실험.실습 보고서

(3) 반

조원: (학번:12131489 ,이름:김영훈)

1. 제목

8주차 실습과제 - Mutex semaphore 사용하기

2. 목적

Mutex semaphore를 사용하여 semaphore에서 발생할 수 있는 우선순위 전도현상(Priority inversion)을 해결한다.

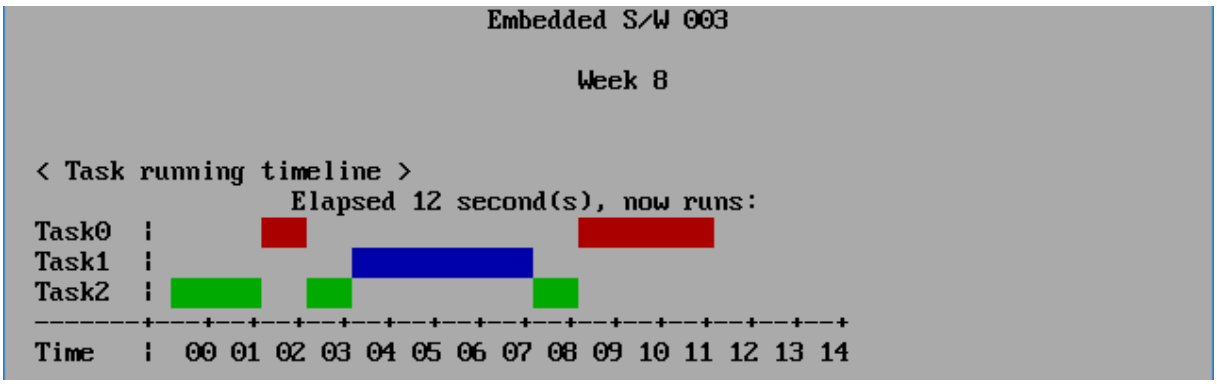
3. 실습에 필요한 기초지식

우선순위 전도현상이 무엇이고 언제 일어나는지 알아야한다. Mutex semaphore에서는 이 문제를 어떻게 해결하는지 알기 위해 PIP(Priority inheritance protocol)에 대한 지식이 필요하다.

4. 실습 절차, 내용 및 결과

Task 0, 1, 2 가 있다. 각 task 의 우선순위는 4, 6, 8 이고 task 0, 2 는 run 함수를 실행시키는데 Mutex 자원을 이용하여 동기화(synchronization)하였다. task0 은 2 초 지연 후 run 함수가 실행되고, task1 은 4 초 지연 후 run 함수가 실행되고, task2 는 지연없이 run 함수가 실행된다.

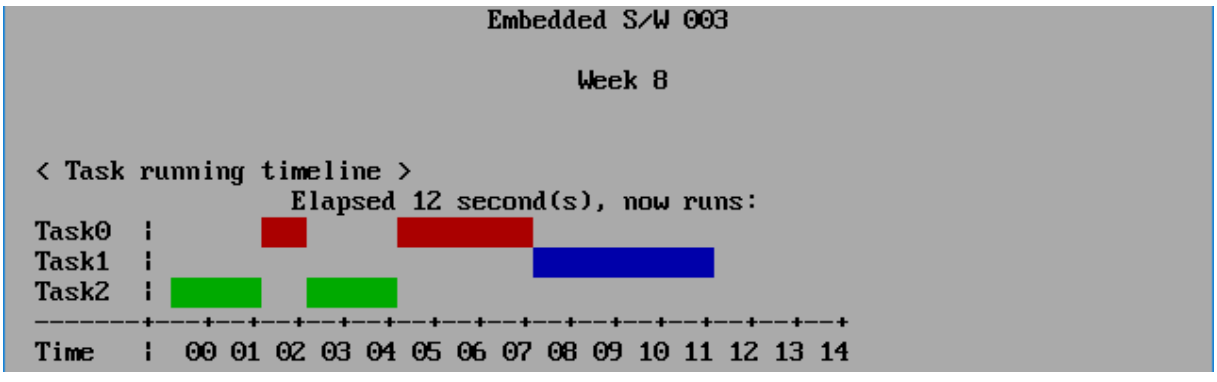
semaphore 사용시



위 실행화면은 mutex 가 아닌 semaphore 를 사용하였을 경우이다. 처음 task2 가 실행되어 run 함수를 2 초간 실행하고, semaphore 자원을 가져간다. 이 때 task0 이 들어오는데 run 함수를 한번 실행시키고, 다음 run 함수부터는 semaphore 자원을 task2 가 가지고 있기 때문에 wait 상태에 빠진다. 다시 task2 가 실행되고, 4 초가 되었을 때 task2 보다 우선순위가 높은 task1 이 실행된다. task1 이 모든 run 함수를 실행시킨 뒤 종료가 되면 우선순위가 더 낮았던 task2 가 다시 실행된다. task2 가 실행을 끝낸 뒤 semaphore post 로 자원을 풀어주고 task0 은 unlock 되어 ready 상태가 되어 실행된다.

이 실행결과를 통해 task0 은 우선순위가 높음에도 lock 이 걸려 무관한 task1 의 수행시간까지 기다리고 있는 문제점을 발견할 수 있었다. 이러한 우선순위가 역전되는 현상을 우선순위 전도현상(Priority inversion)라고 한다.

mutex semaphore 사용시



이 문제를 해결하기 위해 mutex semaphore 를 사용하면, task2 가 pend 로 mutex 자원을 가져간 뒤, task0 이 실행되어 pend 로 자원에 접근할 때 task0 의 우선순위를 task2 에게 넘겨줄 수 있다. 이 방법을 PIP(Priority inheritance protocol)라고 한다. microC/OS-II 에서는 같은 우선순위를 부여할 수 없으므로, mutex 생성 시 계승하고자 하는 우선순위보다 높은 우선순위 3 을 부여했다.

이 방법을 통해 task0 이 task2 에게 우선순위를 계승하고 wait 상태에 빠져든다. 그러면 task2 의 우선순위는 미리 정해 놓은 PIP 숫자 3 으로 우선순위가 상승한다. 따라서 task1 이 ready 상태가 되어도 우선순위가 task2 가 더 높기 때문에 task2 가 실행된다. task2 가 실행이 마치면 mutex 는 post 되어 task0 이 가져가고 ready 상태가 된다. 우선순위가 가장 높은 task0 은 실행되고 종료될 때, ready 상태의 task1 가 실행된다.

5. 결론

이번 실습을 통해 우선순위 전도현상을 해결해야하는 필요성을 느꼈다. semaphore 사용할 때 만약 task1의 수행시간이 상당히 길었더라면, task0은 task1이 끝나고 task2가 lock을 풀어줄 때까지 긴 시간을 기다려야 한다. 이것은 hard real time system에서는 정말 치명적인 문제였을 것이다. 이 문제를 mutex를 이용하여 우선순위 전도현상을 간단히 해결하였다. mutex는 우선순위 계승 프로토콜(PIP)을 할 수 있도록 내부 구현이 되어있어 굉장히 유용하게 쓸 수 있었다. 단, 우선순위 상한 프로토콜(PCP / priority ceiling protocol)은 구현이 되어있지 않기 때문에 이행 블로킹(transitive blocking), 데드락(deadlock)이 발생할 수 있는 문제점은 여전히 남아있었다.