1 병행 프로세스간 상호작용

1 병행 프로세스간 상호작용

1 상호 배제 개념

병행 프로세스에서 프로세스 하나가
공유 자원 사용 시 다른 프로세스들이 동일한
일을 할 수 없도록 하는 방법

- 읽기 연산은 공유 데이터에 동시에 접근해도
  문제 발생 않음

1 병행 프로세스간 상호작용

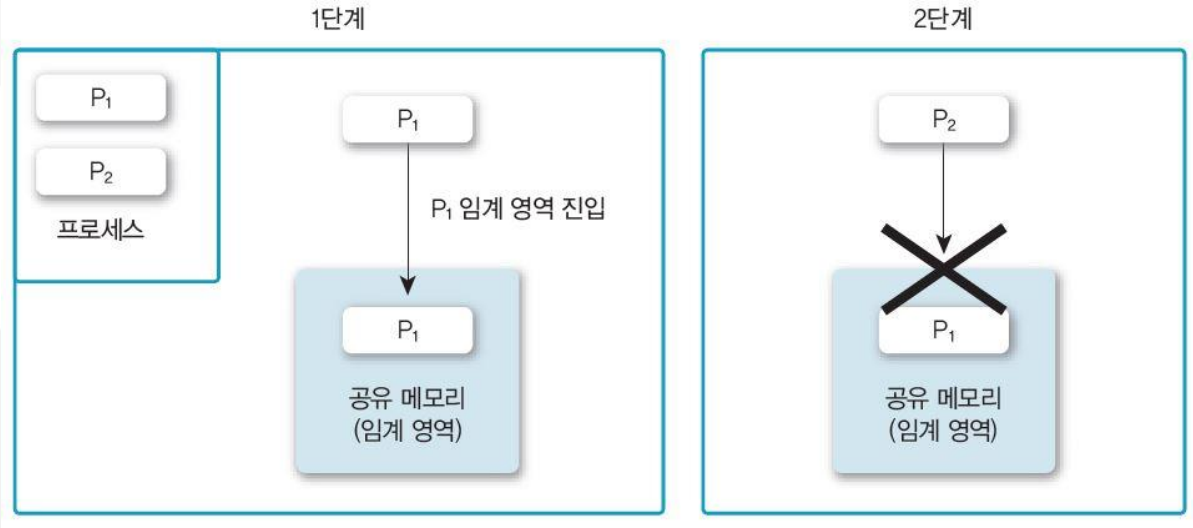
1 상호 배제 개념

※ 동기화란?

- 변수나 파일은 프로세스별로 하나씩 차례로 읽거나 쓰도록 해야 하는데, 공유 자원을 동시에 사용하지 못하게 실행을 제어하는 방법

1 병행 프로세스간 상호작용

2 상호 배제 예제



※ 출처 : 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미

1 병행 프로세스간 상호작용

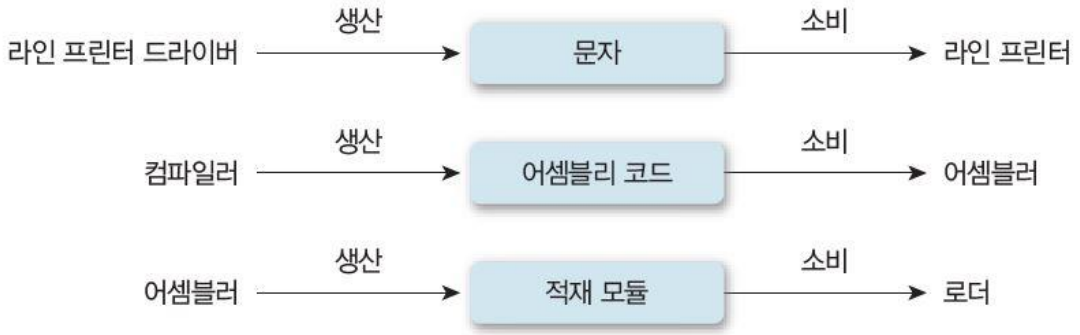
3 상호 배제 조건

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

1 병행 프로세스간 상호작용

4 생산자 소비자 프로세스

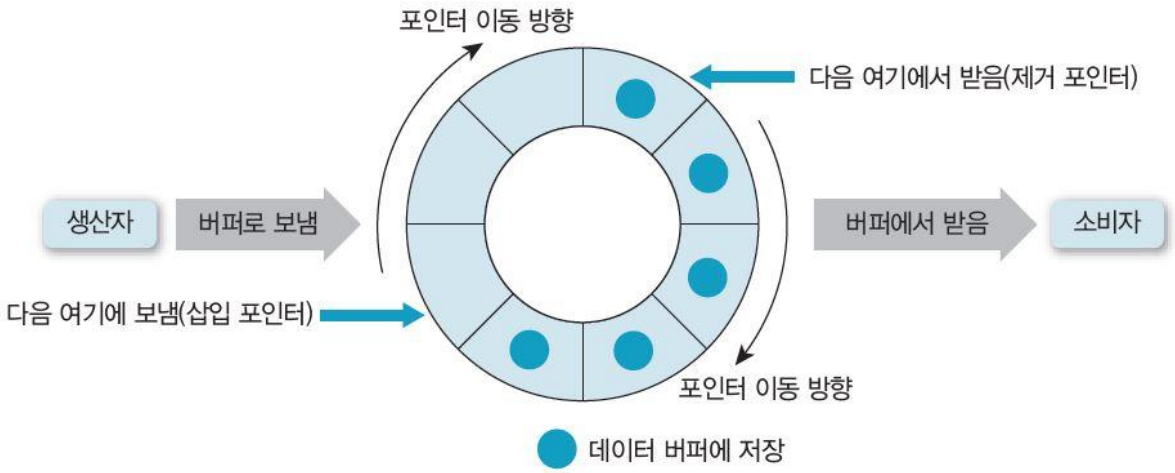
- 운영체제에서 비동기적으로 수행하는 모델
- 생산자 프로세스가 생산한 정보를 소비자 프로세스가 소비하는 형태



※ 출처 : 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미

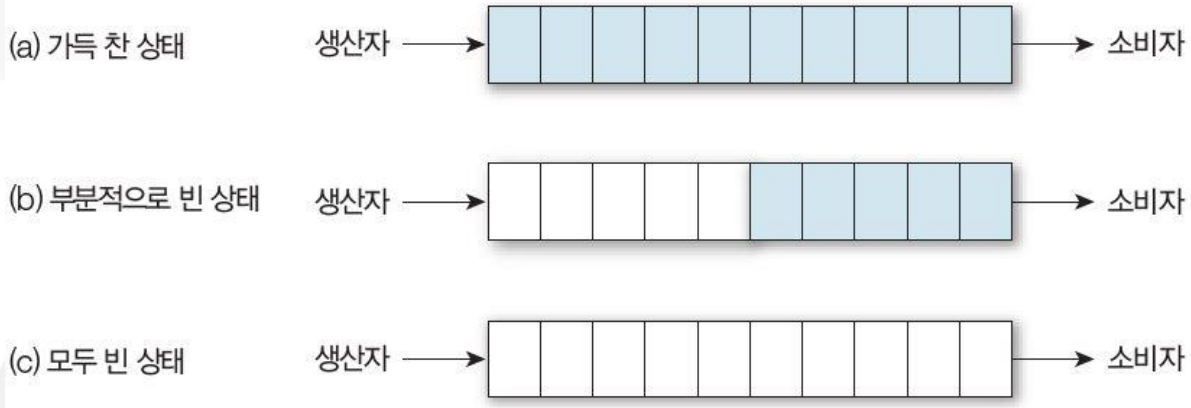
1 병행 프로세스간 상호작용

5 소비자에게 데이터 전송



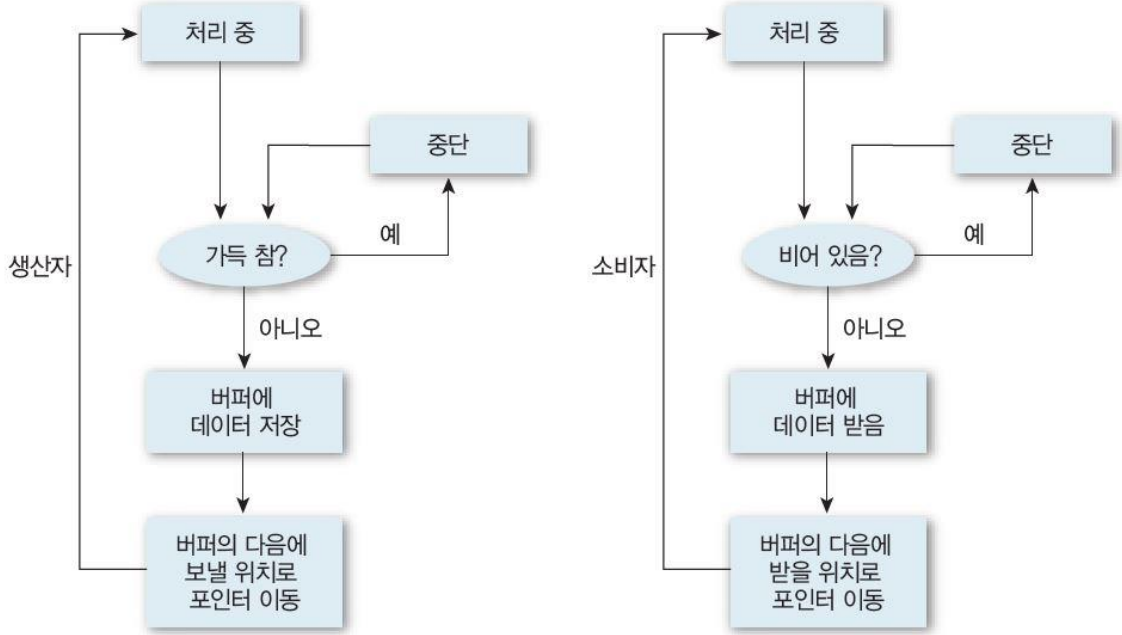
※ 출처 : 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미

6 생산자와 소비자의 공유 버퍼



※ 출처 : 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미

7 생산자와 소비자 문제와 상호배제 해결 방법



※ 출처 : 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미

1 병행 프로세스간 상호작용

8 생산자와 소비자 문제와 상호배제 해결 방법

예제 4-2 생산자와 소비자 프로세스

생산자 프로세스

```
item nextProduced;

while (true) {
    // 버퍼가 가득 차 아무 일도 하지 않음
    while (counter == BUFFER_SIZE);
    buffer[in] = nextProduced;
    in = (in + 1) % BUFFER_SIZE;
    counter++;
}
```

소비자 프로세스

```
item nextConsumed;

while (true) {
    // 버퍼가 비어 아무 일도 하지 않음
    while (counter == 0);
    nextConsumed = buffer[out];
    out = (out + 1) % BUFFER_SIZE;
    counter--;
}
```

※ 출처 : 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미

1 병행 프로세스간 상호작용

9 경쟁상태

- 여러 프로세스가 동시에 공유 데이터에 접근 시, 접근 순서에 따라 실행 결과 달라지는 상황 말함

- 공유데이터에 마지막으로 남는 데이터의 결과를 보장할 수 없는 상황

1 병행 프로세스간 상호작용

9 경쟁상태

- 장치나 시스템이 둘 이상의 연산 동시 실행 시,
  어느 프로세스를 마지막으로 수행한 후
  결과를 저장했느냐에 따라 오류가 발생하므로
  적절한 순서에 따라 수행 해야 함

- 읽기와 쓰기 명령을 거의 동시에 실행해야 한다면, 기본적으로 읽기 명령을 먼저 수행 후 쓰기 명령을 수행하는 접근

2 임계영역

2 임계영역

1 의미

- 프린터처럼 둘 이상의 프로세스가 공유할 수 없는 자원을 임계자원이라 하며 프로그램에서 임계자원을 이용하는 부분

- 공유 메모리가 참조되는 프로그램을 부분으로 다수의 프로세스가 접근 가능한 영역이면서 한 순간에 하나의 프로세스만 사용할 수 있는 영역

2 임계영역

1 의미

- 프로세스들이 공유 데이터를 통해 협력할 때 어떤 프로세스가 임계영역에 들어가면 다른 모든 프로세스는 임계영역으로의 진입이 금지되어야 함

- 임계영역 문제는 프로세스들이 서로 협력하여 자원을 사용할 수 있도록 프로토콜을 설계함으로써 해결할 수 있음

2 임계영역

1 의미

- 각 프로세스는 접근하려는 자원의 임계영역에 들어갈 수 있는지 여부를 미리 요청해야 하는데, 이런 요구를 코드로 구현한 부분을 진입영역이라 함

- 임계영역 다음에는 임계영역에서 수행을 마치고 나갈 프로세스를 선택하는 출구영역이 있음

  ↳ 나머지 코드는 잔류영역이라 하며 임계영역을 마치고 나고 수행함

2 임계영역

1 의미

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

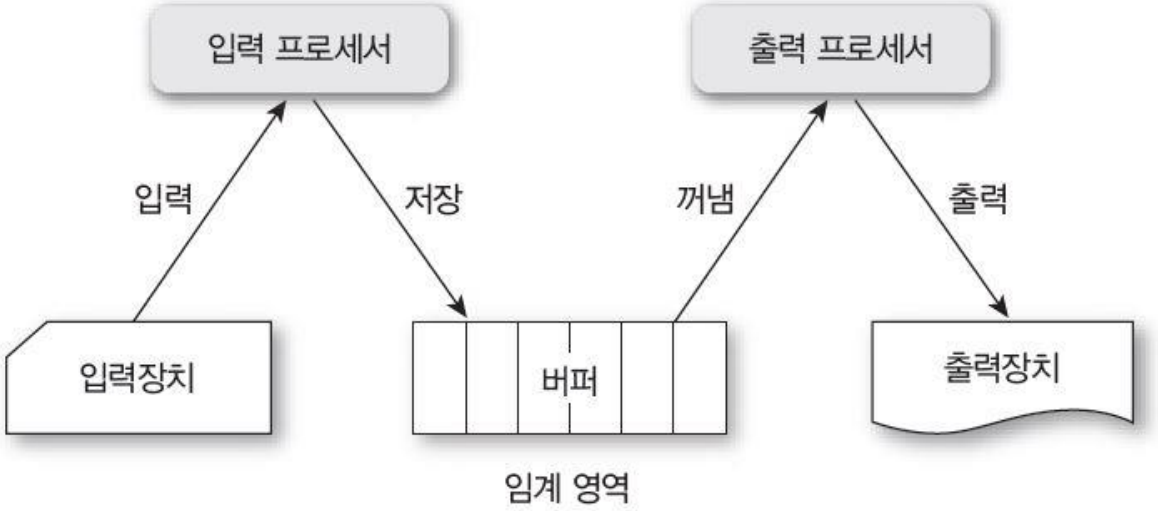
진입 영역

탈출 영역

※ 출처 : 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미

2 임계영역

1 의미



※ 출처 : 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미

2 임계영역의 조건

상호배제

- 프로세스 p1가 임계영역에서 수행 중일 때 다른 프로세스는 임계영역에서 수행할 수 없음

진행

- 임계영역에서 수행하는 프로세스가 없고 여러 개의 프로세스가 임계영역으로 들어가려고 하면 프로세스 선정 알고리즘에 따라 다음 임계영역에서 수행할 대상을 선정
- 다음 임계영역으로 들어갈 프로세스 선택은 무한정 미룰 수 없음

2 임계영역의 조건

제한된 대기

- 한 프로세스가 임계영역을 요청한 후 요청이 수락되기까지 다른 프로세스가 임계영역에 진입할 수 있는 횟수를 제한해야 함

2 임계영역

3 데커알고리즘

두 프로세스가 서로 통신하려고
공유 메모리를 사용하여 충돌 없이 단일 자원을
공유할 수 있도록 허용하는 것

2 임계영역

3 데커알고리즘

예제 4-5 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 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 */;
```

※ 출처 : 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미

2 임계영역

3 데커알고리즘

```
// 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 */;
```

※ 출처 : 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미

3 세마포어

3 세마포어

1 개념

- 다익스트라가 테스 명령어의 문제 해결을 위해 제안
- 상호배제 및 다양한 연산의 순서도 제공
- 세마포 값은 True나 False로, P와 V연산과 관련됨
  - 네덜란드어로 P는 검사, V 증가 의미
  - 음이 아닌 정수 플래그 변수

3 세마포어

1 개념

- 세마포를 의미하는 S는 표준 단위 연산 P(프로세스 대기하게 하는 wait 동작, 임계 영역에 진입하는 연산)와 V(대기 중인 프로세스 깨우려고 신호 보내는 signal 동작, 임계 영역에서 나오는 연산)로만 접근하는 정수 변수

3 세마포어

1 개념

예제 4-9 세마포 정의

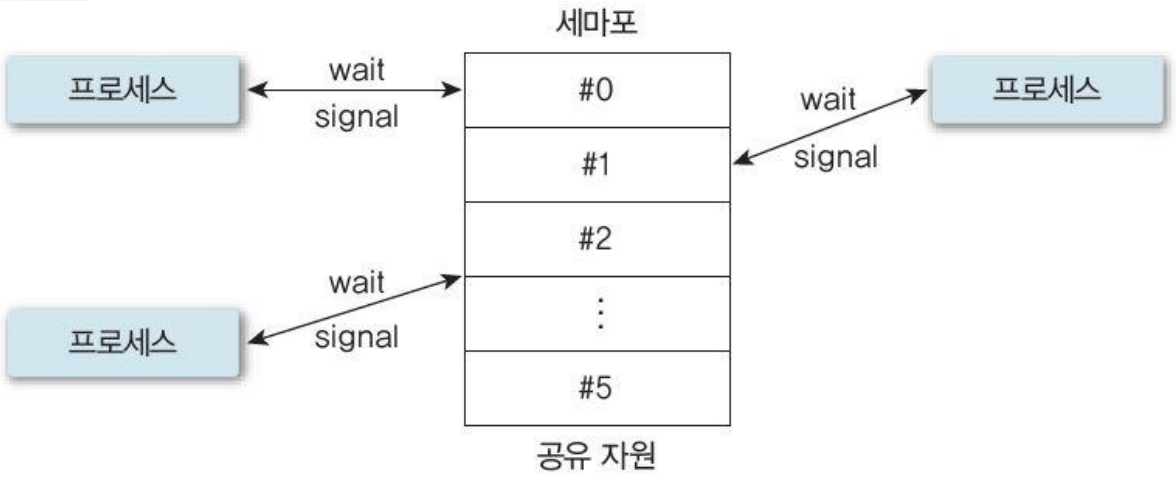
```
P(S) : wait(S) {                    // S = 1로 초기화
          while S <= 0
               ;                    // 바쁜 대기, S > 0 때까지 대기
          S--;
     }

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

※ 출처 : 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미

3 세마포어

2 세마포 공유 자원 접근 관리



※ 출처 : 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미

3 세마포의 프로세스 n개의 임계영역 문제 해결

예제 4-10 프로세스 P_n

```
do {
        wait(mutex);
                // 임계 영역
        signal(mutex);
                // 나머지 영역
} while (1);
```

※ 출처 : 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미

4 세마포 동기화

예제 4-11 세마포 동기화

프로세스 P_1	프로세스 P_2
S_1; signal(synch);	wait(synch); S_2;

※ 출처 : 그림으로 배우는 구조와 원리 운영체제, 구현회, 한빛아카데미