42: Philosophers

Concepts

- The basics of threading a process (프로세스 스레딩에 대한 기본 개념)
- How to work on the same memory space (공유 메모리 공간을 어떻게 작업하는지)
- How to make threads (스레드 만드는 방법)
- Mutex / semaphore / shared memory (뮤텍스 / 세마포어 / 공유 메모리)
- 기본 개념들을 숙지한 채로, 동일한 기본 규칙을 가진 세 개의 개별 프로그램 작성하기

Basic Rules

- Norm (42 의 자체 코딩 컨벤션)을 따라, C 로 작성할 것.
- 어떤 누수, 충돌, 정의되지 않은 행동 혹은 코딩 컨벤션 에러 시 프로젝트는 0 점.
- 원형 테이블에 둘러 앉은 몇 명의 철학자들은 식사를 하거나, 생각하거나, 자고 있다.
- 철학자들은 동시에 하나의 행동만 한다. 이를 테면 식사를 하고 있을 때는 생각하거나, 자지 않는다.
- 원형 테이블 중앙에는 스파게티가 든 큰 접시가 있다.
- 스파게티가 포크 하나로 먹기 어려워서, 철학자들은 양 손에 포크를 들고 스파게티를 먹는다.
- 철학자들은 굶어 죽으면 안 된다.
- 모든 철학자들은 스파게티를 먹고 싶다.
- 철학자들은 서로 말하지 않는다.
- 철학자들은 다른 철학자가 언제 (굶어) 죽을지 모른다.
- 한 명의 철학자가 식사를 끝내면, 그는 포크를 내려놓고 잠자기 시작한다.
- 철학자가 잠자기를 끝내면, 생각하기를 시작한다.
- 한 철학자가 죽으면 시뮬레이션이 끝난다.
- 개별 프로그램은 다음과 같은 동일한 옵션들을 가져야 한다.
    - number_of_philosophers : 철학자의 수, 즉 포크의 갯수
    - time_to_die : 밀리세컨 단위. 한 철학자가 식사를 시작하지 않으면, 그가 마지막 식사를 시작하거나 시뮬레이션이 시작된지 'time_to_die' 초 내에 그는 죽는다.
    - time_to_eat : 밀리세컨 단위. 철학자가 식사를 하는 시간. 이 시간동안 철학자는 포크 두 개를 들고 있다.
    - time_to_sleep: 밀리세컨 단위. 철학자가 자는 시간.
    - number_of_times_each_philosopher_must_eat: (optional argument). 만약 모든 철학자들이 최소한 이 횟수만큼 식사를 했다면, 시뮬레이션은 끝난다. 만약 이 인자가 설정되지 않았다면, 철학자가 죽었을 때만 시뮬레이션이 끝난다.

- 각각의 철학자는 1 부터 ‘number_of_philosophers’ 사이의 숫자들 중 각각의 숫자를 부여받게 된다.
    - 1 부터 ‘number_of_philosophers’ 까지의 순서대로 철학자들은 둘러 앉아 있다. (이를테면 n 번 철학자 옆에는 n+1 번 철학자가 앉아 있다.)
- 철학자의 상태가 변경될 때마다 다음과 같이 출력한다.
    - timestamp_in_ms X has taken a fork
    - timestamp_in_mx X is eating
    - timestamp_in_ms X is sleeping
    - timestamp_in_ms X is thinking
    - timestamp_in_ms X died
    - X 는 철학자가 부여받은 숫자로, timestamp_in_ms 는 밀리세컨 단위의 소요된 시간을 기록
- 출력될 상태들은 다른 철학자의 상태와 섞이거나 혼합되면 안된다
- 한 철학자의 죽음과 그 죽음에 대한 출력 사이에 10 밀리세컨 이상의 시간차가 있으면 안된다
- 강조, 철학자가 죽는 상황을 최대한 피하자.

Expected Outputs

프로그램 1

- Threads 와 mutex 를 사용하는 철학자
- 각각의 철학자 사이에 하나의 포크가 있어서, 결국 모든 철학자들의 양쪽에 포크가 놓여 있게 된다.
- 철학자들이 포크를 중복 사용하는 것을 막기 위해, 각각의 포크의 상태를 mutex 로 보호해야 한다.
- 각각의 철학자는 thread 여야 한다.

프로그램 2

- Threads 와 semaphore 을 사용하는 철학자
- 모든 포크들이 테이블 한가운데에 있다.
- 메모리 안에 상태는 없지만, 사용 가능한 포크의 개수가 semaphore 을 통해 표시된다.
- 각각의 철학자는 thread 여야 한다.

프로그램 3

- processes 와 semaphore 을 사용하는 철학자
- 모든 포크들이 테이블 한가운데에 있다.
- 메모리 안에 상태는 없지만, 사용 가능한 포크의 개수가 semaphore 을 통해 표시된다.
- 각각의 철학자들은 process 여야 하고, 메인 프로세스는 철학자가 아니여야 한다.

References

- 식사하는 철학자 문제

식사하는 철학자들 문제는 전산학에서 동시성과 교착 상태를 설명하는 예시로, 여러 프로세스가 동시에 돌아갈 때 교착 상태가 나타나는 원인을 직관적으로 알 수 있다.

다섯 명의 철학자가 원탁에 앉아 있고, 각자의 앞에는 스파게티가 있고 양옆에 포크가 하나씩 있다. 그리고 각각의 철학자는 다른 철학자에게 말을 할 수 없다. 이때 철학자가 스파게티를 먹기 위해서는 양 옆의 포크를 동시에 들어야 한다. 이때 각각의 철학자가 왼쪽의 포크를 들고 그 다음 오른쪽의 포크를 들어서 스파게티를 먹는 알고리즘을 가지고 있으면, 다섯 철학자는 동시에 왼쪽의 포크를 들 수 있으나 오른쪽의 포크는 이미 가져가진 상태이기 때문에 다섯 명 모두가 무한정 서로를 기다리는 교착 상태에 빠지게 될 수 있다.

또한 어떤 경우에는 동시에 양쪽 포크를 집을 수 없어 식사를 하지 못하는 기아 상태가 발생할 수도 있고, 몇몇 철학자가 다른 철학자보다 식사를 적게 하는 경우가 발생하기도 한다.

에츠허르 데이크스트라의 해결책은 다음과 같다. 각각의 철학자를 P1, P2, P3, P4, P5 라고 하고, 각 철학자의 왼쪽 포크를 F1, F2, F3, F4, F5 라고 하자. P5 를 제외한 네 명은 먼저 Fn 를 집은 후 F(n+1)를 집는 방식을 취한다. 그리고 P5 는 이와 반대로, F1 를 먼저 집은 후 F5 를 집는다. 이것은 원래 방식의 대칭성을 제거하고, 따라서 교착 상태를 막을 수 있다.

- 식사하는 철학자들 문제 해석

중간 중간 엄청 웃기고 꽤 유익하다.... subject 에서 허용된 함수보다 많은 내장 함수들과 데이터 타입들을 사용하고 있다. 상호배제, 교착상태, 기아상태에 관한 자세한 원리들을 알 수 있으니 섭젝 시작 전에 보면 도움이 많이 될 듯.