



Embedded Software

(MicroC/OS-II: 4)

Event flag and semaphore

Fall, 2017



Outline

- Overview
- What to do?
 - 숫자 게임



Overview

- 목적
 - Semaphore 활용하기
 - Event flag 활용하기



Semaphore 사용법

- 변수 선언
 - `OS_EVENT *Sem;`
- 초기화
 - `Sem = OSSemCreate(1);` => 초기값 1
 - parameter : 해당 세마포어 초기값
- **Wait** (세마포어 값 1이상이면 1 감소하고 진행 0이면 대기)
 - `OSSemPend(Sem, 0, &err);`
 - parameter : 세마포어 주소, 기다릴 시간, err
- **Signal** (세마포어 값 1증가)
 - `OSSemPost(Sem);`
 - parameter : 해당 세마포어 주소



Event flag 사용법 (1)

- 변수 선언
 - OS_FLAG_GRP *flag_grp;
- 초기화
 - flag_grp = OSFlagCreate(0x00, &err);
 - parameter : 해당 플래그 초기값, err



Event flag 사용법 (2)

■ Wait

- OSFlagPend(flag_grp, 조건 값, 옵션, 0, &err);
- Parameter : flag의 주소, 조건 값, 옵션, timeout, err
- OPTION 상수
 - OS_FLAG_WAIT_SET_ANY: flag_grp의 값과 조건 값 & 연산 시 하나라도 1비트가 존재하면 진행
 - OS_FLAG_WAIT_SET_ALL : flag_grp의 값과 조건 값이 같을 때 진행
 - +OS_FLAG_CONSUME : 진행 될 경우 flag값 조건 값에서 1인 비트 위치들만 0으로 수정
 - Example : [00001111, 00001000 => 00000111]
OSFlagPend(0x0F,0x08,OS_FLAG_WAIT_SET_ALL+OS_FLAG_CONSUME,0,&err) => flag = 0x07



Event flag 사용법 (3)

- Signal

- OSFlagPost(flag_grp, 조건 값, 옵션, &err);
- Parameter : flag의 주소, 조건 값, 옵션, err
- OPTION 상수
 - OS_FLAG_SET:
flag_grp의 값과 조건 값 OR 연산 수행



Assignments

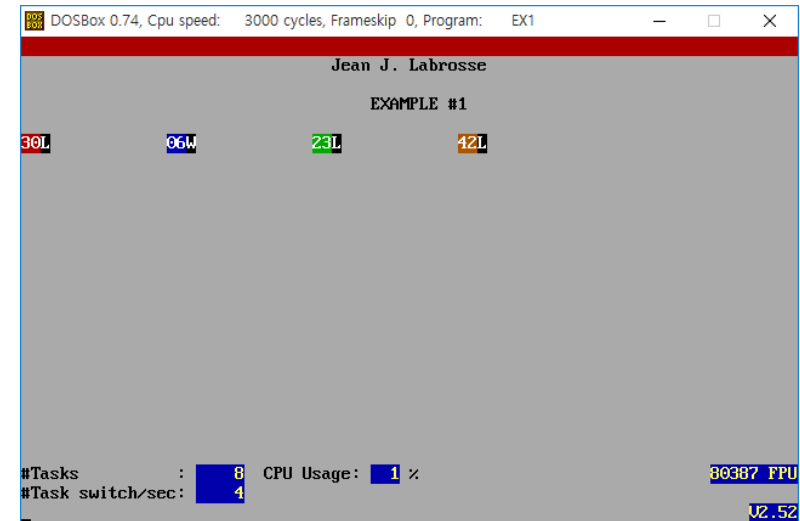
- I-Class에 다음 슬라이드의 과제 제출
- 제출 파일
 - 소스 코드(EX1.C)
 - 보고서(.pdf)
 - 실행 결과 화면 캡처
 - You have to capture and submit the screen on your report
- 파일명
 - 7주차_학번_이름(압축하여 하나의 파일로 제출)
- 제출 기한
 - 다음주 화요일 자정까지



What to do ?

- 1. 5개의 태스크 생성
 - 1개의 Decision task
 - 4개의 Random task
- 2. 4개의 Random task 에서 임의의 숫자를 0부터 63까지 1개 생성할 것
- 3. 4개의 Random task는 해당 숫자를 배열 변수에 저장함
 - 배열 변수
 - OSSemPend, OSSemPost 함수를 활용함
 - OSFlagPost, OSFlagPend 함수를 활용함
- 4. Decision task 는 4개의 태스크 중에서 가장 작은 숫자를 보내온 태스크를 선택하여 해당 배열 변수에 'W' 문자를 저장하고, 나머지 태스크들에게 'L' 이란 문자를 저장한다.
 - 배열 변수를 활용함
 - OSSemPend, OSSemPost 함수를 활용함
 - OSFlagPost, OSFlagPend 함수를 활용함

- 6. 'W' 를 받은 태스크는 색깔을 칠한다.
 - random task 우선순위에 따라서 다음의 색을 칠한다.
 - 가장 높은 태스크
 - 빨강
 - 2번째 우선순위 태스크
 - 파랑
 - 3번째 우선순위 태스크
 - 녹색
 - 4번째 우선순위 태스크
 - 갈색
- 7. 모든 태스크는 잠시 쉬고, 위의 과정을 다시 반복한다.

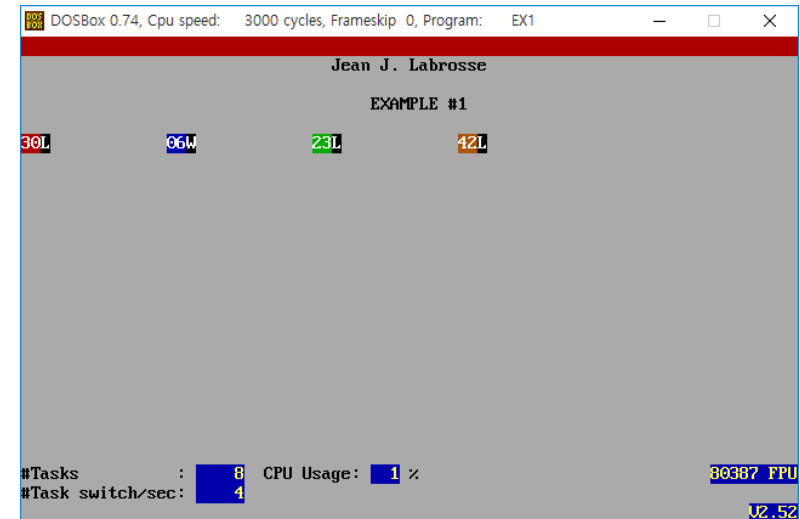


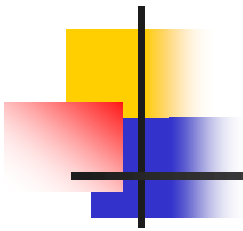


What to do ?

- 1. Create 5 tasks
 - 1 x Decision task
 - 4 x Random task
- 2. Generate a Random number(0-63) on each Random task
- 3. Each Random task put the numbers into the variable array
 - Implement using functions OSSemPend, OSSemPost, OSFlagPost, OSFlagPend
- 4. Decision task picks the minimum number from them and send back 'W' to the task sent the minimum and 'L' to the other 3 tasks.
 - Implement using functions OSSemPend, OSSemPost, OSFlagPost, OSFlagPend

- 6. Task got 'W' prints color
 - According to its own priority, each task prints color below
 - The highest priority task
 - Red
 - The second priority task
 - Blue
 - The third priority task
 - Green
 - The fourth priority task
 - Brown
- 7. Every tasks wait for seconds, iterate every step above.



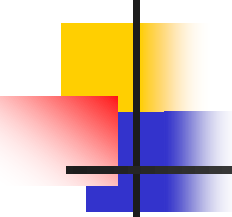


변수

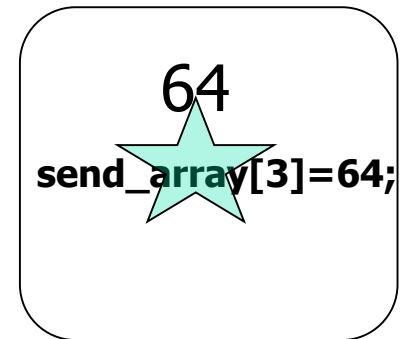
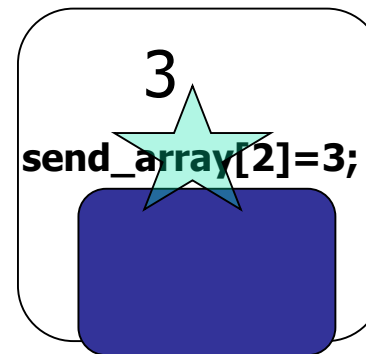
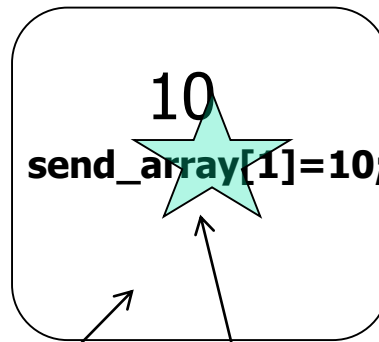
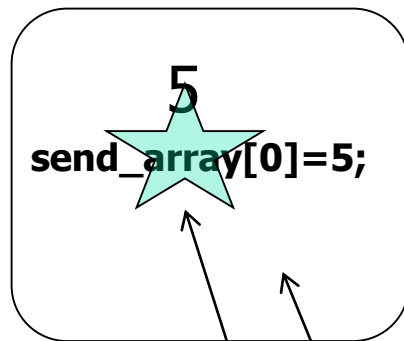
```
int send_array[4];  
char receive_array[4];
```

Semaphore로 보호

```
OS_FLAG_GRP *r_grp;  
OS_FLAG_GRP *s_grp;  
  
s_grp = OSFlagCreate(0x00, &err);  
r_grp = OSFlagCreate(0x00, &err);
```

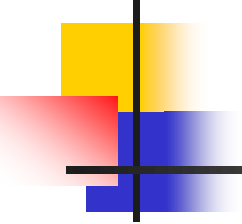


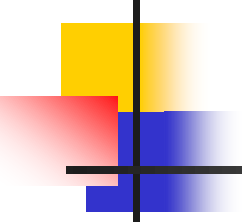
```
OSFlagPend(s_grp, 0x0F, OS_FLAG_WAIT_SET_ALL + OS_FLAG_CONSUME, 0, &err);  
    receive_array[0]='L';  
    receive_array[1]='L';  
    receive_array[2]='W';  
    receive_array[3]='L';  
OSFlagPost(r_grp, 0x0F, OS_FLAG_SET, &err);
```



```
OSFlagPost(s_grp, 0x01, OS_FLAG_SET, &err);  
OSFlagPost(s_grp, 0x02, OS_FLAG_SET, &err);  
OSFlagPend(r_grp, 0x01, OS_FLAG_WAIT_SET_ANY + OS_FLAG_CONSUME, 0, &err);  
OSFlagPend(r_grp, 0x02, OS_FLAG_WAIT_SET_ANY + OS_FLAG_CONSUME, 0, &err);
```

Receive_array 체크

- 
-
- 본 설명은 하나의 예일 뿐이며, **event flag**, **semaphore**를 활용하여 구현할 것

- 
-
- 실습 후 C:\SOFTWARE 폴더 삭제해주세요.
 - Before you go out,
please remove C:\SOFTWARE folder.