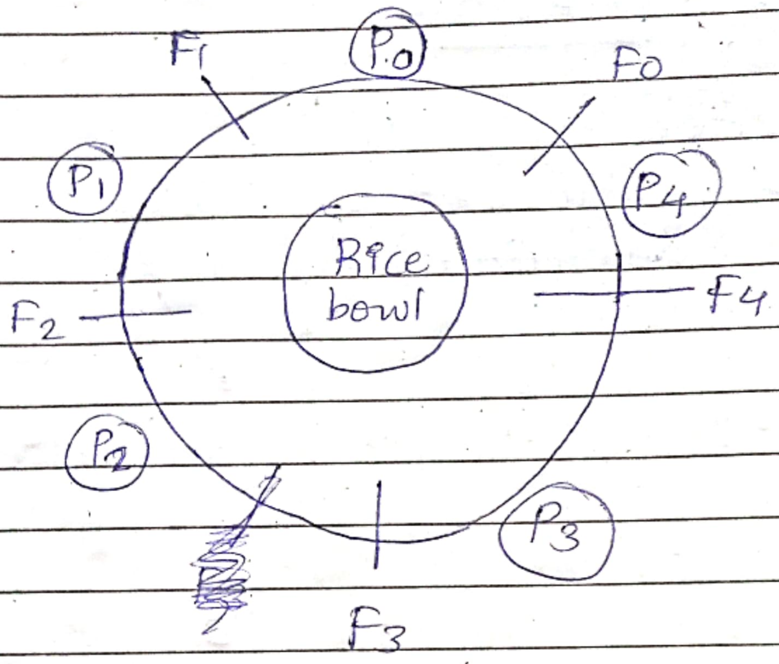
Page No.:
Date. / /

Dining Philosophers problem

Dining table → 5 philosophers
5 forks



```
void Philosopher(void) {
        while (true) {
            Thinking ();
            take_fork(i); ← left fork
            take_fork((i+1)%N); ← right fork
            EAT();
            put_fork(i);
            put_fork((i+1)%N);

        }
}
```

$$\begin{tikzcd} & \text{Think} \\ \text{Philosopher} \arrow[ur] \arrow[dr] & \\ & \text{Eat} \end{tikzcd}$$

Case 1: P0 comes

$i = 0$

left_fork = 0

Right fork = ((0+1) % N) = 1 (f_1)

Eating

f_0

f_1

end

~~Case 2~~: P_1 comes

$i = 1$

left_fork = 1 (f_1)

right_fork = 2 (f_2)

Eating

f_1

f_2

~~Case 3~~: Case 2:

P_0 picks f_0 & P_1 picks f_1

$$\begin{tikzcd}
 & P_0 \arrow[dl] \arrow[dr] & & P_1 \arrow[dl] \arrow[dr] & \\
f_0 & & f_1 \arrow[loop below] & & f_2
\end{tikzcd}$$

When P_1 completes eating then only P_0 would get the fork

problem of race condⁿ occurs

$S[i]$ → use array of semaphores

	S_0	S_1	S_2	S_3	S_4
Initially:	~~1~~ ~~0~~ 1	~~0~~ ~~1~~ 1	1	1	1

[Initialize every semaphore with 1 as when initialized with 0 it waits & gets blocked]

Code:

```
void philosopher(void){
    while (true){
        Thinking();
Entry {  Wait(table_fork(S[i])
         Wait(table_fork(S[(i+1) mod n])

        EAT();
        Signal(Put_fork(i));
        Signal(Put_fork((i+1)%N)

    }
}
```

$P_0 \rightarrow$	S_0	S_1
$P_1 \rightarrow$	S_1	S_2
$P_2 \rightarrow$	S_2	S_3
$P_3 \rightarrow$	S_3	S_4
$P_4 \rightarrow$	S_4	S_0

→ Case 1:

When P_0 comes S_0 & S_1 $(1 \rightarrow 0)$ & while going out $(0 \rightarrow 1)$

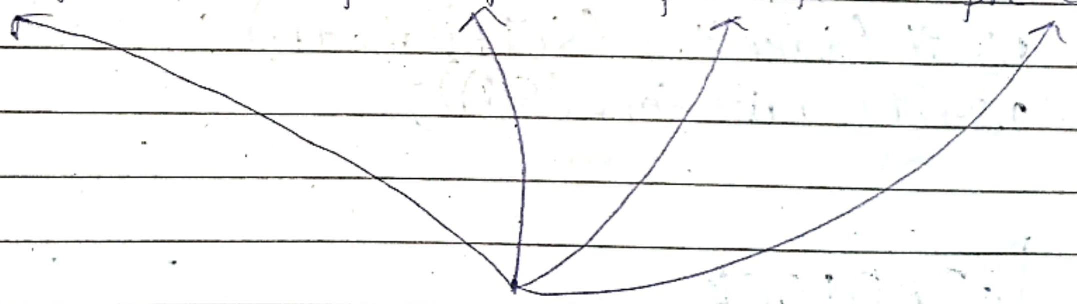
In DB P_0 & P_2 can come. It is a special case of mutual exclusion, as P_0 & P_2 are independent of each other.

Case 3: S_0 S_1 S_2 S_3 S_4
↓ ↓ ↓ ↓ ↓
1 1 1 1 1

P_0 comes first ~~($S_0 \to (0 \to 1)$~~)
$S_0 (1 \to 0)$
& gets pre-empted

P_1 comes P_2 comes P_3 comes P_4 comes

~~$S_1(\to$~~
$S_1 \to 0$ $S_2 \to 0$ $S_3 \to 0$ $S_4 \to 0$
pre-empt pre-empt pre-empt pre-empt



Blocked for right fork as the right fork is acquired by next philosopher

All philosophers take the left hand side fork but gets blocked for the right one. This situation is called deadlock (All processes get blocked)

Date / /

How to remove deadlock? On changing sequence of one process.

P_4 S_0 S_4

S_0	S_1	S_2	S_3	S_4
↓	↓	↓	↓	↓
$\cancel{1}$	$\cancel{1}$	$\cancel{1}$	$\cancel{1}$	1
0	0	0	0	

blocked P_4 S_0 S_4

We can change sequence of any philosopher

Nth philosopher

```
wait (take fork (S(i+1) mod N)
wait (take fork (S i));
```