운영체제의 기초 Project4

전기정보공학부

2017-17088 박찬정

1. 구현한 구조체들

\* eos\_semaphore\_t

본 구조체는 count, wait\_queue, queue\_type 의 세 가지 멤버 변수가 있다. Count는 배정할 수 있는 semaphore의 수, wait\_queue는 본 semaphore을 얻기 위해 기다리는 task의 list, queue\_type은 task의 wait\_queue를 관리하는 방식에 대한 값이다.

\* eos\_mqueue\_t

프로젝트 PPT에서 제시된 대로 queue\_size, msg\_size, queue\_start, front, rear, queue\_type, putsem, getsem 의 멤버 변수가 있고, 여기에 추가적으로 queue의 마지막 주소를 저장하는 queue\_end 변수도 추가하였다.

2. 구현한 함수들에 대한 설명

\* eos\_init\_semaphore()

인자로 받아오는 sem에 대하여 count, queue\_type 값을 저장하고, wait\_queue를 NULL로 초기화한다.

\* eos\_acquire\_semaphore()

우선 sem의 count를 확인한다. 0보다 큰 경우 semaphore를 제공하고 끝난다. count가 0 이하인 경우 semaphore를 배정할 수 없는 경우이고, 이 때 timeout 값에 따라 다르게 행동한다. -1인 경우 바로 0을 리턴한다(실패). Timeout 값이 0인 경우 계속 기다리는 경우이다. 현재 task를 semaphore의 wait queue에 넣고 eos\_schedule을 호출하여 다른 task가 실행되도록 한다. 이후 semaphore가 반환되어 본 task가 ready queue에 들어가면 이후 코드가 계속 실행된다. 이 때 다시 한 번 sem의 count를 확인하고 그에 따라 행동한다. Timeout 값이 0보다 큰 경우 마찬가지로 semaphore의 wait queue에 넣고, eos\_set\_alarm을 통해 timeout 값 만큼의 tick이 지난 후 본 task가 다시 ready queue로 돌아갈 수 있게 한다. 이제 eos\_schedule을 호출하여 다른 task가 실행되도록 하고, 이후 설정한 alarm이 해결되면 본 task가 ready queue에 들어가고 이후 코드가 계속 실행된다.

\* eos\_release\_semaphore()

sem의 count를 1 증가시키고, sem의 wait queue가 빈 경우 그대로 끝난다. Wait queue에 task가 존재하는 경우 해당 task를 ready queue에 넣어 멈춘 지점에서 다시 실행될 수 있게 한다.

\* eos\_init\_mqueue()

eos\_mqueue\_t 의 모든 멤버 변수(queue\_size, msg\_size, queue\_start, queue\_end, front, rear, queue\_type, putsem, getsem)를 인자로 받은 값을 사용하여 초기화한다.

\* eos\_send\_message()

받은 메시지 큐의 입력 semaphore를 받아온다. 그리고 메시지 큐에 인자로 받은 메시지를 저장한다. Queue의 구현 방식상 mq의 rear가 메시지 큐의 끝 주소를 넘어가는지 확인하여 넘는 경우 mq의 rear 값을 메시지 큐의 첫 주소 – 1 으로 바꾼다.

\* eos\_receive\_message()

받은 메시지 큐의 출력 semaphore를 받아온다. 그리고 메시지 큐에 저장된 값을 인자로 받은 주소에 저장한다. Queue의 구현 방식상 mq의 front가 메시지 큐의 끝 주소를 넘어가는지 확인하여 넘는 경우 mq의 front 값을 메시지 큐의 첫 주소로 바꾼다.

3. 테스트 프로그램 수행 결과

텍스트이(가) 표시된 사진

자동 생성된 설명