

Practical 5th

Consider there are N philosophers seated around a circular table with one chopstick between each pair of philosophers.

- Each philosopher needs two chopsticks to eat.
- A philosopher may eat only if he can pick up both adjacent chopsticks.
- One chopstick can be picked up by only one of its adjacent philosophers at a time (not both).

Write a program to solve the problem using process synchronization technique.

CODE :

```
M /c/OS_Practical5TH
briti@LAPTOP-US9V80AL UCRT64 ~
$ gcc --version
gcc.exe (Rev8, Built by MSYS2 project) 15.2.0
Copyright (C) 2025 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

briti@LAPTOP-US9V80AL UCRT64 ~
$ cd /c

briti@LAPTOP-US9V80AL UCRT64 /c
$ mkdir OS_Practical5TH
cd OS_Practical5TH
mkdir: cannot create directory 'OS_Practical5TH': File exists

briti@LAPTOP-US9V80AL UCRT64 /c/OS_Practical5TH
$ mkdir OS_Practical 5
cd OS_Practical 5
-bash: cd: too many arguments

briti@LAPTOP-US9V80AL UCRT64 /c/OS_Practical5TH
$ nano dining.c

briti@LAPTOP-US9V80AL UCRT64 /c/OS_Practical5TH
$ gcc dining.c -o dining -pthread
```

```
M /c/OS_Practical5TH
GNU nano 8.7                                            DINING.C
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>

#define N 5

pthread_mutex_t forks[N];

void* philosopher(void* num) {
    int id = *(int*)num;

    printf("Philosopher %d is thinking\n", id);
    sleep(1);

    if (id == N - 1) {
        // Last philosopher picks right fork first
        pthread_mutex_lock(&forks[(id + 1) % N]);
        printf("Philosopher %d picked right fork\n", id);

        pthread_mutex_lock(&forks[id]);
        printf("Philosopher %d picked left fork\n", id);
    } else {
        // Others pick left fork first
        pthread_mutex_lock(&forks[id]);
        printf("Philosopher %d picked left fork\n", id);

        pthread_mutex_lock(&forks[(id + 1) % N]);
        printf("Philosopher %d picked right fork\n", id);
    }

    printf("Philosopher %d is eating\n", id);
    sleep(1);
}
```

```

pthread_mutex_unlock(&forks[id]);
pthread_mutex_unlock(&forks[(id + 1) % N]);

printf("Philosopher %d finished eating\n", id);

return NULL;
}

int main() {
    pthread_t thread_id[N];
    int i, phil[N];

    for (i = 0; i < N; i++)
        pthread_mutex_init(&forks[i], NULL);

    for (i = 0; i < N; i++) {
        phil[i] = i;
        pthread_create(&thread_id[i], NULL, philosopher, &phil[i]);
    }

    for (i = 0; i < N; i++)
        pthread_join(thread_id[i], NULL);

    return 0;
}

```

^G Help **^O** Write Out **^F** Where Is **^K** Cut **^T** Execute **^C** Location **M-U** Undo
^X Exit **^R** Read File **^L** Replace **^U** Paste **^J** Justify **^G** Go To Line **M-E** Redo
M-A Set Mark **M-6** Copy **M-B** Whe

OUTPUT :

```

priti@LAPTOP-US9V80AL UCRT64 /c/OS_Practical5TH
$ ./dining
Philosopher 0 is thinking
Philosopher 1 is thinking
Philosopher 3 is thinking
Philosopher 2 is thinking
Philosopher 4 is thinking
Philosopher 4 picked right fork
Philosopher 2 picked left fork
Philosopher 3 picked left fork
Philosopher 1 picked left fork
Philosopher 4 picked left fork
Philosopher 4 is eating
Philosopher 3 picked right fork
Philosopher 3 is eating
Philosopher 4 finished eating
Philosopher 0 picked left fork
Philosopher 3 finished eating
Philosopher 2 picked right fork
Philosopher 2 is eating
Philosopher 2 finished eating
Philosopher 1 picked right fork
Philosopher 1 is eating
Philosopher 0 picked right fork
Philosopher 0 is eating
Philosopher 1 finished eating
Philosopher 0 finished eating

```