과제3 결과보고서

IEEE Code of Ethics

(출처: http://www.ieee.org)

We, the members of the IEEE, in recognition of the importance of our technologies in affecting the quality of life throughout the world, and in accepting a personal obligation to our profession, its members and the communities we serve, do hereby commit ourselves to the highest ethical and professional conduct and agree:

1. to accept responsibility in making decisions consistent with the safety, health and welfare of the public, and to disclose promptly factors that might endanger the public or the environment;
2. to avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist;
3. to be honest and realistic in stating claims or estimates based on available data;
4. to reject bribery in all its forms;
5. to improve the understanding of technology, its appropriate application, and potential consequences;
6. to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations;
7. to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;
8. to treat fairly all persons regardless of such factors as race, religion, gender, disability, age, or national origin;
9. to avoid injuring others, their property, reputation, or employment by false or malicious action;
10. to assist colleagues and co-workers in their professional development and to support them in following this code of ethics.

위 IEEE 윤리헌장 정신에 입각하여 report를 작성하였음을 서약합니다.

학 부: 전자공학과
제출일: 2021. 11. 28.
과목명: 전자공학운영체계
교수명:
분 반:
성 명: 윤건

전자공학운영체계 과제3 결과보고서

201721045 윤건

□ 소스 코드

○ os_core.c

```
static  void  OS_SchedNew(void) {

    INT8U mostUrgentPriority = OS_LOWEST_PRIO;
    INT32S mostUrgentDeadline = 0x7FFFFFFF;

    for (INT8U i = 0; i <= OS_LOWEST_PRIO - 1; i++) {

        if ((OSTCBPrioTbl[i] != (OS_TCB*)0) && (OSTCBPrioTbl[i] != OS_TCB_RESERVED)) {

            OS_TCB* current = OSTCBPrioTbl[i];

            // Modification here

            // Task ready check
            // Meaning that there is no predelay for that task in table
            if ( current->OSTCBDly == 0 ) {

                // Timeout processing (optional)
                // if : the deadline has passed cueent time tick
                //  This  must  happen  first  to  prevent  false  change  of  most  urgent
deadline
                if ( current->deadline < (INT32S)OSTime ) {
                    // Using the same processing given at pdf page 19, the deadline
update logic.
                    current->deadline += current->period;
                    fprintf(stderr, "%u\t%-17s%u%-27s\n", OSTime - 1, "Deadline fail :
", current->OSTCBPrio, " -> extend deadline to next period");
                    continue;
                }

                // select a task with the closest deadline
                // if : currnet task in table has faster deadline than any before =>
than this is the most urgent
                if ( current->deadline < mostUrgentDeadline ) {
                    mostUrgentPriority  =  current->OSTCBPrio;      //  this  task  is  the
```

```
most urgent
                    mostUrgentDeadline = current->deadline;      // and this task has
the fastest deadline
                }
            }
            ////////////////////////////////////////////////
        }
    }
    OSPrioHighRdy = mostUrgentPriority;                          // validate the most
urgent priority
}
```

주어진 함수에서 우선 OS TCB Dly가 0인 것을 확인하여 쌓여있는 Pre-delay가 있는지 확인하고 진행한다. 즉 실행 가능한 상태인지 확인한다.

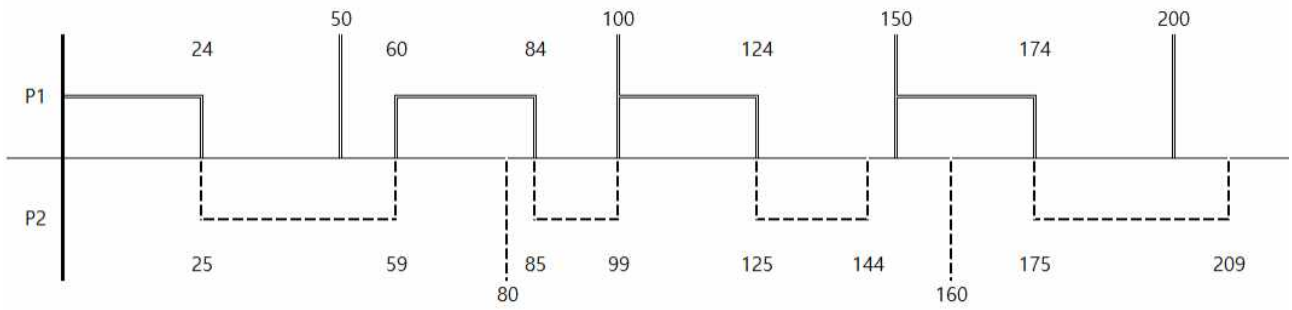
다음으로 테이블 안의 모든 task에 대해 가장 빠른 deadline을 찾는다. 이는 loop을 돌고 있으므로 변수를 선언하여 각 loop에서 더 작은 deadline을 비교하고 업데이트하는 것으로 찾을 수 있다. 제일 빠른 deadline을 가지는 Priority 또한 지역변수에 저장한다.

loop 이후에 전역변수 OS Priority Highest Ready에 찾은 가장 빠른 priority를 업데이트한다. 이는 이미 주어진 함수의 일부분이다.

Timeout이 발생한 것은 현재 OS의 Time tick과 주어진 task의 deadline을 비교하는 것으로 판별할 수 있다. 이를 처리하는 것은 강의노트의 19페이지에 deadline을 처리하는 방식, 현재 deadline에 task의 주기만큼 더해 연장하는 방식을 차용하였다. 이로서 Timeout 되었을 경우 이와 관련된 메시지를 출력하고 한 번의 주기적 실행을 건너뛴다. 이는 deadline을 변동하므로 가장 빠른 deadline을 찾는 알고리즘 이전에 실행되어야 한다.

□ 실행 결과

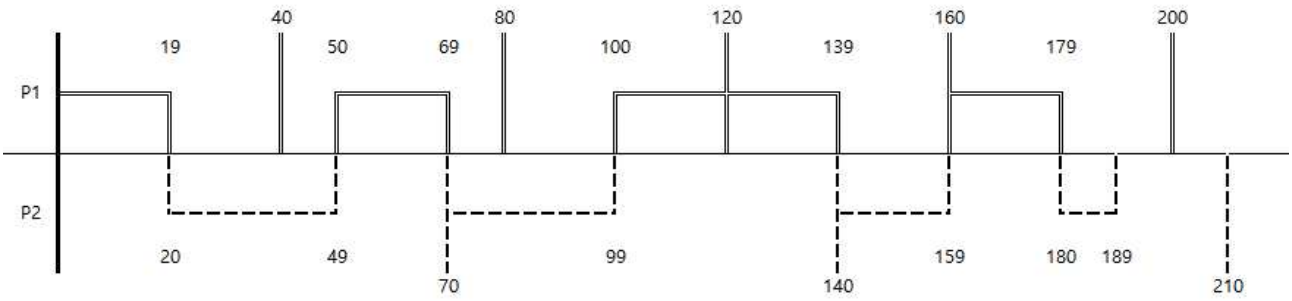
○ 기존 주어진 task set에 대해 예상되는 결과를 확인할 수 있다.




```
D:₩Documents₩아주대학교₩전공₩2021-2 [3-2]₩전자공학운영체계₩HW3₩EDF_scheduler_homework₩Evalboards₩Microsoft₩Windows₩OS2₩Visual Studio₩Debug₩OS2.exe
OSTick    created, Thread ID 33404
Task[ 63] created, Thread ID 23412
Task[  1] created, Thread ID 23512
Task[  2] created, Thread ID  6172
Task[  1] '?' Running
24      Complete     1   2
Task[  2] '?' Running
59      Complete     2   1
84      Complete     1   2
99      Preempt      2   1
124     Complete     1   2
144     Complete     2  63
Task[ 63] 'uC/OS-II Idle' Running
149     Preempt     63   1
174     Complete     1   2
209     Complete     2   1
```

○ 또 다른 성공 예시. 이 또한 예상한 결과를 얻을 수 있었다.

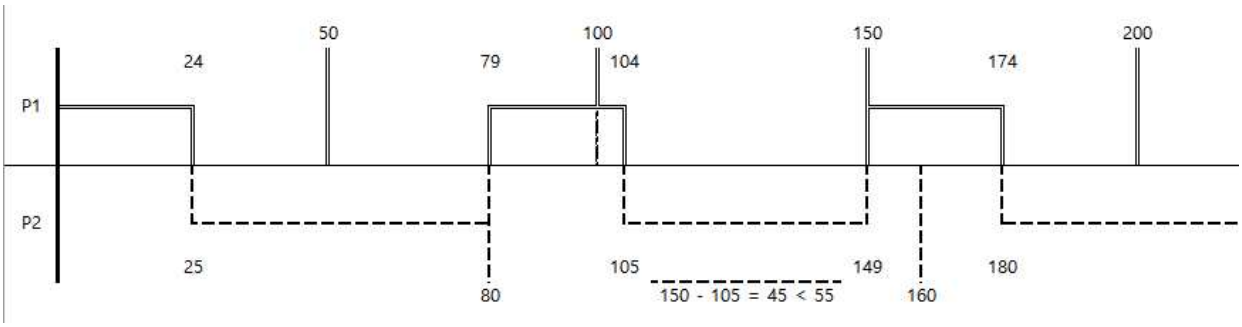
```
69      INT32S limits[][3] = {
70              { 0, 0, 0 },//Prio0
71              { 20, 40, 0 },//Prio1
72              { 30, 70, 0 },//Prio2
73      };
```




```
D:₩Documents₩아주대학교₩전공₩2021-2 [3-2]₩전자공학운영체계₩HW3₩EDF_scheduler_homework₩Evalboards₩Microsoft₩Windows₩OS2₩Visual Studio₩Debug₩OS2.exe
OSTick    created, Thread ID 13008
Task[ 63] created, Thread ID  1444
Task[  1] created, Thread ID 21216
Task[  2] created, Thread ID 35232
Task[  1] '?' Running
19      Complete     1   2
Task[  2] '?' Running
49      Complete     2   1
69      Complete     1   2
79      Preempt      2   1
99      Complete     1   2
119     Complete     2   1
139     Complete     1   2
159     Preempt      2   1
179     Complete     1   2
189     Complete     2  63
Task[ 63] 'uC/OS-II Idle' Running
```

○ 실패 예시

```
69    INT32S limits[][3] = {
70            { 0, 0, 0 },//Prio0
71            { 25, 50, 0 },//Prio1
72            { 55, 80, 0 },//Prio2
73    };
```



Tick 100에서 P1의 Timeout이, Tick 160에서 P2의 Timeout이 일어남을 예상할 수 있다.

```
D:₩Documents₩아주대학교₩전공₩2021-2 [3-2]₩전자공학운영체계₩HW3₩EDF_scheduler_homework₩Evalboards₩Microsoft₩Windows₩OS2₩Visual Studio₩Debug₩OS2.exe
OSTick    created, Thread ID 34732
Task[ 63] created, Thread ID 23324
Task[  1] created, Thread ID 29544
Task[  2] created, Thread ID 33692
Task[  1] '?' Running
24      Complete     1   2
Task[  2] '?' Running
79      Complete     2   1
100     Deadline fail :  1 -> extend deadline to next period
100     Preempt      1  63
Task[ 63] 'uC/OS-II Idle' Running
101     Preempt     63   1
105     Complete     1  63
149     Preempt     63   1
159     Preempt      1   2
160     Deadline fail :  2 -> extend deadline to next period
160     Preempt      2   1
175     Complete     1   2
```

예상한 결과를 얻을 수 있었다.