1. 사용한 자료구조

Vector :

최종 스케쥴링된 태스크들을 담는 scheduling, 주기적 태스크 및 비주기적 태스크들을 담고 있는 tt, tc, at, ac에 사용하였다. 벡터는 적은 양의 자료에 유리하고 크기 변경 및 순차 접근이 가능하므로 벡터를 사용하였다. 또한 iterator를 이용하여 find 함수를 이용해 탐색할 수 있으므로 매 시간마다 시간과 주기가 일치하는 태스크를 발견할 때 사용하였다.

Queue

우선순위가 높은 태스크들이 시간과 주기가 일치하여 실행되어야할 때, 또한 polling에서는 polling 주기가 되어 aperiodic task를 실행하여야 할 때, 해당 태스크들을 저장하고 실행하기 위해 queue를 사용하였다. Queue를 사용한 이유는 우선순위가 높은 순서대로 먼저 삽입되고 또한 우선순위가 높을수록 먼저 pop되어야하기 때문에 FIFO의 queue로 구현하였다.

// 전반적인 코드의 흐름을 document에서 설명하고, 코드 각각에 대한 내용은 주석으로 처리하였습니다.

1. Background APS algorithm 코드 구현 설명

먼저, 입력받은 태스크들을 각 변수 tt, tc, at, ac에 초기화시키고, 다른 변수들 또한 초기화한다.

Hyper period를 LCM을 이용하여 계산한 뒤, time이 hyperperiod 보다 작을 때까지 루프를 돌린다.

먼저, 각 태스크들이 deadline을 넘기지 않았는지 확인하기 위해 태스크의 주기와 주기가 한바퀴 돌았을 때 false를 나타내는 boolean 배열을 이용하여 매 시간마다 확인한다.

Deadline을 넘기지 않았다면, 시간과 주기가 일치하는 태스크가 있는지 확인하고, 존재하는 경우 현재 task의 index를 saveIdx에 저장한 뒤, idx에는 preem 태스크의 index를 저장한다. 만약 주기가 일치하는 태스크가 하나가 아닌 여러 개가 존재하는 경우에는 우선순위가 높은 순서대로 queue에 삽입한다.

cnt변수는 computation time을 체크하기 위한 변수인데, 만약 computation time과 cnt가 일치한다면 다음 태스크로 넘어가고(idx++), 또한 que에 preem 태스크들이 존재하는 경우에는 que 에 저장된 태스크들을 먼저 실행시킨다.

Idx가 태스크의 개수를 넘어 선 경우에는 aperiodic task를 실행한다. 현재 시간까지 도착한 aperiodic task가 존재하는 경우 que에 저장하고 실행시킨다. 방문한 aperiodic task는 1로 방문했다고 처리해준다.

Aperiodic task가 존재하는 경우에는 aperiodic task를 스케쥴링해주고, 아닌 경우에는 idle을 나타내기 위해 -9999를 스케쥴링 벡터에 삽입한다. 이 때, 만약 aperiodic task가 스케쥴링 된 경우 실행 시간을 ap\_exec에 저장해 이후 average waiting time을 구할 때 사용한다.

Average waiting time은 앞서 저장한 aperiodic task가 실행되었을 때의 시간과 도착했을 때의 시간 차를 task의 개수로 나눈 값으로 한다.

1. Polling APS algorithm 코드 구현 설명

전반적인 내용은 background와 동일하지만, polling 주기마다 polling을 실행하여 그 시간까지 도착한 aperiodic task가 있는 경우, polling server의 capacity 만큼 실행시켜주고 다시 기존의 프로세스를 계속 진행하도록 구현해주었다.