Introduction to Operating Systems
Project #3: Periodic Tasks and Priority Scheduling

05/12/2022

Prof. Seongsoo Hong
sshong@redwood.snu.ac.kr
SNU RTOSLab
Dept. of Electrical and Computer Engineering
Seoul National University

Agenda

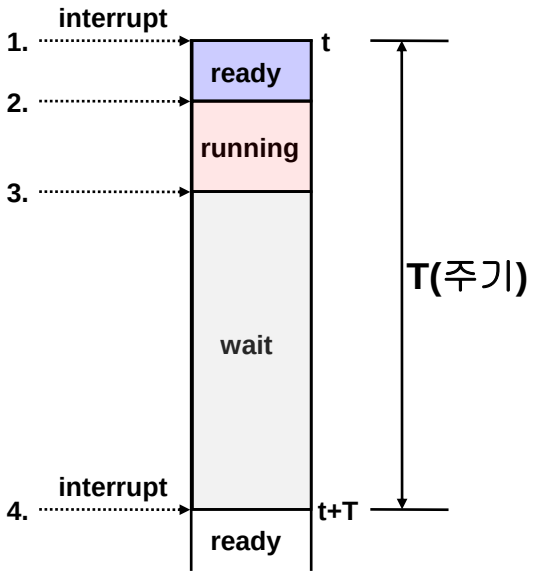
I. **Goal of Project #3**
II. Project Description
III. Project Submission

과제 목적 (1)

- 과제 목적
  - 우선순위 스케쥴러의 구현
  - 카운터와 알람을 이용하여
    주기적으로 깨어나는 주기 태스크의 구현

과제 목적 (2)

- 타겟 시나리오
  1. 주기가 돌아오면 타이머 인터럽트 핸들러가 태스크를 큐에 삽입(ready)
  2. 큐에서 우선순위가 가장 높아졌을 때, 스케쥴러에 의해 CPU 점유권을 얻어 작업 수행(running)
  3. 수행이 끝나면 다음 주기가 될 때까지 기다림(waiting)
  4. 다음 주기가 돌아오면 1~3의 과정을 반복



Agenda

I. Goal of Project #3

II. Project Description

III. Project Submission

과제 설명 (1)

- 수정해야 할 API 목록
  - `core/task.c`
    1. eos_tcb_t – 태스크 컨트롤 블록
    2. eos_schedule() – 태스크 스케줄링
    3. eos_create_task() – 태스크의 생성

- 새로 구현해야 할 API 목록
  - `core/task.c`
    4. eos_set_period() – 태스크 주기 설정
    5. eos_sleep() – 태스크를 재움
    6. _os_wakeup_sleeping_task() – 태스크를 깨움
  - `core/timer.c`
    7. eos_set_alarm() – 알람 설정 및 해제
    8. eos_trigger_counter() – 카운터 증가 및 콜백함수 호출

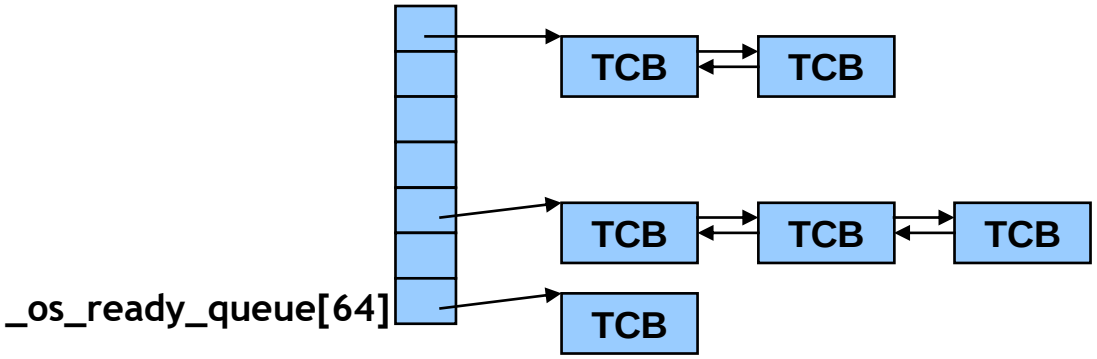
과제 설명 (2)

- eos_schedule() – 태스크 스케줄링
  - 과제 2에서 구현한 **eos_schedule()** 함수를 멀티 레벨 ready 큐를 이용하여 우선순위 스케줄링이 가능하도록 구현
  - _os_ready_queue[] 구조체 이용

가장 높은 우선순위를 찾거나 **ready** 큐에 태스크를 삽입/제거할 때 **O(1)** 스케줄러 모듈의 **API**를 이용한다.

과제 설명 (3)

- 멀티 레벨 ready 큐
  - O(1) 스케줄러에 의해 가장 높은 우선순위를 얻은 뒤, 멀티 레벨 ready 큐에서 TCB 포인터를 얻어 옴
  - 우선순위가 가장 높은 태스크가 여러 개일 경우, FIFO 방식에 의해 태스크 선택
  - 낮은 숫자가 높은 우선순위를 의미



과제 설명 (4)

- O(1) 스케줄러
  - Bitmap을 이용하여 O(1)의 시간에 우선순위가 가장 높은 태스크를 찾을 수 있음
  - O(1) 스케줄러는 `core/scheduler.c`에 이미 구현됨

- 제공되는 API

_os_get_highest_priority()	가장 높은 우선순위를 얻어옴 다음에 수행시킬 태스크를 찾을 때 사용
_os_set_ready()	비트맵의 해당 우선순위를 1로 세팅 태스크를 ready 큐에 삽입할 때 사용
_os_unset_ready()	비트맵의 해당 우선순위를 0으로 세팅 태스크를 ready 큐에서 제거할 때 사용

과제 설명 (5)

- eos_create_task() – 태스크의 생성
  - 과제 2에서 구현한 eos_create_task() 함수를 다음과 같이 변경
    1. 추가로 필요한 TCB의 필드를 초기화
    2. 생성된 태스크는 queue에 넣고 ready 상태로 설정

과제 설명 (6)

4. eos_set_period() – 태스크의 주기 설정

- 태스크를 주기 태스크로 만들기 위해 필요한 TCB 필드를 설정

형	void eos_set_period(eos_tcb_t *task, int32u_t period)
입력	*task: 대상 태스크의 TCB *period: 태스크의 주기
출력	없음

과제 설명 (7)

- eos_sleep() – 태스크의 수행을 멈추고 대기시킴
  - Argument인 tick은 0으로 들어온다고 상정할 것
  - eos_set_alarm() 함수를 호출하여
    다음 주기까지 남은 시간만큼 알람 설정
    - 콜백 함수: _os_wakeup_sleeping_task()

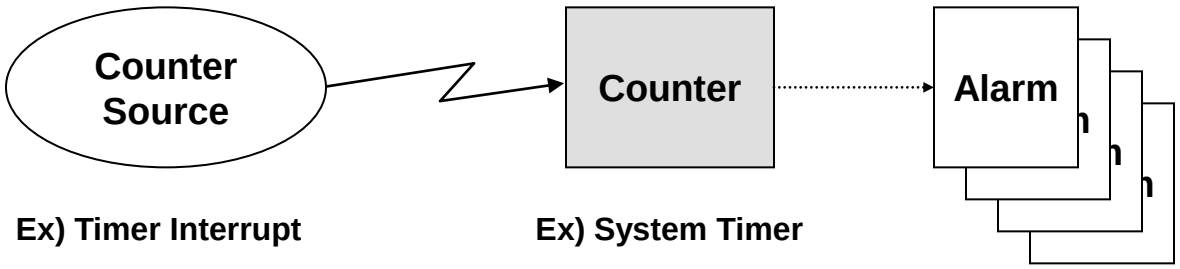
과제 설명 (8)

- _os_wakeup_sleeping_task() – 태스크를 깨움
  - 태스크를 ready 상태로 바꾸고 스케줄링

형	void _os_wakeup_sleeping_task (eos_tcb_t *task)
입력	*task: 깨울 태스크의 TCB
출력	없음

과제 설명 (9)

- 카운터와 알람
  - 카운터
    - 타이머 인터럽트 등 특정한 동작이 발생한 횟수를 센다
    - 기본적으로 한 개의 카운터(시스템 카운터)가 존재한다
  - 알람
    - 카운터가 정해진 값에 도달하면 등록된 콜백 함수를 호출한다
    - 카운터마다 알람 큐가 존재하며
      알람들은 남은 시간 순으로 정렬되어 있다



과제 설명 (10)

- eos_set_alarm() – 알람 설정 및 해제
  1. 카운터의 알람큐에서 알람을 제거
  2. Timeout이 0이거나 entry가 NULL일 경우 리턴 (알람 해제)
  3. 인자로 전달된 값들로 알람 구조체의 각 필드를 채움
  4. 카운터의 알람큐의 적절한 위치에 알람 추가

형	void eos_set_alarm(eos_counter_t* counter, eos_alarm_t* alarm, int32u_t timeout, void (*entry)(void *arg), void *arg)
입력	*counter: 알람을 설정/해제할 카운터 구조체 *alarm: 설정/해제할 알람 구조체 *timeout: 알람이 만료될 시간 *entry: 알람이 만료되었을 때 호출될 콜백 함수 *arg: 콜백 함수의 전달 인자
출력	없음

과제 설명 (11)

- eos_trigger_counter() – 카운터 증가 및 콜백함수 호출
  1. 카운터의 tick을 1 증가시킴
  2. 카운터의 알람 큐를 확인하여 만료된 알람의 콜백함수 호출

형	void eos_trigger_counter(eos_counter_t* counter)
입력	counter: 증가시킬 카운터 구조체
출력	없음

Agenda

I. Goal of Project #3
II. Project Description
III. **Project Submission**

제출물 (1)

- 제출물
  - 수정한 EOS 코드
  - 보고서
    - 정의한 구조체에 대한 설명
    - 구현한 함수들에 대한 설명
    - 테스트 프로그램 수행 결과

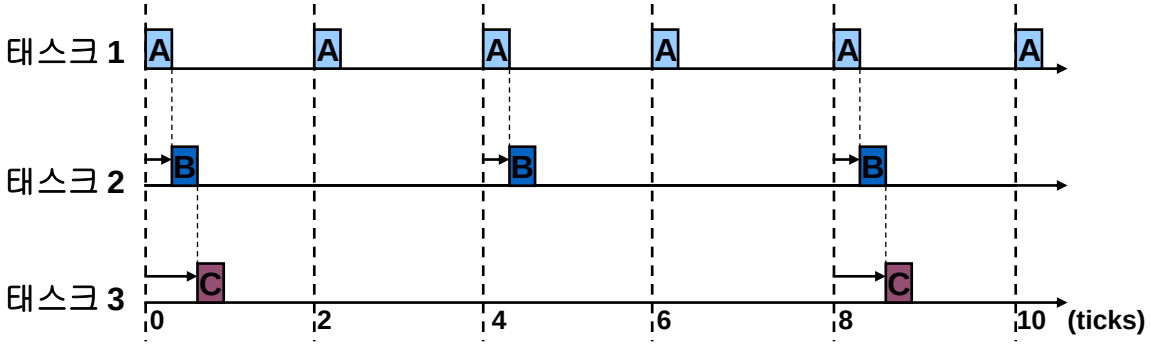
- 주의사항
  - 주어진 함수 prototype을 변경하지 말 것
  - 가능한 새로운 함수나 글로벌 변수 추가를 하지 말 것
  - 보고서 분량은 5페이지 이하

제출물 (2)

- 테스트 프로그램
  - 세 개의 태스크 생성

	우선순위	주기(tick)	역할
태스크 1	1	2	'A' 출력
태스크 2	10	4	'B' 출력
태스크 3	50	8	'C' 출력

  - 각 태스크의 동작
    - while() 문을 수행하면서
      각 loop마다 특정 문자열을 출력한 다음 eos_sleep(0) 호출



제출물 (3)

```
#include <core/eos.h>
#define STACK_SIZE 8096

int8u_t stack1[STACK_SIZE];    // stack for task1
int8u_t stack2[STACK_SIZE];    // stack for task2
int8u_t stack3[STACK_SIZE];    // stack for task3
eos_tcb_t tcb1;            // tcb for task1
eos_tcb_t tcb2;            // tcb for task2
eos_tcb_t tcb3;            // tcb for task3
```

제출물 (4)

```
void task1() {
	while(1) { PRINT("A\n"); eos_sleep(0); }// 'A' 출력 후 다음 주기까지 기다림
}

void task2() {
	while(1) { PRINT("B\n"); eos_sleep(0); }// 'B' 출력 후 다음 주기까지 기다림
}

void task3() {
	while(1) { PRINT("C\n"); eos_sleep(0); }// 'C' 출력 후 다음 주기까지 기다림
}

void eos_user_main() {
	eos_create_task(&tcb1, stack1, STACK_SIZE, task1, NULL, 1);		// 태스크 1 생성
	eos_set_period(&tcb1, 2);

	eos_create_task(&tcb2, stack2, STACK_SIZE, task2, NULL, 10);		// 태스크 2 생성
	eos_set_period(&tcb2, 4);

	eos_create_task(&tcb3, stack3, STACK_SIZE, task3, NULL, 50);		// 태스크 3 생성
	eos_set_period(&tcb3, 8);
}
```

제출물 (5)

- 제출 기한
  - 5/26(목) 11:59 PM 전까지
- 제출 방법
  - ETL에 업로드
    - 보고서는 워드 파일 또는 PDF
    - 소스 코드는 make clean을 하여 정리 후 압축
      - hal/current를 삭제 후 압축하는 것을 권장
      - 압축 에러의 원인이 되는 경우가 있기 때문
      - symbolic link일 뿐이기에 삭제해도 make all을 할 때 다시 생성

Question or Comment?