이벤트 : 프로그램에 의해 감지되고 처리될 수 있는 동작이나 사건

* 프로그램 동작 과정과 함께 동시에 처리되도록 되어 진다.
* 리눅스에는 이벤트 기능이 존재하지 않으므로 직접 구현해야 한다.

Windows 이벤트 구현 함수

* CreateEvent(LPSECURITY\_ATTRIBUTES IpEventAttributes, BOOL bManualReset, BOOL bInitialState, LPTSTR IpName);
  + 이벤트 생성 함수
  + BOOL bManualReset: 수동(TRUE), 자동(FALSE)
    - 수동 : SetEvent 이벤트 발생시 ResetEvent를 해줘야 이벤트 다시 발생
    - 자동 : ResetEvent를 안해도 이벤트 다시 발생
  + BOOL bInitialState: TRUE(처음부터 이벤트 발생 상태), FALSE(이벤트 대기)
  + LPTSTR IpName: 이벤트 네임 부여
* SetEvent(hEvent);
  + hEvent 발생
* ResetEvent(hEvent);
  + hEvent 재생(수동일 경우만)
* WaitForSingleObject(), WaitForMultiObject()
  + 어느시간 동안 대기 상태
* CloseHandle()
  + 이벤트 핸들러 종료
* Windows에 존재하는 CreateEvent, SetEvent, ResetEvent, WaitForSingleObject, CloseHandle 함수를 pthread.h 헤더 사용하여 구현

이벤트 라이브러리 구현에 사용한 변수

* Sem\_ t

이벤트 라이브러리 구현에 사용한 함수

* 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\_timedwait(sem\_t \* restrict sem, construct timespec \*restrict abstime)
  + 세마포어를 정해진 시간만큼 기다렸다가 타임아웃이 끝날 때 종료
  + Parameter
    - Sem : 세마포어 객체
    - Abstime : CLOCK\_REALTIME 클럭에 기반한 시간
  + 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