

The Dining Philosophers Problem

The Dining Philosophers Problem

- In this project, you need to write a program to simulate the famous dining philosophers problem.
- This problem will require implementing a solution using Pthreads mutex locks and condition variables.

The Dining Philosophers Problem

- Begin by creating five philosophers, each identified by a number 0, 1, 2, 3 and 4. Each philosopher will run as a separate thread.
- Philosophers alternate between **thinking** and **eating**. To simulate both activities, have the thread **sleep for a random period** from one to three seconds.
每個哲學家都要 : sleep(think) -> try to eat -> eat
 - Each philosopher should think for a while and then become hungry.
 - If the philosopher is able to eat, the job she should do is go sleeping.

The Dining Philosophers Problem

- When a philosopher **wishes to eat**, she invokes the function

```
pickup_forks(int philosopher_number)
```

- Philosopher number identifies the ID of the philosopher wishing to eat.

- When a philosopher **finishes eating**, she invokes

```
return_forks(int philosopher_number)
```

- When we want to make a philosopher try to eat, she invokes the function **test(int philosopher_number)**

放下以後(return_forks)，就要去測試他兩旁的哲學家要不要吃飯

The Dining Philosophers Problem

- Since Pthreads is typically used in C programs—and since **C does not have a monitor**— we accomplish locking by **associating a condition variable with a mutex lock**.
- Condition variables in Pthreads use the pthread_cond_t data type and are initialized using the pthread_cond_init() function.

```
pthread_mutex_t mutex;  
pthread_cond_t cond_var;
```

```
pthread_mutex_init(&mutex, NULL);
```

- E.g.

```
pthread_cond_init(&cond_var, NULL);
```

The Dining Philosophers Problem

- The thread of each philosopher will be created and joined in order.
 - 0, 1, 2, ..., 4
 - Use a philosophers function as the input of pthread_create() to control the philosophers' actions.

```
45 void philosophers(int n)
46 {
47     //thinking
48
49     // become hungry
50
51
52     //start eating
53
54     //end eating
55     return ;
56 }
```

The Dining Philosophers Problem

- The thread of each philosopher will be created and joined in order.
 - 0, 1, 2, ..., 4
 - Use a function as the input of `pthread_create()` to control the philosophers' actions.
- You should print these lines out in the correct situations:
 - Philosopher %d is now THINKING for %d seconds.
 - Philosopher %d is now HUNGRY and trying to pick up forks.
 - Philosopher %d can't pick up forks and start waiting.
 - Philosopher %d returns forks and then starts TESTING %d and %d.
 - Philosopher %d is now EATING.

The Dining Philosophers Problem

```
rohan@rohan-VirtualBox:~/Desktop$ ./hw3.out
Philosopher 0 is now THINKING for 2 seconds
Philosopher 1 is now THINKING for 2 seconds
Philosopher 2 is now THINKING for 1 seconds
Philosopher 3 is now THINKING for 2 seconds
Philosopher 4 is now THINKING for 3 seconds
Philosopher 2 is now HUNGRY and trying to pick up forks.
Philosopher 2 IS NOW EATING.
Philosopher 0 is now HUNGRY and trying to pick up forks.
Philosopher 0 IS NOW EATING.
Philosopher 1 is now HUNGRY and trying to pick up forks.
Philosopher 1 fails to pick up forks and then starts waiting.
Philosopher 3 is now HUNGRY and trying to pick up forks.
Philosopher 3 fails to pick up forks and then starts waiting.
Philosopher 4 is now HUNGRY and trying to pick up forks.
Philosopher 4 fails to pick up forks and then starts waiting.
Philosopher 2 returns forks and then starts TESTING 1 and 3.
Philosopher 3 IS NOW EATING.
Philosopher 0 returns forks and then starts TESTING 4 and 1.
Philosopher 1 IS NOW EATING.
Philosopher 3 returns forks and then starts TESTING 2 and 4.
Philosopher 4 IS NOW EATING.
Philosopher 1 returns forks and then starts TESTING 0 and 2.
Philosopher 4 returns forks and then starts TESTING 3 and 0.
rohan@rohan-VirtualBox:~/Desktop$
```

- E.g.

The Dining Philosophers Problem

- Hw4_{studentID}.rar:
 - Hw4.c(90%)
 - Hw4_report(10%)
 - Tell us how you implement your homework and show us your results.
- 0 will be given to cheaters, so don't copy & paste your friend's code directly.
- Deadline:5/28(TUE.) 23:59
- And, of course, we will pick $\frac{1}{4}$ of all students to demo in person.