실험.실습 보고서

(3) 반

조원: (학번:12131489 ,이름:김영훈)

1. 제목

7주차 실습과제 - Event flag와 Semaphore 사용하기

2. 목적

Event flag 사용하여 실행 되어야할 task의 순서를 맞춘다. Semaphore를 활용하여 critical section을 보호한다.

3. 실습에 필요한 기초지식

Event flag와 Semaphore의 내부 구조에 대해서 알면 이해하기 수월하다. Semaphore를 사용할 때는 critical section이 왜 발생하면 안되는지에 대한 이해가 필요하고, 잘 활용할 줄 알아야한다. Event flag의 동작 원리를 잘 알고 있어야 어떨 때 wait이 풀리는지 이해할 수 있다.

4. 실습 절차, 내용 및 결과

main 에서는 TaskStart task 를 생성한다. taskStart task 는 TaskStartCreateTasks() 함수를 호출 시킨다. 이 함수에서 event flag 2 개와 Semaphore 1 개를 생성한다. 그리고 RandomTask 4 개, DecisionTask 1 개를 만들어 실행한다.

RandomTask 숫자 생성 및 전달, 출력

이 task 중에서 DecisionTask 가 먼저 실행되어도 s_grp event flag 가 0xff 이 될 때까지 Pend 로 wait 상태에 빠지게 된다(s_grp 을 Post 시켜주는 것은 RandomTask 에 있고 각 task 마다 bit 한 자리씩 1 로 만든다) RandomTask 가 실행될 경우 random 한 숫자를 생성시켜 send_buffer 에 넣는다. 그리고 자신의 task 번호에 해당되는 bit 를 s_grp 에 Post 해준다. 그리고 r_grp event flag 에 자신이 해당되는 bit 가 1 이 될 때 까지 Pend 하여 wait 상태로 빠진다.

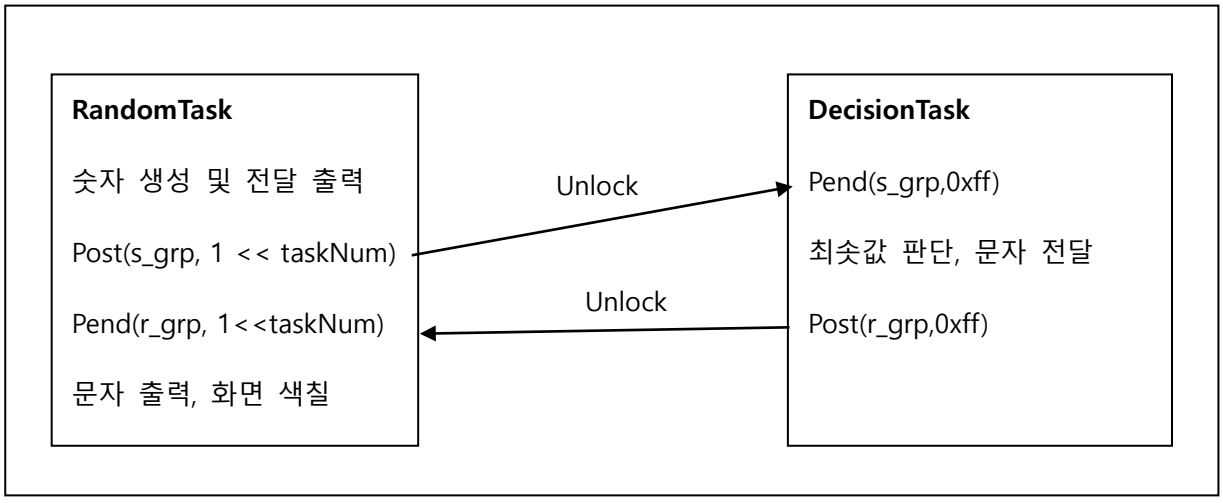
DecisionTask 는 각 task 로부터 받은 총 4 개의 s_grp Post 를 통해 flag 를 0xff 에서 0x00 으로 만들 수 있다. 이 때 wait 상태는 풀리게 되며 다음 코드를 실행할 수 있게 된다.

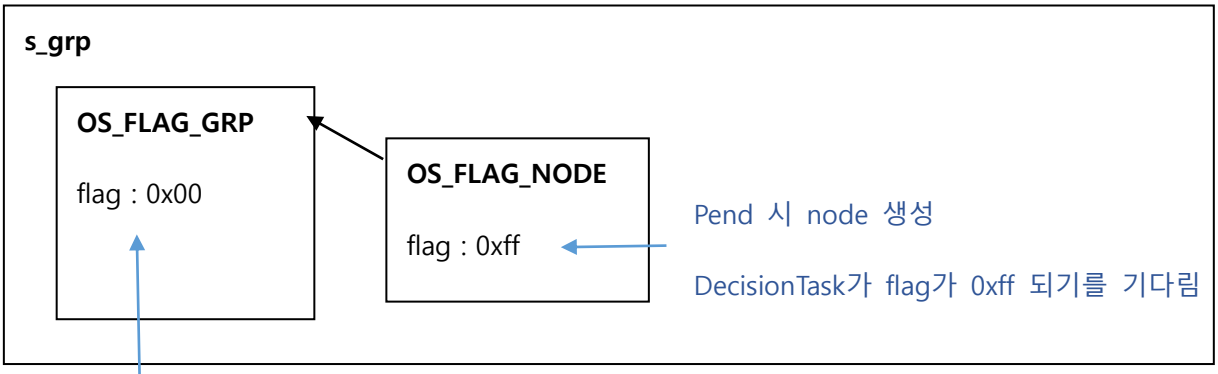
DecisionTask 최솟값 판단, 문자 전달

각 RandomTask 가 send_buffer 에 채운 숫자들을 비교하여 최소값을 찾는다. 최소값을 갖는 task 에게는 'W', 나머지는 'L'을 receive_buffer index 번호를 이용해 구별하여 저장한다. 이 작업을 마쳤으면 r_grp 을 Post 로 0xff 값을 줘서 모든 RandomTask 가 r_grp Pend 가 풀리도록 할 수 있다. 그리고 DecisionTask 는 for 문이 반복되어 처음으로 돌아가 다시 s_grp 의 flag 가 0xff 가 되도록 기다린다.

RandomTask 문자 출력, 화면 색칠

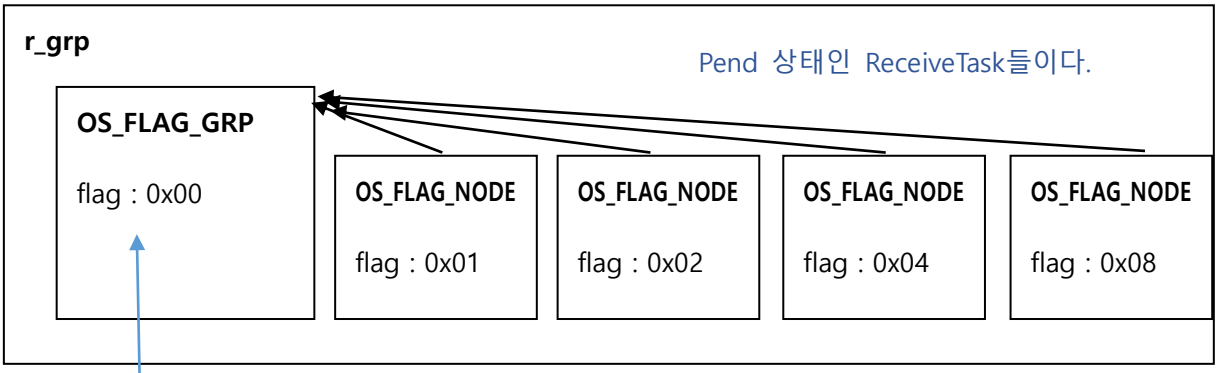
DecisionTask 의 r_grp Post 로 실행된 task 들은 각 알맞은 화면의 위치에 문자를 출력한다. 문자가 'W'인 task 는 화면에 task 에 해당되는 색깔을 출력한다. 각 task 들은 잠시 쉬고 다시 for 문을 반복하여 처음으로 돌아간다.





Post(s_grp, 1 << taskNum)

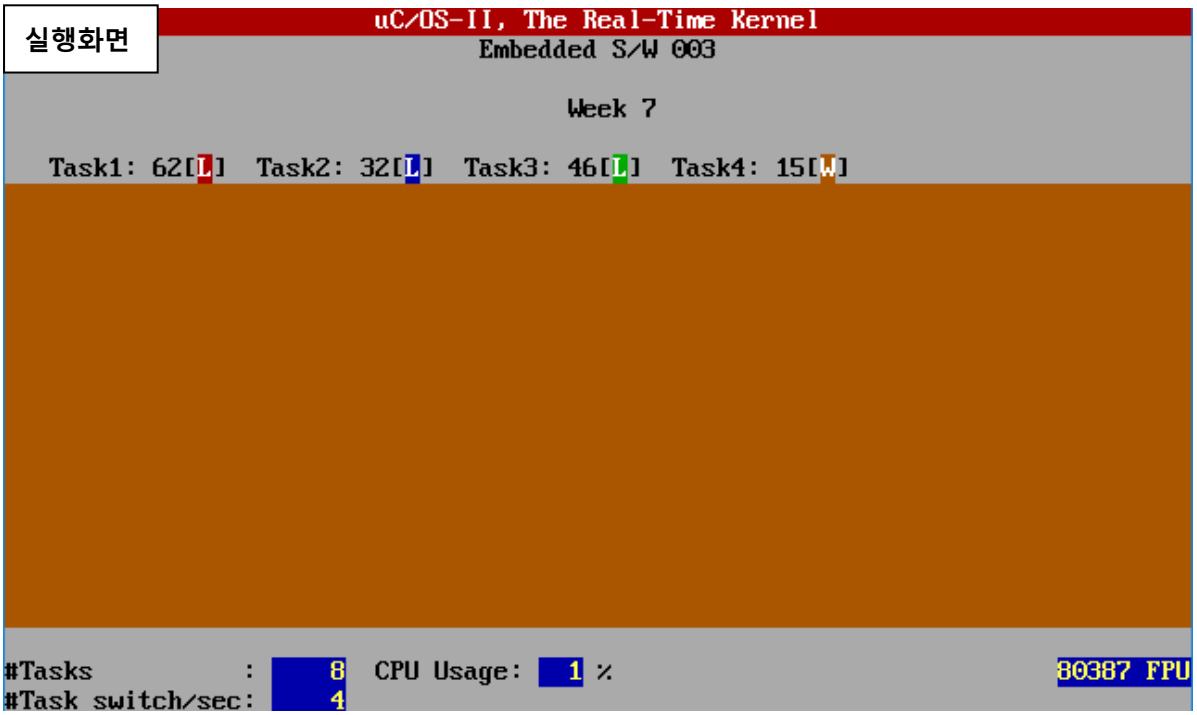
ReceiveTask가 Post할 때 하위 4비트 중 자신의 task에 해당되는 bit를 1로 만듬



Post(r_grp, 0xff)

DecisionTask가 Post하면 flag를 0xff로 만들어 각 대기하고 있는 ReceiveTask를 실행상태로 만든다.

실행화면



5. 결론

이번 실습을 통해 Semaphore와 event flag를 활용해보았다. event flag는 task간의 실행 순서 대해 동기화 시키는데 굉장히 유용하게 쓰였다. 이를 통해 이전 과제에서 이용한 message queue, mailbox가 필요 없이 전역배열을 이용하여 동기화를 할 수 있었다. 또한 flag를 사용자가 설정함으로써 다양하게 쓰일 수 있는 장점이 있었다.