Ubinos model

- 소개:
  - 본 model은 OSEK/VDX OS기반으로 확장하여 개발하였다. Ubinos책에서(사물 점퓨팅 이론 및 실습) 있는 API기빈으로 최대한 mapping하였다.
  - model 파일 list:
    - config.h
    - ubinos.h ----------→ 기본 상수 및 자료구조 정의
    - os.h ----------→ 해드파일 연결
    - model.c
    - mylib.c ----------→
    - initialize.c ----------→ 객체 정적 초기화
    - kernel.c ----------→ API 함수 모델
    - messageQ.c
    - message.h
    - readyQ.h
    - readyQ.c
    - waitingQ.c
    - waitingQ.h
  - API list:
    - Task 부분:
      - int task_create(unsigned char tid);
      - int TerminateTask();
      - int task_sleep(unsigned int time);
      - int time_checker(unsigned char);-> sleeping 중인 task가 resume 하는지 check
      - int Round_Robin_Schedule();

- Mutex 부분:
  - int mutex_create(mutex_pt*)
  - int mutex_delete(mutex_pt*);
  - int mutex_lock(mutex_pt*);
  - int mutex_unlock(mutex_pt*)
  - int mutex_is_locked(mutex_pt*);
  - int mutex_lock_timed(mutex_pt*, unsigned int time);
  - int mutex_time_checker(mutex_pt *, unsigned char tid);
- semaphore 부분:
  - int sem_create(sem_pt *);
  - int sem_delete(sem_pt *);
  - int sem_take(sem_pt*);
  - int sem_give(sem_pt*);
  - int sem_take_timed(sem_pt*, unsigned int time);
  - int sem_time_checker(sem_pt*, unsigned char tid);
- message 부분
  - int msgq_create(msgq_pt* , unsigned int, unsigned int);
  - int msgq_receive(msgq_pt*, unsigned char *);
  - int msgq_send(msgq_pt*, unsigned char *);
- start up & shut down:
  - void ubik_comp_statr();
  - void ShutDownOS();

- model 사용법 (필요한면 수정):
  - config 파일 부분:
    - Task 몇 개 있는지
    - 최대한 Tid 및 최소한 Tid 얼마인지
    - 최대한 우선순위 정의
    - Tid 정의

```
#ifndef CONFIG_H_
#define CONFIG_H_
#define NUM_OF_TASKS 10
#define MIN_TASK_ID 1
#define MAX_TASK_ID 10
#define MAX_PRIORITY 5

#define Task1 1
#define Task2 2
#define Task3 3
#define Task4 4


#define testalarm 1
#endif
```

  - Initialize 파일 부분:
    - task의 priority 정의
    - 밑에 그대로 사용

```
    task_static_info[1].max_act_cnt = 1;
    task_static_info[1].prio = 3;
```

```
task_dyn_info[1].dyn_prio = task_static_info[1].prio;
task_dyn_info[1].act_cnt = 0;
```

  - Source code 부분:
    - 해더파일 include
      - #Include "os.h"
    - TASK macro 및 jump macro
      - #define TASK(t) TASK_##t()
      - 여기인 tid는 실제 사용할 때 대응한 task id 로 바꾸면 된다. (Task 1인 경우 jump_1{)

```
#define jump_tid(){\
        switch (current_pc[tid])\
        {\
                case 0:\
                        goto L_tid_0;break;\
                case 1:\
                        goto L_tid_1;break;\
                case 2:\
                        goto L_tid_2;break;\
                case 3:\
                        goto L_tid_3;break;\
        }\
}
```

    - Task create, Terminate 및 Round Robin

```
flag = task_create(1);
```

      - flag는 1인 경우 create successful, 0인 경우는 더 높은 priority인 task가 수행하고 있으면 create한 task가 readyQ으로 들어간다.
      -

```
flag = TerminateTask();
```

      - 수행중인 task가 terminate

```
extern int Round_Robin_Schedule();
```

```
current_pc[1]++;
flag = Round_Robin_Schedule();
if(flag)
        return;
```

      - Round robin 해야 하는 위치에서 Round_Robin_Schedule() 호출하면 된다.

(context switching point 미이 기록해야 한다)

- Scheduler(running 함수):

```
void running()
{
        while(current_tid >= 0)
        {
                if(current_tid == 1)
                        TASK(1);
                else if(current_tid == 2)
                        TASK(2);
        }
}
```

◆ Mutex 부분:

- memory allocation 및 mutex create

```
mutex_pt *mutex;
mutex = (mutex_pt*)malloc(sizeof(mutex_pt));
```

- muntx lock 및 unlock

```
flag = mutex_lock(mutex);
flag =  mutex_unlock(mutex);
```

- Mutex lock timed 및 time checker

```
int mutex_checker;
void running()
{
        while(current_tid >= 0)
                mutex_checker++;
        if(mutex_checker<10)
        {
                ...
        }
        else if(mutex_checker == 10)
        {
                mutex_time_checker(mutex,1);
                mutex_checker = 0;
        }
}
```

```
current_pc[1]++;
flag = mutex_lock_timed(mutex, 1000);
if (flag)
        return;
```

◆ Semaphore 부분:

- memory allocation 및 sem create

```
sem_pt* sem;
sem = (sem_pt*)malloc(sizeof(sem_pt));
sem_create(sem);
```

- Sem give 및 take

```
sem_flag[1] = sem_give(sem);
sem_flag[3] = sem_take(sem);
```

- Sem take timed는 mutex 랑 똑같은 식으로 구현 하면 된다.

◆ Message queue 부분:

- memory allocation 및 message 생성

```
msgq_pt* _g_msgq;
_g_msgq = (msgq_pt*)malloc(sizeof(msgq_pt));
msgq_create(_g_msgq, 25, 0);
```

- Meesage send 및 receive

```
mess_flag[1] = msgq_send(_g_msgq, buf);
mess_flag[2] = msgq_receive(_g_msgq, buf1);
```

- 참고 예제

■ RR_test:

◆ 같은 priority인 task가 Round robin schedule는 정상적으로 수행되는지 check하는 case.

- mutex_test1.c 및 mutex_test2.c
  - 간단한 mutex lock 및 unlock test 하는 case
  - Mutex priority 특성 test 하는 case
- sem_test.c
  - semaphore behavior check 하는 case
- messageQ.c
  - message queue 부분 test 하는 case