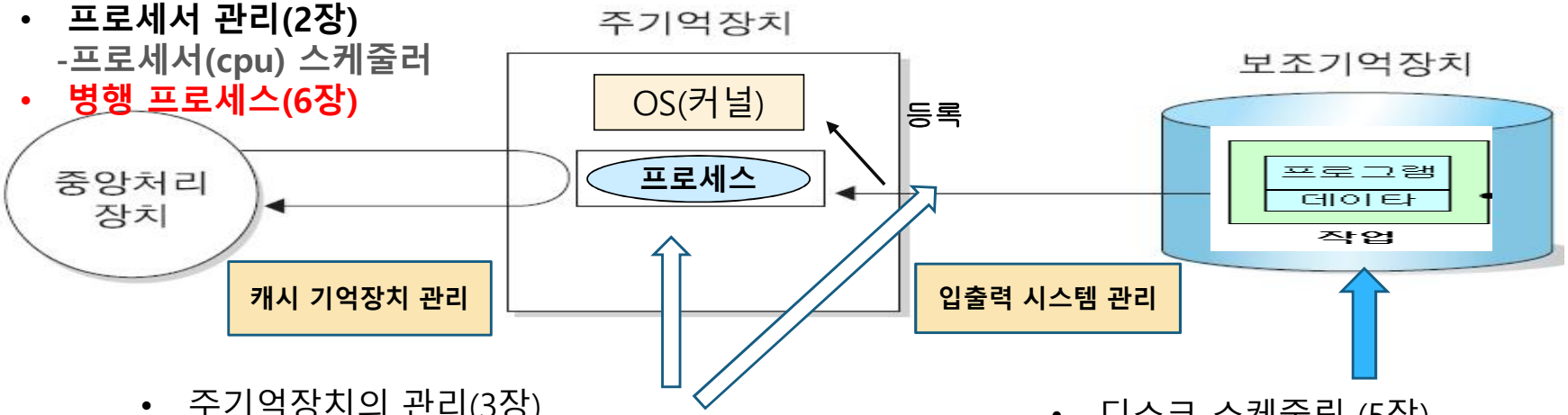
6장 프로세스 간 동기화 및 통신

6장 프로세스 간 동기화 및 통신 (병행 프로세스)

- 병행 프로세스
- 병행 처리의 문제점
- 임계 구역
- 상호 배제 기법
- 동기화 기법
- 세마포어
- 모니터

6장 프로세스 간 동기화 및 통신 (병행 프로세스)

- **프로세서 관리(2장)**
  -프로세서(cpu) 스케줄러
- **병행 프로세스(6장)**



- 주기억장치의 관리(3장)
  -주기억장치 관리 기법
  =>인출 기법,배치 기법,교체 기법, 할당 기법
- 가상 메모리 관리(4장)
  -분산 할당 기법
  =>페이징 기법, 세그먼테이션 기법
  -페이지 교체 기법
  => FIFO(First In First Out),
  OPT(Optimal Replacement), LRU, LFU, NUR, SCR

- 디스크 스케줄링 (5장)
  -FCFS, SSTF, SCAN, C-SCAN
  => 탐색시간 최소화
- 파일 시스템(5장)
  -파일 구조
  -디렉토리 구조

학습 내용

- 6장 프로세스 간 동기화 및 통신(병행 프로세스)
  - 병행 프로세스 개요
  - 병행 처리의 문제점
  - 병행 프로세스의 종류
  - 상호배제의 개념
  - 임계영역
  - 상호 배제 기법
  - 데커의 알고리즘

병행 프로세스 개요

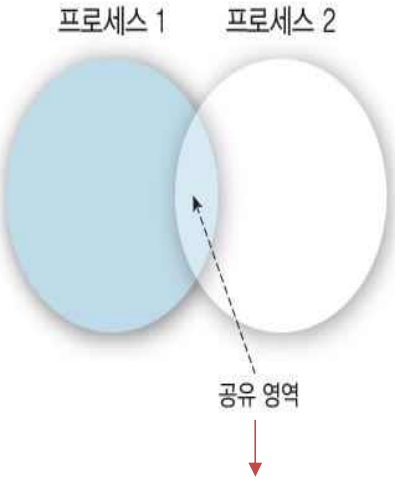
- 병행 프로세스
  - 두 개 이상의 프로세스들이 동시에 존재하며 실행상태에 있는 것을 의미
  - 한정된 컴퓨터 하드웨어나 자원을 공유하고, 동시에 작업을 수행하기 위해 사용하는 개념
  - 프로세서 하나는 한 번에 프로세스 하나만 실행 할 수 있지만. 운영체제가 프로세서를 빠르게 전환, **프로세서 시간을 나눠 마치 프로세스 여러 개를 동시에 실행하는 것처럼 보이게 하는 것**
  - 여러 프로세스들이 독립적으로 실행되는 것을 독립적 병행프로세스, 서로 협력하며 동시에 실행되는 것을 협동적 병행프로세스라고 한다.
    - 서로 관련 없이 독립적으로 수행 ➔ 독립적 병행 프로세스(단일 처리 시스템)
    - 다른 프로세스들과의 협력을 통해서 기능을 수행 ➔ 협력적 병행 프로세스
  - 병행 프로세스는 다중 프로그래밍과 다중처리시스템이나 분산처리시스템에서 중요한 개념으로 사용된다.
  - 협력적 병행 프로세스
    - 제한된 자원을 공유하기 위하여 상호 작용이 필요
    - 프로세스들을 동기화하지 않으면 교착상태, 임계영역 문제, 결과를 예측할 수 없는 상황 등 여러 문제들이 발생
    - 동기화 필요

병행 프로세스 개요

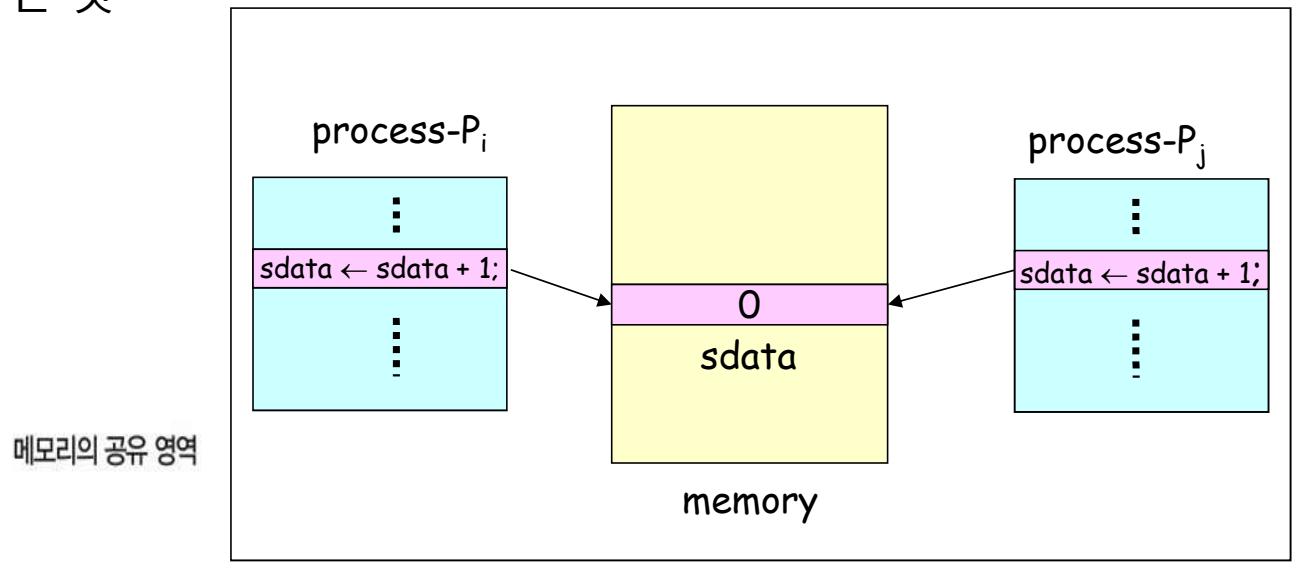
- 병행성과 병렬성 비교
- 공통점: 동일한 시간 동안 여러 프로세스를 동시 실행
- 차이점:
  - 병렬성 : 병렬성을 기반으로 한 병렬 컴퓨팅은 **다중 프로세서 시스템에서** 동일한 시간에 별도의 프로세서에서 실행 하는 것
  - 병행성 : 병행성을 기반으로 한 동시 컴퓨팅은 모두 동일한 시간에 실행 할 필요가 없는 것으로 예를 들어 병행 프로세스는 시분할로 각 프로세스의 실행 단계를 전환하여 어느 한 순간에는 프로세스 하나를 실행하도록 하는 방법으로 단일 프로세서 시스템에서도 가능함

병행 프로세스 개요

- 병행 프로세스 개념
  - 한정된 컴퓨터 하드웨어나 자원을 공유하고, 동시에 작업을 수행하기 위해 사용하는 개념
  - 운영체제가 프로세서를 빠르게 전환, 프로세서 시간 나눠 마치 프로세스 여러 개를 동시에 실행하는 것처럼 보이게 하는 것



모든 프로세스가 동시에 공유



메모리의 공유 영역

공유 영역 : 공유 자원으로 CPU, 메모리, 디스크, 입출력장치 버퍼등이 해당됨
=> 메모리 자원은 공유 영역에서 병렬(parallel)로 사용
=> 입출력 장치 일부나 프로세서는 한번에 프로세스 하나만 사용 할 수 있는 공유 자원임

병행 처리의 문제점

- 병행 프로세스의 문제는 컴퓨터 시스템의 자원에는 어느 한 시점에 하나의 프로세스가 할당되어 수행되는데, 동시에 두 개 이상의 프로세스를 병행처리 하면 여러 가지 문제점이 발생
- 문제점
  - 공유 자원 상호 배타적 사용(프린터, 통신망 등은 한 순간에 프로세스 하나만 사용)
  - 한 기능을 공유해 수행하는 두 프로세스 간의 동기화 문제(synchronization)
  - 두 프로세스 간 데이터 교환을 위한 통신 문제(communication)
  - 실행 순서와는 무관하게 항상 같은 결과를 얻을 수 있어야 하는 확정성 문제 (determinancy)
  - 교착상태 문제(deadlock)
  - 프로그래밍 언어를 통한 병행 처리 문제(concurrent programming)
  - 올바른 실행을 검증하는 문제(verification)

  =>해결방법 : 임계구역, 상호배제 기법, 동기화 기법, 교착 상태 해결

상호배제의 개념

상호배제(mutual exclusion)의 개념

- 병행 프로세스에서 프로세스 하나가 공유 자원 사용 시 다른 프로세스들이 동일한 일을 할 수 없도록 하는 방법(기법, 알고리즘)

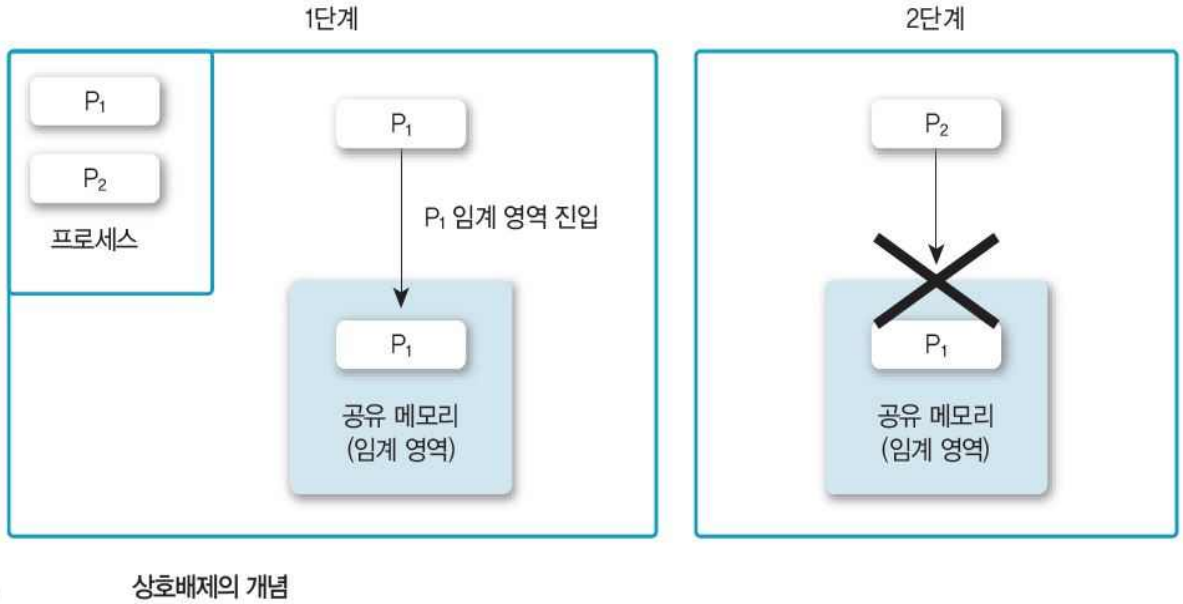
- 읽기 연산은 공유 데이터에 동시에 접근해도 문제 발생 않음.

- 동기화 : 변수나 파일은 프로세스별로 하나씩 차례로 읽거나 쓰도록 해야 하는데,
  공유 자원을 동시에 사용하지 못하게 실행을 제어하는 방법 뜻 함.
  - 동기화는 순차적으로 재사용 가능한 자원을 공유하려고 상호작용하는 프로세스 사이에서 나타남
  - 동기화로 상호배제 보장할 수 있지만, 이 과정에서 교착 상태와 기아 상태가 발생할 수 있음

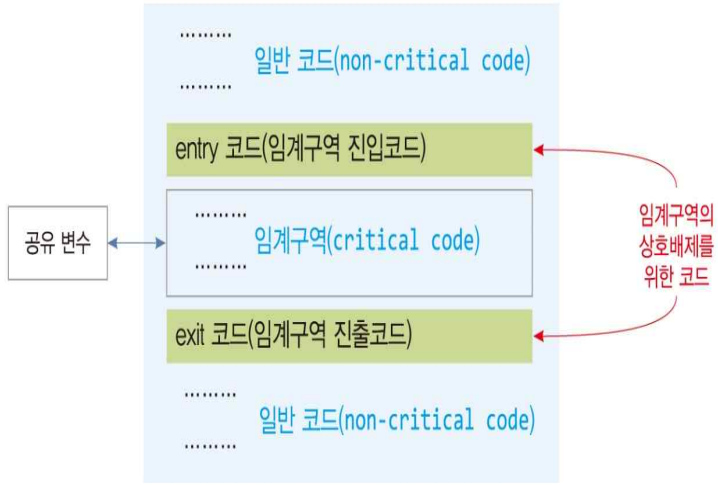
상호배제의 개념

상호배제의 구체적인 예

임계자원critical resource : 두 프로세스가 동시에 사용할 수 없는 공유 자원
임계영역(구역)critical section : 임계 자원(공유 데이터)에 접근하고 실행하는 프로그램 코드 부분



상호배제의 개념



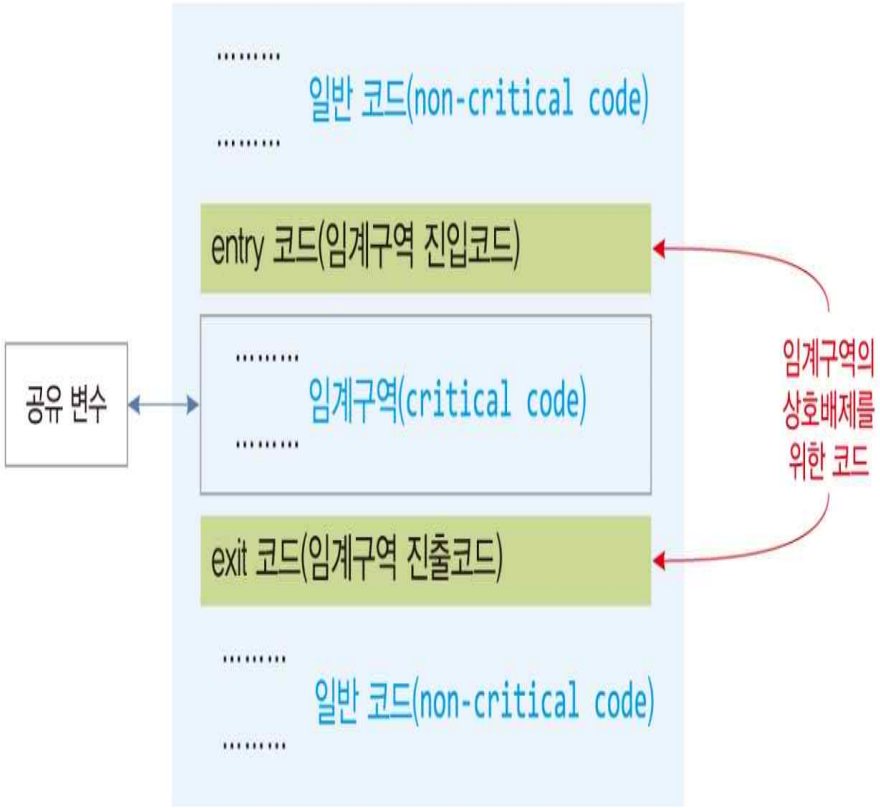
상호배제의 조건

❶ 두 프로세스는 동시에 공유 자원에 진입 불가.
❷ 프로세스의 속도나 프로세서 수에 영향 받지 않음
❸ 공유 자원을 사용하는 프로세스만 다른 프로세스 차단 가능
❹ 프로세스가 공유 자원을 사용하려고 너무 오래 기다려서는 안 됨

상호배제의 개념

■ 상호배제의 구체적인 예

임계자원critical resource : 두 프로세스가 동시에 사용할 수 없는 공유 자원
임계영역(구역)critical section : 임계 자원(공유 데이터)에 접근하고 실행하는 프로그램 코드 부분



1. 일반 코드(non-critical code)
   - 공유 데이터를 액세스하지 않는 코드

2. 임계구역 진입 코드(entry code)
   - 임계구역에 진입하기 전 필요한 코드 블록
   - 현재 임계구역을 실행 중인 프로세스가 있는지 검사
     - 없다면, 다른 프로세스가 들어오지 못하도록 조치
     - 있다면, 진입이 가능해질 때까지 대기

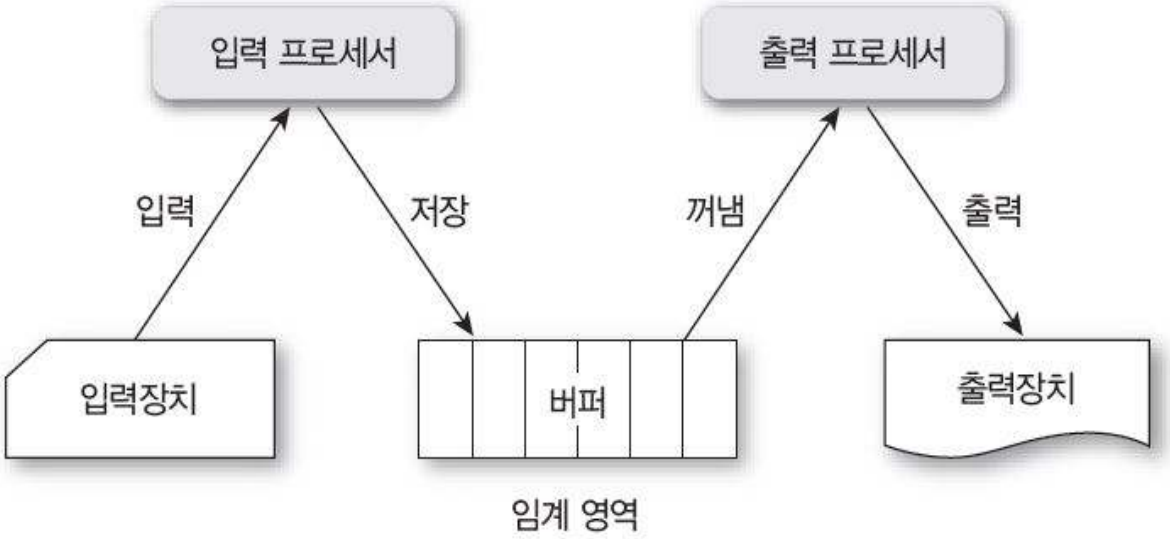
3. 임계구역 코드(critical code)

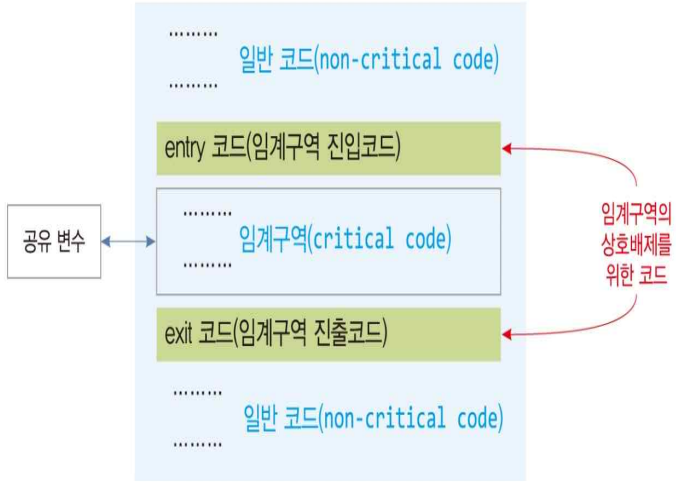
4. 임계구역 진출 코드(exit code)
   - 임계구역을 마칠 때 필요한 코드 블록
   - 대기중인 프로세스가 임계구역에 진입할 수 있도록, 진입 코드에서 취한 조치 해제

임계영역

임계 영역의 개념

- 임계자원critical resource : 두 프로세스가 동시에 사용할 수 없는 공유 자원
- 임계영역critical section : 임계 자원에 접근하고 실행하는 프로그램 코드 부분
- 다수의 프로세스 접근 가능하지만, 어느 한 순간에는 프로세스 하나만 사용 가능
- 예



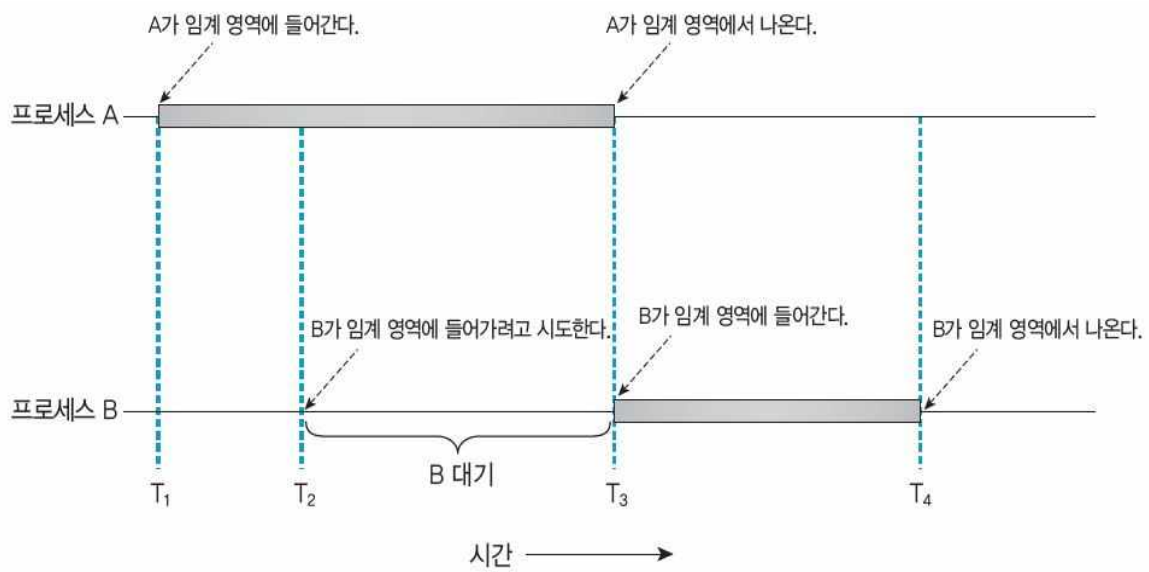


임계 영역 예

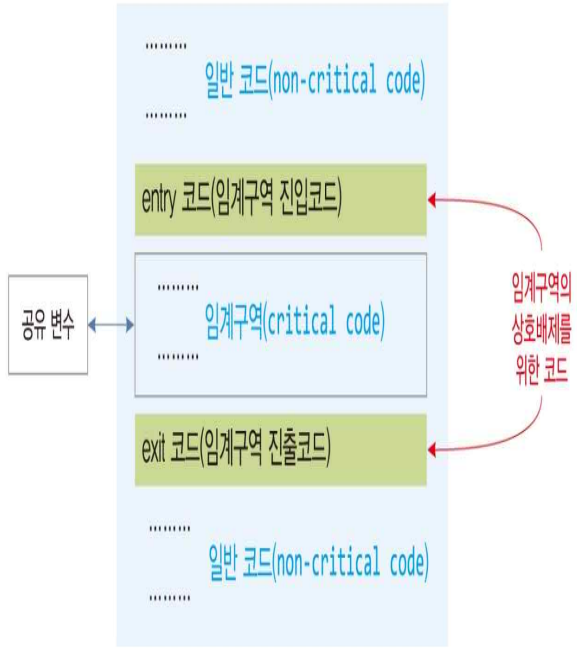
임계영역

임계 영역 이용한 상호배제

- 간편하게 상호배제 구현 가능(자물쇠와 열쇠 관계)
  - 프로세스가 진입하지 못하는 임계 영역(자물쇠로 잠근 상태)
- 어떤 프로세스가 열쇠 사용할 수 있는지 확인하려고 검사 하는 동작과 다른 프로세스 사용 금지하는 동작으로 분류
- 예



임계 영역을 이용한 상호배제



임계영역

병행 프로세스에서 영역 구분

```
do {
        while (turn != i);      ← 진입 영역
        /* 임계 영역 */
        turn = j;               ← 탈출 영역
        /* 나머지 영역 */
} while (TRUE)
```

병행 프로세스에서 영역 구분

-진입 영역은 각 프로제스가 임계 영역에 들어갈 수 있도록 요청하는 영역
-TURN은 전역 변수로 0으로 초기화한 후 I가 아니면 대기하고 I이면 임계 영역에 진입한다.
-탈출 영역은 임계 영역에서 나가는 프로세스를 선택하는 영역으로,TURN을 J로 변경한다.
-나머지 영역은 임계 영역을 나와 수행한다.

임계 영역의 조건

- 어떤 프로세스가 공유 자원을 접근(access)하고 있는 동안 그 프로세스는 임계 구역에 있다고 함.
- 임계 구역에 접근한 프로세스에게는 상호 배제가 보장 되어야 함

❶ 상호배제 : 어떤 프로세스가 임계 영역에서 작업 중, 다른 프로세스 임계 영역 진입 불가
❷ 진행 : 임계 영역에 프로세스가 없는 상태에서 어떤 프로세스가 들어갈지 결정
❸ 한정 대기 : 다른 프로세스가 임계 영역을 무한정 기다리는 상황 방지 위해 임계 영역에 한 번 들어갔던 프로세스는 다음에 임계 영역에 다시 들어갈 때 제한

상호 배제 기법(알고리즘)

- 임계 구역 문제 해결을 위한 세가지 충족요건
  - 상호 배제
    - 한 프로세스가 그들의 임계 구역에서 실행되고 있다면 다른 프로세스는 그 임계 구역에 들어갈 수 없다.
  - 진행
    - 임계구역으로 진입하려는 프로세스가 있을 경우, 진입을 결정하는데 참여할 수 있으며, 이 선택은 무한 연기 되어서는 안 된다.
  - 한정된 대기
    - 진입을 요청한 이후부터 그 요청이 허용될 때까지 다른 프로세스들의 임계구역 진입허용 횟수에 한계가 있어야
    - 임의의 프로세스의 기아(starvation)를 예방

상호 배제 기법(알고리즘)

- 특정 프로세스가 공유 자원을 사용하고 있을 경우 다른 프로세스가 해당 공유 자원을 사용하지 못하게 제어하는 기법이다.
- 여러 프로세스가 동시에 공유 자원을 사용하려 할 때 각 프로세스가 번갈아가며 공유 자원을 사용하도록 하는 것으로 임계 구역을 유지하는 기법이다.
- 상호 배제 기법을 구현하기 위한 방법에는 소프트웨어적인 방법과 하드웨어적인 방법이 있다.

  - 소프트웨어적 구현 방법
    - 두 개의 프로세스 기준 :데커 알고리즘, 피터슨 알고리즘
    - 여러 개의 프로세스 기준 : lemport 의 빵집 알고리즘(각 프로세스에게 번호를 부여하여 자원을 사용하도록 하는 방법)
  - 하드웨어적 구현방법
    - Test And Set 기법(원자 명령(cpu 명령)으로 함)과 Swap 명령어 기법이 있다.
    - **오늘날 대부분 하드웨어적 방법 사용**
      - 1970년대 Intel Pentium에서 시작, 현재 대부분의 CPU에 원자 명령 있음

데커의 알고리즘

데커의 알고리즘 개념

- 두 프로세스가 서로 통신하려고 공유 메모리를 사용하여 충돌 없이 단일 자원을 공유할 수 있도록 허용하는 것
- 다익스트라로 임계 영역 문제에 적용
- 병행 프로그래밍 상호배제 문제의 첫 번째 해결책
- 각 프로세스 플래그 설정 가능, 다른 프로세스 확인 후 플래그 재설정 가능
- 프로세스가 임계 영역에 진입하고 싶으면 플래그 설정하고 대기

데커의 알고리즘 특징

- 특별한 하드웨어 명령문 필요 없음
- 임계 영역 바깥에서 수행 중인 프로세스가 다른 프로세스들이 임계 영역 진입 막지 않음
- 임계 영역에 들어가기를 원하는 프로세서 무한정 기다리게 하지 않음

데커의 알고리즘

▪ 데커의 알고리즘을 사용한 상호배제

데커의 알고리즘을 이용한 상호배제

```
// 프로세스가 공유하는 데이터 flag[] : 부울(boolean) 배열, turn : 정수
flag[0] = false;
flag[1] = false;
turn = 0;                          // 공유 변수, 0 또는 1

// 프로세스 P0:                    // 프로세스 P0의 임계 영역 진입 절차
❶ flag[0] = true;                  // P0의 임계 영역 진입 표시
❷ while (flag[1] == true) {        // P1의 임계 영역 진입 여부 확인
      if (turn == 1) {             // P1이 진입할 차례가 되면
        ❹ flag[0] = false;         // 플래그를 재설정하여 P1에 진입 순서 양보
          while (turn == 1) {      // turn을 바꿀 때까지 대기
        ❺     // 바쁜 대기
          }
          flag[0] = true;          // P1이 임계 영역에 재진입 시도
      }
}
❸ /* 임계 영역 */;
turn = 1;                          // P1에 진입 순서 제공
flag[0] = false;                   // P0의 임계 영역 사용 완료 지정
/* 나머지 영역 */;                 // P0이 나머지 영역 수행

// 프로세스 P1
flag[1] = true;
while (flag[0] == true) {
     if (turn == 0) {
          flag[1] = false;
          while (turn == 0) {
               // 바쁜 대기
          }
          flag[1] = true;
     }
}
/* 임계 영역 */;
turn = 0;
flag[1] = false;
/* 나머지 영역 */;
```

❶에서 P0은 flag[0]을 true로 설정, 자신이 임계 영역으로 들어간다는 사실 알림
❷에서 while 문 검사하여 P1의 임계 영역 진입 여부 확인, P1의 flag[1]이 false이면
❸ P0이 임계 영역으로 진입하고, true이면
❹ P1이 임계 영역에 진입할 차례라서 플래그 false로 재설정 후
❺ while 문에서 순환하며 대기
여기서 공유 변수 turn은 두 프로세스 P0과 P1이 동시에 임계 영역으로 들어가려고 충돌하는 것 방지

6장 프로세스 간 동기화 및 통신 (병행 프로세스)

- 병행 프로세스
- 병행 처리의 문제점
- 임계 구역
- 상호 배제 기법
- 동기화 기법
- 세마포어
- 모니터

학습 내용

- 6장 프로세스 간 동기화 및 통신(병행 프로세스)
  - 동기화 기법
    - 세마포어
    - 모니터

동기화 기법(Synschronization)

- 동기화 기법은 두 개 이상의 프로세스를 한 시점에서는 동시에 처리할 수 없으므로 각 프로세스에 대한 처리순서를 결정하는 것으로 , 상호배제의 한 형태이다.
- 동기화를 구현할 수 있는 방법
  - 세마포어
    - N개의 공유 자원을 다수의 프로세스가 공유하기 위한 기법
    - 각 프로세스에 제어 신호를 전달하여 순서대로 작업을 하는 기법이다.
    - 세마포어에는 P연산(wait 동작)과 V(signal 동작)이 사용된다.
  - 모니터
    - 내부의 프로시저와 데이터 정보를 은폐시켜서 다른 외부 프로시저가 접근하거나 변경하지 못하도록 하는 기법이다.
    - 모니터에는 wait 연산과 signal 연산이 사용된다.

세마포어(semaphore)

■ 세마포어 개념과 동작

- 세마포어는 "신호기","깃발"을 뜻하며, 각 프로세스에 제어 신호를 전달하여 순서대로 작업을 하는 기법이다.
- N개의 공유 자원을 다수의 프로세스가 공유하기 위한 기법
  - Ex) n개의 프린터가 있는 경우, 프린터를 사용하고자 하는 다수 프로세스의 프린터 사용 관리

- 각 프로세스에 제어신호를 전달하여 순서대로 작업을 수행하도록 하는 기법이다.
- E.J.Dijkstra가 제안하였으며, 사용되는 연산에는 P와 V, 초기치 연산(Semaphore Initialize)이 있고, 상호 배제의 원리를 보장한다.
- S는 P와 V 연산으로만 접근 가능한 세마포어 변수로, 공유 자원의 개수를 나타내며 0 또는 1이나 0 또는 양의 값을 가질 수 있다.
- P 연산 : 자원을 사용하려는 프로세스들의 진입 여부를 자원의 개수(S)를 통해 결정하는 것으로, Wait 동작이라고 함
- V 연산 : 대기중인 프로세스를 깨우는 신호(Wake Up)로서, Signal 동작이라고 함

■ 예

세마포 예 : 열차 차단기

세마포(semaphore)

- 세마포 정의 :동기화 도구

세마포 정의

```
P(S) : wait(S) {                    // S = 1로 초기화
        while S <= 0
            ;                       // 바쁜 대기, S > 0 때까지 대기
        S--;
    }

V(S) : signal(S) {
        S++;                        // 다른 프로세스의 접근 허용
    }
```

```
P(S) {
    //임계 영역에 진입하는 연산
    //자원 요청 시 실행하는 연산
    //자원 사용 허가를 얻는 과정
      while S ≤ 0  Do skip ; //S>0 될 때까지 대기
     S-- ; //자원 점유를 알림. 자원의 수 감소
    }
```

```
V(S)  {
     //임계 영역에서 나오는 연산
    //자원 반환 시 실행하는 연산
    //자원 사용이 끝났음을 알리는 과정
    S++ ; //자원을 반납하였으므로 자원의 수 증가
     }
```

- P(프로세스 대기하게 하는 wait 동작, 임계 영역에 진입하는 연산):원자명령(CPU 명령)
- V(대기 중인 프로세스 깨우려고 신호 보내는 signal 동작, 임계 영역에서 나오는 연산)):원자명령(CPU 명령)
- 세마포를 의미하는 S는 표준 단위 연산 P와 V로만 접근하는 정수 변수로 공유 자원의 개수를 나타남

- 프로세스가 자원을 사용하려고 할 경우 먼저 S를 통해 다름 프로세스가 자원을 공유하고 있는지 조사
- 자원을 점유 하였으면 자원이 점유 되었다는 것을 알리고 다른 프로제스가 이미 자원을 점유한 상태이면 기다린다.

세마포(semaphore)

- 세마포를 이용한 상호배제

- 세마포를 이용한 N개 프로세스의 상호배제 구현 절차

```
semaphore S=1;

while (1) {
        P(S) ;
        // 임계영역
        ...
        V(S) ;
    }
```

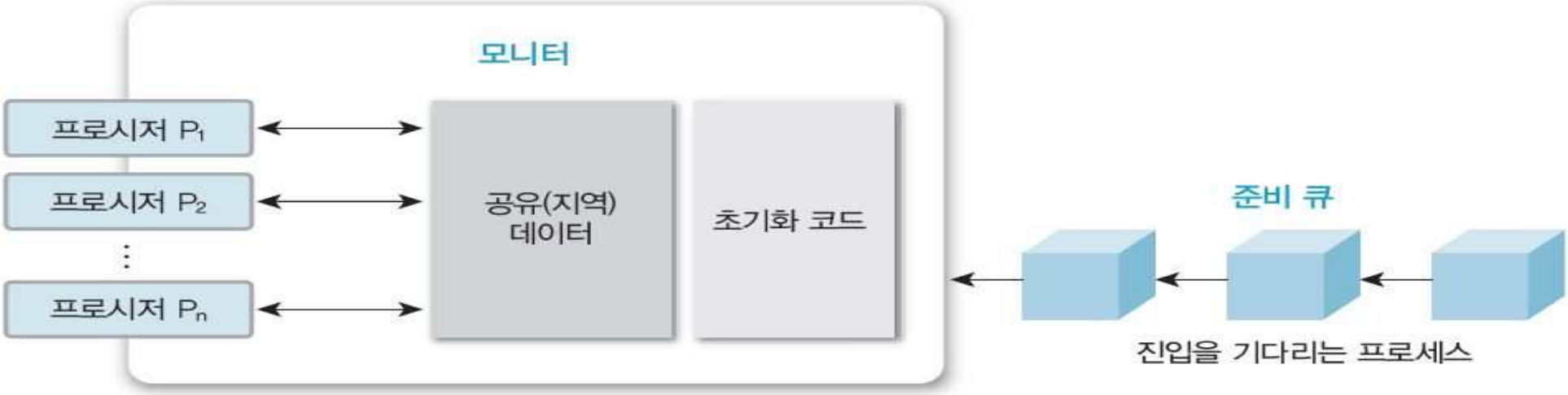
```
P(S) {
    //임계 영역에 진입하는 연산
    //자원 요청 시 실행하는 연산
    //자원 사용 허가를 얻는 과정
      while S ≤ 0  Do skip ; //S>0 될 때까지 대기
     S-- ; //자원 점유를 알림. 자원의 수 감소
    }
```

```
V(S)  {
     //임계 영역에서 나오는 연산
    //자원 반환 시 실행하는 연산
    //자원 사용이 끝났음을 알리는 과정
    S++ ; //자원을 반납하였으므로 자원의 수 증가
     }
```

모니터(monitor)

■ 모니터의 개념과 구조

- 세마포의 오용으로 여러 가지 오류가 쉽게 발생하면 프로그램 작성 곤란.
  이런 단점 극복 위해 등장
- 핸슨Hansen 제안, 호Hoare 수정한 공유 자원과 이것의 임계 영역 관리 소프트웨어 구성체
- 여러 프로세스들간의 공유 데이터와 임계 지역 코드들의 집합
- 사용자 사이에서 통신하려고 동기화하고, 자원에 배타적으로 접근할 수 있도록 프로세스가 사용하는 병행 프로그래밍 구조



모니터의 구조

모니터(monitor)

- 모니터의 구조

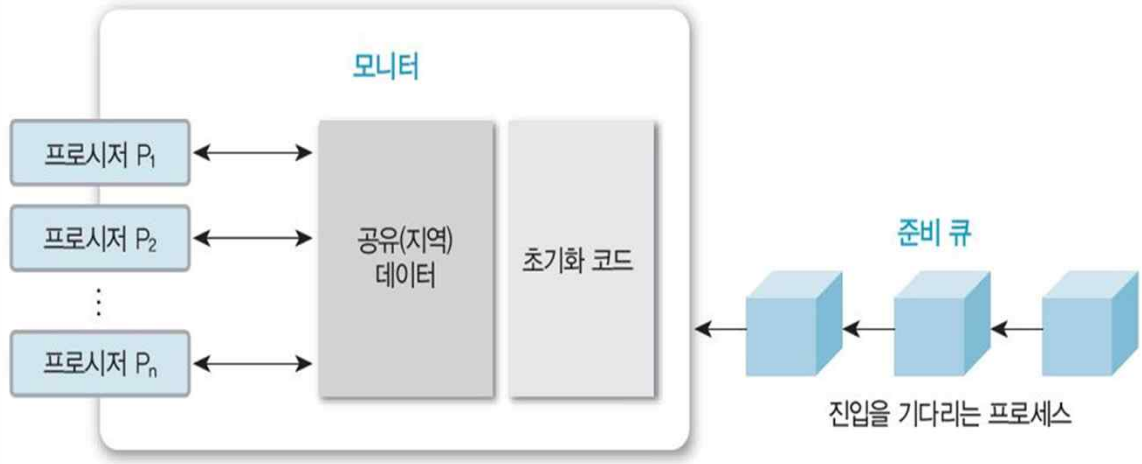
  내부의 프로시저와 데이터 정보를 은폐시켜서 다른 외부 프로시저가 접근하거나 변경하지 못하도록 하는 기법이다.

  고급언어에서 개발자의코드를 상호배제하게끔 만든 추상화 데이터 형태(도구)

모니터의 구조

```
Monitor monitor_name
{
                                        // 공유 데이터 변수 선언
        procedure P1(. . . .) {
                . . . .
        }
        procedure P2(. . . .) {
                . . . .
        }
        .
        .
        procedure Pn(. . . .) {
                . . . .
        }

        initialization_code(. . . .) {    // 코드 초기화
                . . . .
        }
}
```



모니터의 구조

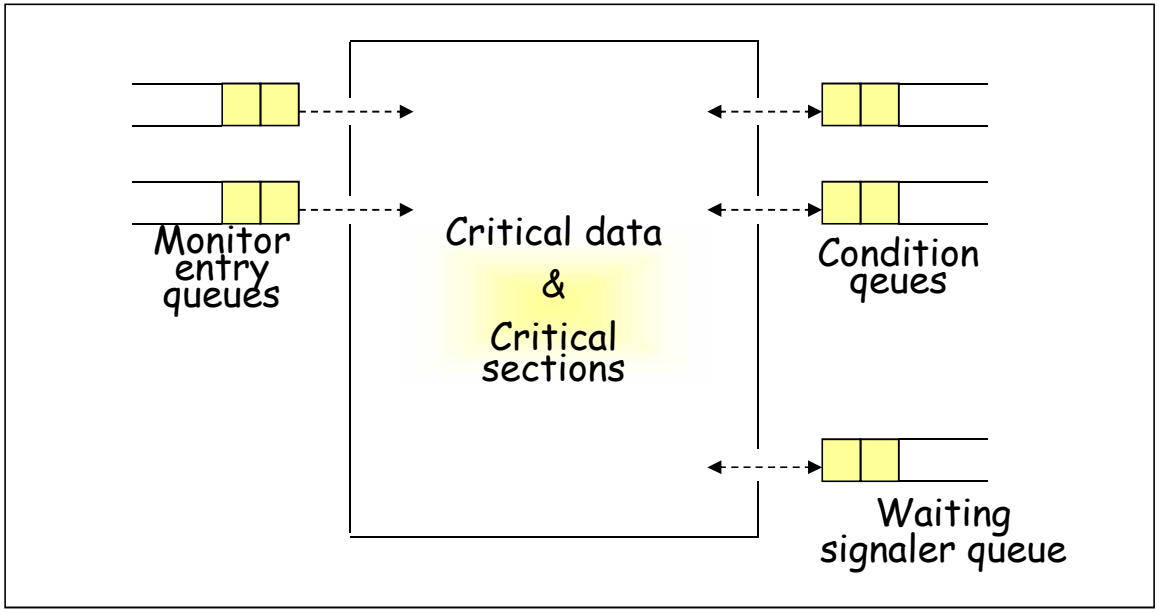
모니터(monitor)

모니터의 정의

- 여러 프로세스들간의 공유 데이터와 임계 지역 코드들의 집합

모니터의 구조

모니터의 구조



- 모니터의 기본 내용
  - 모니터 진입 큐(entry queue)
    - 모니터 내의 프로시져 수만큼 존재
  - 상호 배제 메커니즘 자동 보장
    - 모니터 내에는 항상 하나 이하의 프로세스만 진입 가능
  - 정보 은폐 (information hiding)
    - 공유 데이터에 대한 접근
      - 모니터 내에 진입하여 자신이 호출한 프로시져를 실행하는 방법 외에 다른 방법으로 접근 불가능함
  - 조건 큐 (condition queue)
    - 모니터 내에 진입 후 공유 데이터의 값이 원하는 값이 아닐 경우프로세스가 원하는 조건에 해당하는 조건 큐에서 대기함
  - 신호제공자 큐 (signaler queue)
    - 모니터에 항상 하나의 신호제공자 큐가 존재
    - signal() 명령을 실행한 프로세스가 임시 대기

모니터(monitor)

- **모니터와 세마포 비교**
  - 세마포는 V 연산으로 사용자가 지연 없이 실행 재개
  - 반면 모니터는 signal 연산으로 모니터 잠금 해제할 때만 다시 시작
  - 따라서 병렬 프로그래밍에서는 세마포보다 모니터가 더 오류가 적고 쉽게 작성
    - 사용자들이 병행 프로그래밍을 보다 쉽게 할 수 있도록 지원함
    - 객체지향 개념 가짐
    - 코딩 중 오류 발생 가능성을 크게 줄임
  - 모니터의 주된 단점은 프로그래밍 언어의 일부로 구현하고 컴파일러가 그 코드 생성해야 한다는 점.
  - 이는 컴파일러가 병행 프로세스에서 임계 영역에 접근 제어할 수 있는 운영 체제를 이해해야 한다는 부담.
  - 병행성 지원하는 자바Java, C#, 비주얼 베이직Visual Basic, 에이다Ada 등 언어만 모니터 지원

병행 처리의 결론

- 병행 프로세스의 문제는 컴퓨터 시스템의 자원에는 어느 한 시점에 하나의 프로세스가 할당되어 수행되는데, 동시에 두 개 이상의 프로세스를 병행처리 하면 여러 가지 문제점이 발생

=>해결방법 : 임계구역, 상호 배제 기법, 동기화 기법, 교착 상태 해결

상호배제 방법들

수준	방법	종류
고급	소프트웨어로 해결	• 데커의 알고리즘 • 램포트의 베이커리(빵집) 알고리즘 • 크누스의 알고리즘 • 핸슨의 알고리즘 • 다익스트라의 알고리즘
	소프트웨어가 제공 : 프로그래밍 언어와 운영체제 수준에서 제공	• 세마포 • 모니터
저급	하드웨어로 해결 : 저급 수준의 원자 연산	TestAndSet[TAS](테스)

6장 임계 구역, 상호 배제

1. 다중 프로그래밍 운영체제에서 한 순간에 여러 개의 프로세스에 의하여 공유되는 데이터 및 자원에 대하여, 한 순간에는 반드시 하나의 프로세스에 의해서만 자원 또는 데이터가 사용되도록 하고, 이러한 자원이 프로세스에 의하여 반납된 후 비로소 다른 프로세스에서 자원을 이용하거나 데이터를 접근 할 수 있도록 지정된 영역을 의미하는 것은?
가. locality 나. semaphore 다. Critical section 라. Working set

2. 한 프로세스가 공유 메모리 혹은 공유 파일을 사용하고 있을 때 다른 프로세스들이 사용하지 못하도록 배제 시키는 제어 기법을 무엇이라고 하는가?
가. deadlock 나. Mutual exclusion 다. interrupt 라. Critical section

3. 임계영역에 대한 설명으로 옳은 것은?
가. 프로세스들의 상호 배제가 일어나지 않도록 주의해야 한다.
나. 임계영역에서 수행중인 프로세스는 인터럽트가 가능한 상태로 만들어야 한다.
다. 어느 한 시점에서 둘 이상의 프로세스가 동시에 자원 또는 데이터를 사용하도록 지정된 공유 영역을 의미한다.
라. 임계 영역에서의 작업은 신속하게 이루어져야 한다.

4. 모니터에 대한 설명으로 옳지 않은 것은?
가. 모니터의 경계에서 상호배제가 시행된다.
나. 자료 추상화와 정보은폐기법을 기초로 한다.
다. 순차적으로 재사용 가능한 특정 공유 자원 또는 공유 자원 그룹을 할당 하는 데 필요한 데이터 및 프로시저를 포함하는 병행성 구조이다.
라. 모니터 내의 데이터는 모니터 외부에서도 액세스 할 수 있다.

5. 병행 프로그래밍 기법 하에서 발생할 수 있는 오류에 대한 오류 방지 방법이 아닌 것은?
가. 세마포어 나. 비동기화 다. 상호배제 라. 모니터