

Practical 6

Aim : Considered there are N philosophers seated around a circular table with one chopstick between each pair of philosophers. There is one chopstick between each philosopher. A philosopher may eat if he can pick up the two chopsticks adjacent to him. One chopstick may be picked up by any one of its adjacent followers but not both. Write a program to solve the problem using process synchronization technique.

Code:

```
#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>


#define N 5

#define THINKING 0

#define HUNGRY 1

#define EATING 2


int state[N];

int phil[N] = {0, 1, 2, 3, 4};


sem_t mutex;

sem_t S[N];


#define LEFT (phnum + N - 1) % N

#define RIGHT (phnum + 1) % N
```

Practical 6

```
void test(int phnum)
```

```
{
    if (state[phnum] == HUNGRY &&
        state[LEFT] != EATING &&
        state[RIGHT] != EATING)
    {
        state[phnum] = EATING;
        sleep(1);
        printf("Philosopher %d takes chopsticks %d and %d\n",
            phnum + 1, LEFT + 1, phnum + 1);
        printf("Philosopher %d is Eating\n", phnum + 1);
        sem_post(&S[phnum]);
    }
}
```

```
void take_chopstick(int phnum)
```

```
{
    sem_wait(&mutex);
    state[phnum] = HUNGRY;
    printf("Philosopher %d is Hungry\n", phnum + 1);
    test(phnum);
    sem_post(&mutex);
    sem_wait(&S[phnum]);
}
```

```
void put_chopstick(int phnum)
```

```
{
```

Practical 6

```
sem_wait(&mutex);

state[phnum] = THINKING;

printf("Philosopher %d puts down chopsticks %d and %d\n",
      phnum + 1, LEFT + 1, phnum + 1);

printf("Philosopher %d is Thinking\n", phnum + 1);

test(LEFT);

test(RIGHT);

sem_post(&mutex);
}
```

```
void* philosopher(void* num)
{
    int* i = num;

    while (1)
    {
        sleep(1);

        take_chopstick(*i);

        sleep(1);

        put_chopstick(*i);
    }
}
```

```
int main()
{
    pthread_t thread_id[N];

    sem_init(&mutex, 0, 1);
```

Practical 6

```
for (int i = 0; i < N; i++)

    sem_init(&S[i], 0, 0);

for (int i = 0; i < N; i++)

{

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

    printf("Philosopher %d is Thinking\n", i + 1);

}

for (int i = 0; i < N; i++)

    pthread_join(thread_id[i], NULL);

return 0;

}
```

Output:

```
MINGW64/c/Users/user/Desktop/dining_philosophers
user@DESKTOP-J0D0N7F MINGW64 ~/Desktop (master)
$ pwd
/c/Users/user/Desktop
user@DESKTOP-J0D0N7F MINGW64 ~/Desktop (master)
$ mkdir dining_philosophers
cd dining_philosophers
user@DESKTOP-J0D0N7F MINGW64 ~/Desktop/dining_philosophers (master)
$ nano dining.c
user@DESKTOP-J0D0N7F MINGW64 ~/Desktop/dining_philosophers (master)
$ gcc --version
gcc.exe (x86_64-posix-seh-rev0, Built by MinGW-w64 project) 8.1.0
Copyright (C) 2018 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.
user@DESKTOP-J0D0N7F MINGW64 ~/Desktop/dining_philosophers (master)
$ ls
dining.c
user@DESKTOP-J0D0N7F MINGW64 ~/Desktop/dining_philosophers (master)
$ cat dining.c
user@DESKTOP-J0D0N7F MINGW64 ~/Desktop/dining_philosophers (master)
$ gcc dining.c -o dining -lpthread
c:/Program Files (x86)/CodeBlocks/MinGW/bin/./lib/gcc/x86_64-w64-mingw32/8.1.0/../../../../x86_64-w64-mingw32/lib/./lib/libmingw32.a(lib64_libmingw32_a-crt0_c.o):crt0_c.o:
ence to 'WinMain'
collect2.exe: error: ld returned 1 exit status
user@DESKTOP-J0D0N7F MINGW64 ~/Desktop/dining_philosophers (master)
$ nano dining.c
user@DESKTOP-J0D0N7F MINGW64 ~/Desktop/dining_philosophers (master)
$ gcc dining.c -o dining -lpthread
./dining
Philosopher 1 is Thinking
Philosopher 2 is Thinking
Philosopher 3 is Thinking
Philosopher 4 is Thinking
Philosopher 5 is Thinking
Philosopher 4 is Hungry
Philosopher 4 takes chopsticks 3 and 4
Philosopher 4 is Eating
Philosopher 3 is Hungry
Philosopher 5 is Hungry
```