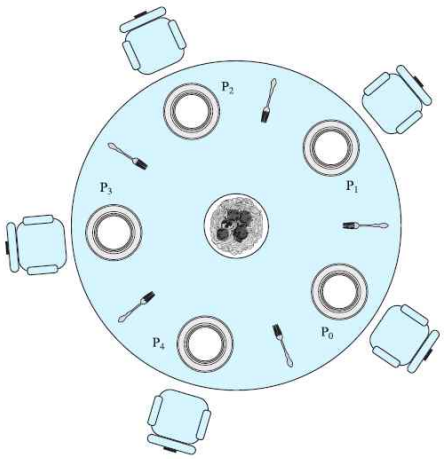
[고급프로그래밍: Pthread Synchronization 디버깅 문제: 생각하는 철학자]

- 제출마감: 2019년 10월 27일(일) 오후 11시 59분
- e-class의 과제물로 제시된 생각하는 철학자 프로그램을 상호배제가 완벽하게 수행되면서도 교착상태(deadlock)가 발생하지 않는 프로그램으로 재 작성하시오.
- 과제 제출: e-class에 dining-학번.c 파일로 업로드 하시오.
- dining.c 원본 파일: computer.kpu.ac.kr:/tmp/advancedOS 디렉토리에 있으므로 복사해서 사용
- **컴파일 옵션: gcc dining.c -lpthread**

[문제 1-1] 첨부 파일로 주어진 dining.c는 생각하는 철학자(dining philosopher) 문제를 POSIX Thread를 이용하여 구현한 프로그램이다. 원래 작성된 프로그램은 상호배제 기법(pthread_mutex_lock()과 pthread_mutex_unlock())을 이용하여 작성되어 실행을 시키면 교착상태에 빠져야 하는데, 교착상태에 빠지지 않고 모든 철학자가 식사를 모두 마치게 되어, 특정 시기에 다른 철학자와 포크를 동시에 공유할 수 있다는 모순에 빠진다. 잘못된 부분을 찾아 주어진 프로그램을 변경하여 특정 시기에 하나의 포크는 한 철학자에게만 배정될 수 있는 프로그램으로 변경하시오. (10점)

[문제 1-2] 특정 시기에 하나의 포크가 한 철학자에게만 배정될 수 있는 프로그램으로 변경하여 수행시키면, 변경된 프로그램은 교착상태(deadlock)에 빠지게 된다. dining.c에 정의된 카운팅 세마포어(counting semaphore) 변수 room을 이용한 세마포어 함수 sem_wait()와 sem_post()를 사용하여 교착상태가 발생하지 않는 프로그램으로 재작성하시오. (10점)

[생각하는 철학자(dining philosopher) 문제란]



생각하는 철학자 문제는 5명의 철학자가 원탁 테이블에 앉아 식사하는 것으로 Dijkstra가 소개하였다. 같은 집에서 살고 있는 철학자의 삶은 사색과 식사로 구성된 단순한 삶이다. 또한 철학자들은 오랜 사색을 통해 스파게티가 자신들의 사색에 도움이 되는 유일한 음식이라는 결론에 도달하였다. 철학자들은 스파게티를 먹기 위해 2개의 포크를 사용한다. 식탁의 배치는 위 그림과 같다. 원탁 테이블의 가운데에는 스파게티가 담긴 큰 그릇이 하나 있다. 그리고 각 철학자가 개인적으로 사용하는 접시가 5개 있으며, 접시 양쪽에는 포크가 있다(결국, 테이블 위에는 5개의 포크가 있다). 철학자는 배가 고프면 자신의 정해진 위치로 가서 큰 그릇에 담긴 스파게티를 자신의 접시에 담아 먹는다. 이때 문제에서 철학자는 식사를 위해 두 개의 포크를 사용해야 한다는 것이며, 포크가 인접한 철학자가 사용하지 않을 때만 사용가능하다는 것이다. 철학자는 자신의 접시 양 옆에 놓인 포크를 사용하려 한다. **이때 다음 두 가지가 고려되어야 한다. 첫째, 임계영역이 지켜져야 한다. 즉 두 명의 철학자가 동시에 같은 포크를 사용할 수는 없다. 둘째, 교착상태나 기아에 빠져서는 안 된다.**

[dining.c]

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>

void *func(int n);
pthread_t philosopher[5];
pthread_mutex_t chopstick[5];

int main()
{
        int i;
        void *msg;
        for(i=1;i<=5;i++)
        {
                pthread_mutex_init(&chopstick[i],NULL);
        }
        for(i=1;i<=5;i++)
        {
                pthread_create(&philosopher[i],NULL,(void *)func,(int *)i);
        }
        for(i=1;i<=5;i++)
        {
                pthread_join(philosopher[i],&msg);
        }
        printf("\n");
         for(i=1;i<=5;i++)
         {
                pthread_mutex_destroy(&chopstick[i]);
         }
        return 0;
}

void *func(int n)
{
        printf ("\nPhilosopher %d is thinking ",n);
        printf("\nPhilosopher %d is in the dining room ", n);
        pthread_mutex_lock(&chopstick[n]);
            sleep(3);
        pthread_mutex_lock(&chopstick[(n+1)%5]);
            printf ("\nPhilosopher %d is eating ",n);
            sleep(2);
        pthread_mutex_unlock(&chopstick[n]);
        pthread_mutex_unlock(&chopstick[(n+1)%5]);
```

```
        printf ("\nPhilosopher %d finished eating ",n);
}
```

[입력] 없음

[문제 1-1: 출력 예제]

Philosopher 1 is thinking
Philosopher 1 is in the dining room
Philosopher 2 is thinking
Philosopher 2 is in the dining room
Philosopher 3 is thinking
Philosopher 3 is in the dining room
Philosopher 4 is thinking
Philosopher 5 is thinking
Philosopher 5 is in the dining room
Philosopher 3 is eating
Philosopher 3 finished eating
Philosopher 2 is eating
Philosopher 4 is in the dining room
Philosopher 2 finished eating
Philosopher 1 is eating
Philosopher 1 finished eating
Philosopher 5 is eating
Philosopher 5 finished eating
Philosopher 4 is eating
Philosopher 4 finished eating

[문제 1-2: 출력 예제]

Philosopher 0 is thinking
Philosopher 0 is in the dining room
Philosopher 1 is thinking
Philosopher 1 is in the dining room
Philosopher 2 is thinking
Philosopher 2 is in the dining room
Philosopher 3 is thinking
Philosopher 3 is in the dining room
Philosopher 4 is thinking
Philosopher 4 is in the dining room
<이 시점에서 중단됨: 교착상태에 빠짐>