뮤텍스&세마포어.

- 공유된 자원에 여러 프로세스 또는 스레드가 동시 접근하면서 발생하는 문제 (동기화 문제). → 공유된 자원 속 하나의 데이터에는 한 번에 하나의 프로세스만 접근할 수 있도록 제한하기 위해 등장.

(스레드: stack 제외한 메모리를 공유하기 때문에.)

ex) 화장실에 한 번에 1명만 사용할 수 있을 때 여러 사람 들어가는 것을 방지하기 위해 열쇠를 만듦. 열쇠 가진 사람만 사용. X면 대기.

1. 세마포어 : 리소스의 상태를 나타내는 간단한 카운터. (정수 변수).
   - 1개의 공유 자원에 대해 제한된 개수의 프로세스 또는 스레드만 접근 가능하도록. (카운터는 1 이상).
   - 세마포어는 atomic하게 제어되는 정수변수로 일반적으로 값이 0이면 block을하고 0보다 크면 접근함과 동시에 세마포어 값 1 감소. (종료하고 나갈 때도 값을 1 증가.)
     ⇒ 접근되는 자원은 C.S.로 퍼포먼스가 떨어질 수 있어서 주의 필요.
     ⇒ 데드락 가능성 존재.
   - 구현 방법 : Busy-waiting ⇒ 반복문 ∞ → 진입가능할 때 C.S진입.
     ⇒ 반복문 계속 돌 때도 문맥교환이 계속 일어나므로 효율↓ (CPU 점유) 쓸데없이
     ⇒ 어떤 프로세스가 먼저 C.S에 들어갈 지 처리 불가.

코드)

```
wait(S) {   // 사용 대기
    while(S<=0);
    S--;
}
signal(S) {   // 반환.
    S++;
}
```

2. 뮤텍스 : 여러 스레드가 C.S에 동시에 접근하지 못하도록 막는 기법.
- 두 가지 연산만 필요.
① lock : C.S에 들어갈 권한을 얻는다. → 만약 사용중이면 대기
② unlock : C.S 사용 끝났음을 알린다. (사용한 스레드가 반납).
- 뮤텍스를 위한 세 가지 알고리즘 존재.
① 데커
② 피터슨 ?
③ 제과점.

3. 뮤텍스, 세마포어 차이점
- 가장 큰 차이점은 관리하는 동기화 대상의 갯수.
=> 뮤텍스는 동기화 대상이 오직 하나, 세마포어는 하나 이상일 때 사용.
- 세마포어는 뮤텍스가 될 수 있고, 뮤~~X~~세
- 뮤텍스는 lock을 소유 가능 (반납 포함 소유주가), 세마포어는 소유할 수 없다.
- 세마포어는 시스템 범위에 걸쳐있고 파일형태로 존재,
뮤텍스는 프로세스 범위 가지며, 프로세스 종료 시 clean up.