Mutex & Semaphore

# Mutex : local

* 쓰레드의 동작을 동기화하기 위한 도구
* 다른 쓰레드가 어떤 공유 변수를 수정하는 동안에는 그 변수에 접근하지 않도록, 쓰레드의 공유 자원 사용을 동기화할 수 있다.

# Mutex 사용 함수

* Pthread\_mutex\_init(pthread\_mutex\_t \*mutex, const pthread\_mutex\_attr \*attr)
  + Mutex 객체 초기화
  + Parameter
    - Pthread\_mutx\_t \*mutex : mutex 객체
    - Pthread\_mutex\_attr \*attr : 뮤텍스의 특징 (default : NULL)
  + Return
    - 성공적으로 수행시 0, 실패했을시 -1 반환
* Pthread\_mutex\_destroy(pthread\_mutex\_t \*mutex)
  + Mutex 객체 소멸 / Mutex가 반드시 unlock상태여야 한다.
  + Parameter
    - Pthread\_mutex\_t \*mutex: mutex 객체
  + Return
    - 성공적으로 수행시 0, 실패했을시 -1 반환
* Pthread\_mutex\_lock(pthread\_mutex\_t \*mutex)
  + Critical section 들어갈 때 mutex lock 요청
* Pthread\_mutex\_unlock(pthread\_mutex\_t \*mutex)
  + Critical section에서 모든 작업을 마치고 mutex lock을 돌려주기 위해 사용

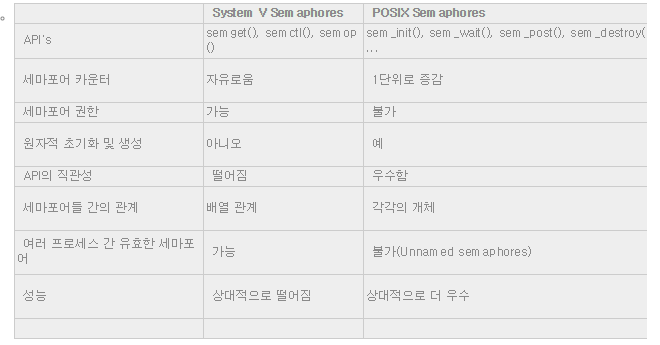
# Semaphore : 커널 객체

* 지정된 수만큼의 쓰레드가 동시에 실행되도록 동기화하는 것이 가능

System V Semaphore VS. POSIX Semaphore

차이점:

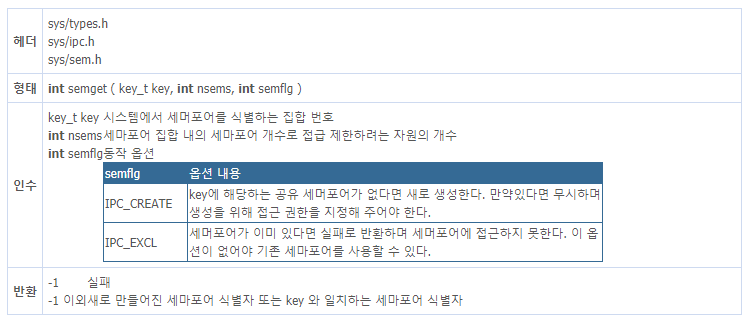
* System V는 세마포어 카운트를 얼마나 할 것인지 조절 가능 / POSIX는 1개씩
* System V는 세마포어 권한 변경 가능 / POSIX 권한 조작X
* 사용자 관점에서 System V는 어색 / POSIX는 간단
* 확장성 : POSIX > System V 🡪 Server/Client 시나리오에서는 POSIX 사용이 더 좋다.
* System V는 세마포어 배열 생성 / POSIX는 하나만 생성 🡪 System V 메모리 사용 무겁다
* POSIX 세마포어는 프로세스 차원 세마포어에 대한 메커니즘 제공 / 비영구적 세마포어



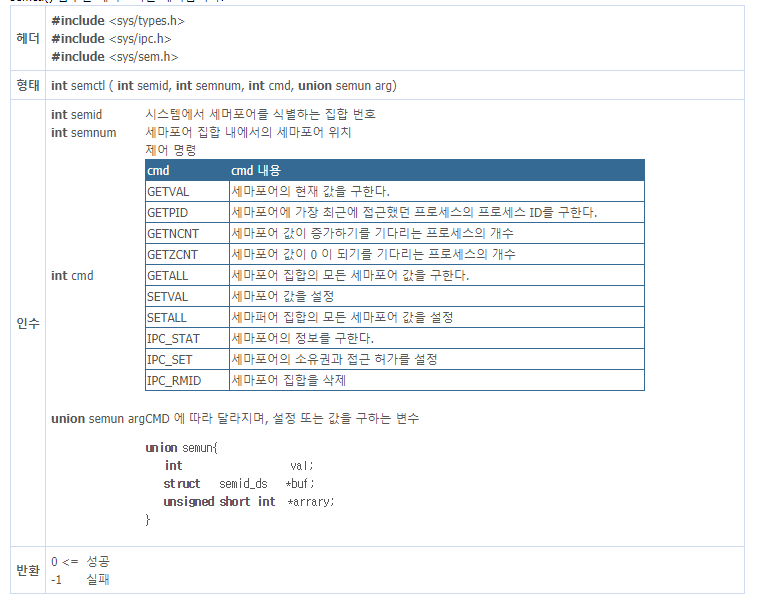
출처 : <https://illos.tistory.com/148>

# System V Semaphore

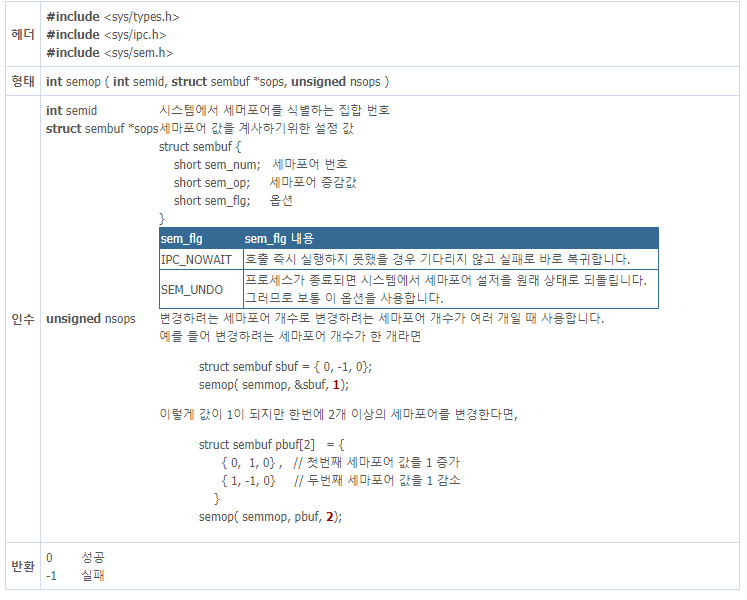
* Semget(key\_t key, int nsems, int semflg)



* Semctl(int semid, int semnum, int cmd, union semun arg)



* Semop(int semid, struct sembuf\* sops, unsigned nsops)



# POSIX Semaphore

* Int sem\_open(const char \*name, int oflag, …)
  + Named Semaphore 객체 초기화를 위해 사용
  + Parameter
    - Name : 세마포어 객체 이름
    - Oflag : 세마포어 생성 및 접근 가능 여부 제어
    - Etc : named semaphore 는 파일 형태로 저장되기 때문에 접근권한과 count
  + Return
    - 성공적으로 수행되면 세마포어 주소 반환 / 실패시 SEM\_FAILED 반환
* Int sem\_close(sem\_t \*sem)
  + Named 세마포어 종료 지시
  + Parameter
    - Sem : 세마포어 객체
  + Return
    - 성공적으로 수행되면 0 반환 / 에러 발생시 오류 값 반환
* Int sem\_getvalue(sem\_t \* restrict sem, int \* restrict sval)
  + 세마포어의 상태값을 sval에 저장
  + Parameter
    - Sem : 세마포어 객체
    - Sval : 세마포어 값을 저장하는 메모리 위치
  + Return
    - 성공적으로 수행되면 0 반환 / 에러 발생시 오류 값 반환
* Int sem\_post(sem\_t \*sem)
  + 세마포어 잠금 해제
  + 세마포어 값 증가
  + Parameter
    - Sem : 세마포어 객체
  + Return
    - 성공적으로 수행되면 0 반환 / 그렇지 않다면 오류 값 반환
* Int sem\_wait(sem\_t \* restrict sem)
  + 세마포어 잠금
  + 세마포어 값 감소
  + Parameter
    - Sem : 세마포어 객체
  + Return
    - 성공적으로 수행되면 0 반환 / 에러 발생시 오류 값 반환
* Int sem\_unlink(const char \*name)
  + 이름을 가진 세마포어 제거
  + Parameter
    - Name : 제거해야할 세마포어 이름
  + Return
    - 성공적으로 수행되면 0 반환 / 에러 발생시 오류 값 반환