PS9

The Dining Philosophers

นายฐากูร ทองเป๋

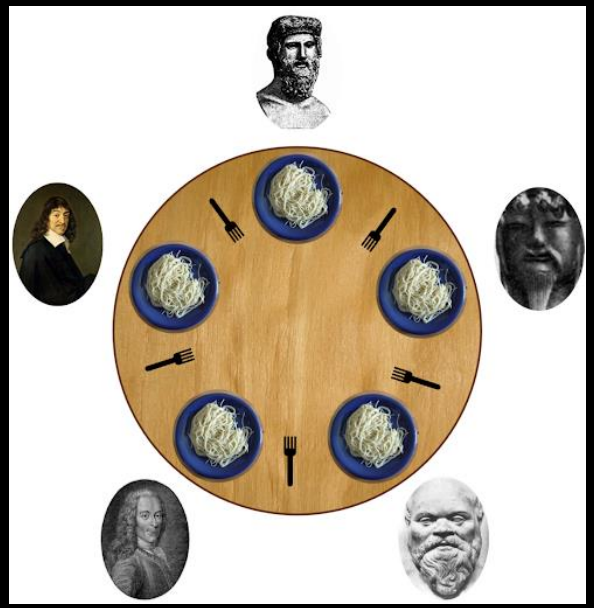
61070501020

นายฐาปนนท์ สดงาม

61070501021

นายลาภิศ ถิระเลิศพานิชย์

61070501044



อธิศพัฒน์ พงษ์อำไพ

61070501055

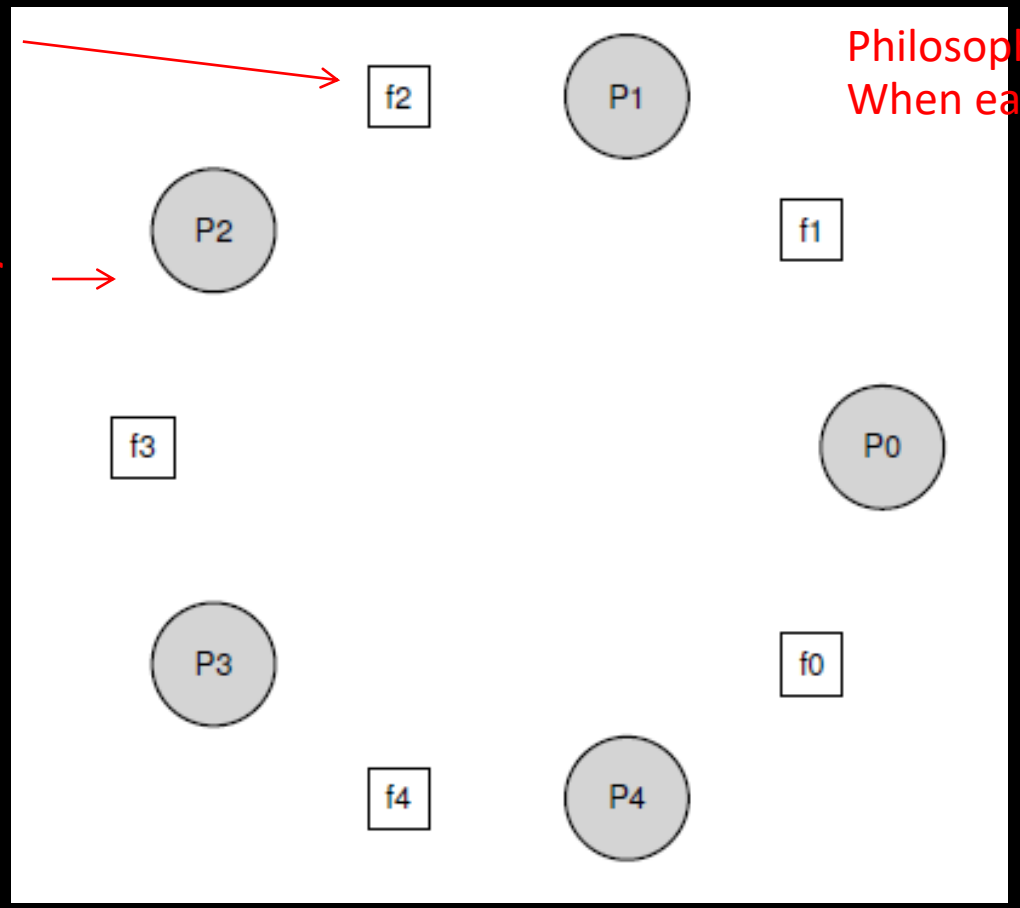
นายกฤตนัน ฉลอง

61070501002

The Dining Philosophers

fork

Philosopher



Philosophers either think or eat
When eat, they need two forks

```
while (1)
{
    think();
    getforks(p);
    eat();
    putforks(p);
}
```

Write the routines getforks(p) and putforks(p) such that there is no deadlock, no philosopher starves, and as many philosophers can eat at the same time as possible

Some Hints

- When philosopher *p* wishes to refer to the fork on their left, they simply call left(p)
- Similarly, the fork on the right of a philosopher is referred to by calling right(p)
  - int left(int p) { return p; }
  - int right(int p) { return (p + 1) % 5; }
- Declare the semaphores as follows:
  - sem_t forks[5];

Now do it

Variable

```c
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

#define Hungry 1
#define Thinking 0
#define Eating 2
int philosopher_states[5];
int phil[5] = {0, 1, 2, 3, 4};
sem_t mutex;
sem_t forks[5];
int left(int p) { return p; }

int right(int p) { return (p + 1) % 5; }
```

Dining philosopher เป็นการแก้ปัญหาการที่หลาย Thread มีการเข้าถึงตัวแปร ร่วมกัน ทำให้เกิด Race condition

สถานะของ Philosopher มี 3 สถานะ
เมื่อ Philosopher ต้องการใช้ Fork สถานะจะเปลี่ยนเป็น Hungry
เมื่อ Philosopher ไม่ต้องการใช้ Fork หรือ เป็นสถานะเริ่มต้น สถานะจะเป็น Thinking
เมื่อ Philosopher กำลังใช้งาน Fork สถานะจะเปลี่ยนเป็น Eating
sem_t mutex เป็นตัวแปรที่มีไว้สำหรับป้องกันไม่ให้ Philosopher เข้ามาหยิบส้อมพร้อมกัน
sem_t forks เป็นตัวแปรไว้สำหรับตรวจสอบสถานะของ Philosopher ตัวนั้นว่าได้กินหรือยัง

ฟังก์ชัน left มีไว้สำหรับ รีเทิร์นส้อมที่อยู่ด้านซ้าย
ฟังก์ชัน Right มีไว้สำหรับ รีเทิร์นส้อมที่อยู่ด้านขวา

Philosopher	P0	P1	P2	P3	P4
Philosopher_states	Thinking	Thinking	Thinking	Thinking	Thinking
Sem_t fork	0	0	0	0	0

Philosopher	P0	P1	P2	P3	P4
Philosopher_states	Thinking	Thinking	Eating	Thinking	Thinking
Sem_t fork	0	0	1	0	0

Philosopher

```
void *philosopher(void *num)
{
        while (1)
        {
                int *i = num;
                sleep(1);
                getforks(*i);
                sleep(1);
                putforks(*i);
        }
}
```

เป็น function thread ของ philosopher แต่ละคนที่จะทำการหยิบส้อม กิน วางส้อม ไปเรื่อย ๆ

พยายามหยิบส้อมและกิน

วางซ่อมแล้วส่งให้คนถัดไปทำงาน

Test And Eat

เป็น function ที่ใช้ทดสอบว่า ด้านซ้ายและขวา มีคนกำลังกินอาหารอยู่หรือไม่ หากไม่มีคนกินอยู่ตัว philosopher ก็จะทำการหยิบส้อมทั้งสองข้างและกินอาหาร

ทดสอบว่า ด้านซ้ายและขวา มีคนกำลังกินอาหารอยู่หรือไม่ และตัวเองกำลังหิวอยู่หรือไม่

```
void test_and_eat(int p)
{
        if (philosopher_states[p] == Hungry && philosopher_states[left((p + 4) % 5)] != Eating && philosopher_states[right(p)] != Eating)
        {
                philosopher_states[p] = Eating;
                sleep(1);
                printf("Philosopher %d takes fork %d and %d\n", p, left(p), right(p));
                printf("Philosopher %d is Eating\n", p);
                sem_post(&forks[p]);
        }
}
```

เปลี่ยน state จาก hungry เป็น eating

Sleep 1 วินาทีเพื่อจะสามารถ monitor การทำงานได้

ปลด philosopher p ออกจาก queue เนื่องจากทำการกินแล้ว

Getforks & Putforks

```c
void getforks(int p)
{
        sem_wait(&mutex);
        philosopher_states[p] = Hungry;
        printf("Philosopher %d is Hungry\n", p);
        test_and_eat(p);
        sem_post(&mutex);
        sem_wait(&forks[p]);
}
```

getforks() เป็นเมดธอดที่ใช้สำหรับหยิบส้อมเพื่อใช้ทำงาน

เปลี่ยนสถานะ mutex ให้ lock เพื่อป้องกันไม่ให้ process อื่นเรียกใช้
เปลี่ยนสถานะ philosopher เป็น Hungry
สั่งปริ้นต์แสดงสถานะของ philosopher -> hungry
ตรวจสอบสถานะให้กินได้หรือกินไม่ได้ด้วยเมธอด test_and_eat()
เปลี่ยนสถานะ mutex เป็น unlock
เปลี่ยนสถานะ fork เป็น lock

```c
void putforks(int p)
{
        sem_wait(&mutex);
        philosopher_states[p] = Thinking;
        printf("Philosopher %d putting fork %d and %d down\n", p, left(p), right(p));
        printf("Philosopher %d is thinking\n", p);
        test_and_eat(left(p));
        test_and_eat(right(p));
        sem_post(&mutex);
}
```

putforks() เป็นเมดธอดที่ใช้สำหรับวางส้อมเพื่อให้ process ข้าง ๆ ที่ต้องการใช้ต่อ

เปลี่ยนสถานะ mutex ให้ lock เพื่อป้องกันไม่ให้ process อื่นเรียกใช้
เปลี่ยนสถานะ philosopher เป็น Thinking
สั่งปริ้นต์แสดงสถานการณ์วางส้อม
สั่งปริ้นต์แสดงสถานะของ philosopher -> thinking
ตรวจสอบสถานการณ์กินของ process ข้าง ๆ แล้วให้ทำงานต่อ (ถ้าต้องการกิน)
เปลี่ยนสถานะ mutex เป็น unlock

```c
int main()
{
        pthread_t thread_id[5];
        sem_init(&mutex, 0, 1);
        for (int i = 0; i < 5; i++)
        {
                sem_init(&forks[i], 0, 0);
        }
        for (int i = 0; i < 5; i++)
        {

                // create philosopher processes
                pthread_create(&thread_id[i], NULL,
                                        philosopher, &phil[i]);

                printf("Philosopher %d is thinking\n", i);
        }
        for (int i = 0; i < 5; i++)
        {
                pthread_join(thread_id[i], NULL);
        }
}
```

Result

```
lapis@DESKTOP-L7AE34B:~/os$ ./dinning_philosophers
Philosopher 0 is thinking
Philosopher 1 is thinking
Philosopher 2 is thinking
Philosopher 3 is thinking
Philosopher 4 is thinking
Philosopher 4 is Hungry
Philosopher 4 takes fork 4 and 0
Philosopher 4 is Eating
Philosopher 0 is Hungry
Philosopher 3 is Hungry
Philosopher 2 is Hungry
Philosopher 2 takes fork 2 and 3
Philosopher 2 is Eating
Philosopher 1 is Hungry
Philosopher 4 putting fork 4 and 0 down
Philosopher 4 is thinking
Philosopher 0 takes fork 0 and 1
Philosopher 0 is Eating
Philosopher 2 putting fork 2 and 3 down
Philosopher 2 is thinking
Philosopher 3 takes fork 3 and 4
Philosopher 3 is Eating
Philosopher 0 putting fork 0 and 1 down
Philosopher 0 is thinking
Philosopher 1 takes fork 1 and 2
Philosopher 1 is Eating
Philosopher 4 is Hungry
Philosopher 3 putting fork 3 and 4 down
Philosopher 3 is thinking
Philosopher 4 takes fork 4 and 0
Philosopher 4 is Eating
Philosopher 2 is Hungry
Philosopher 0 is Hungry
Philosopher 1 putting fork 1 and 2 down
Philosopher 1 is thinking
```