Classical Problem of Synchronization

By
Dr. Upasana Pandey
Department of Computer Science & Engineering
IMS Engineering College (College Code:143)

Classical Problem of Synchronization

1. Producer Consumer Problem
2. Reader Writer Problem
3. Dining Philosopher Problem
4. Barber Shop Problem

1. Semaphores for Producer Consumer Problem

Buffer size n=5

1	2	3	4	5

Semaphore mutex=1
 	
Semaphore Empty=n (Total number of buffer, suppose n=5)
 	
Semaphore Full=0 (acts as counter to keep track for free buffer).
	void producer ()
	{
	while(true)
	{
	wait (Empty);
	wait (mutex);
	append();
	signal(mutex);
	signal(Full);
	}
	}
		void consumer()
		{
		while(true)
		{
		wait(Full);
		wait(mutex);
		take();
		signal(mutex);
		signal(Empty);
		}
		}

2. Semaphores for Reader Writer Problem

- A database is to be shared among several concurrent processes.
- Some processes may want only to read the database- Readers
- Other may want to update (read + write)-Writers
- Readers and writes are accessing a shared resource by the following rules:
  - Readers can read simultaneously.
  - Only one writer can write at any time.
  - When a writer is writing, no reader can read.
  - If there is any reader reading, all incoming writers must wait. Thus readers have higher priority.

2.Semaphores for Reader Writer Problem (cont.)

	Writer Process	Reader Process

Semaphore
mutex=1
Semaphore wrt=1
int readcount=0
	do
	{
	wait (wrt)
	write operation
	signal (wrt)
	} while (true);
		do
		{
		wait (mutex)
		readcount + +
		if (readcount==1)
		wait (wrt)
		signal (mutex)
		read operation
		wait (mutex)
		readcount - -
		(if readcount = = 0)
		signal(wrt)
		signal (mutex)
		} while (true);

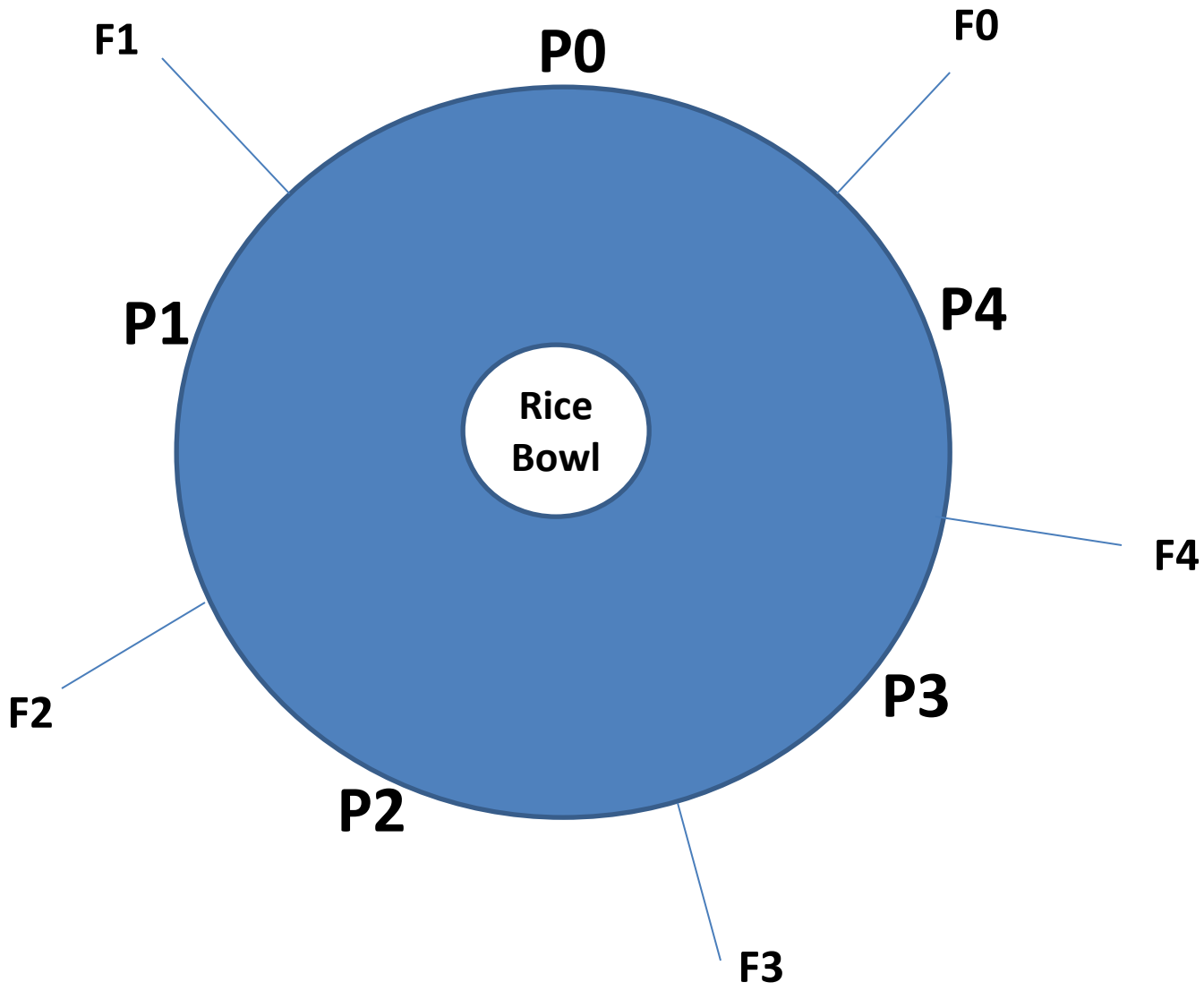
2. Semaphores for Reader Writer Problem (cont.)

- Mutual Exclusion satisfied.
- Progress satisfied.
- But violating bounded waiting: when a write comes in, it waits until no reader is reading.

3. Semaphores for Dining Philosopher Problem

- “Five philosophers sit around a circular table”.
- Each philosopher spend his life alternatively thinking and eating.
- In the centre of the table is a large plate of food.
- A philosopher needs two forks to eat.
- One fork is placed between each pair of philosopher and they agree that each will only use the fork to his immediate left and then right.
- There are five philosopher processes numbered 0 to 4. Fork is also numbered through 0 to 4.

3. Semaphores for Dining Philosopher Problem



3. Semaphores for Dining Philosopher Problem

```
Semaphore s[5];
void philosopher (void)
{
   while(true)
    {
      think();
      wait(takefork(Si)); // Left  Fork
      wait(takefork((Si+1)%5); //Right Fork
      eat();
      signal(putfork(Si));
      signal(putfork((Si+1)%5);
     }
}
```

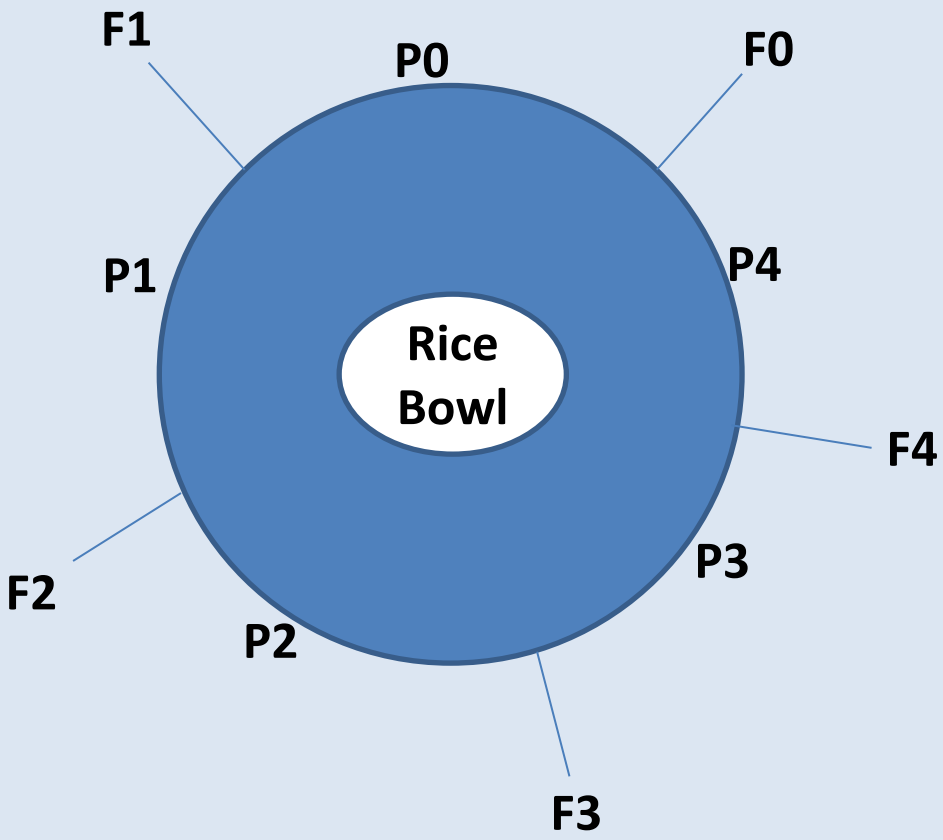
s[5]={1,1,1,1,1};
s[0]=1, s[1]=1, s[2]=1, s[3]=1, s[4]=1

P0	S0	S1
P1	S1	S2
P2	S2	S3
P3	S3	S4
P4	S4	S0

Outcome: Deadlock occurred.

3. Semaphores for Dining Philosopher Problem

Solution: P4 Philosopher first take right fork and then take left fork.



s[0]=1, s[1]=1, s[2]=1, s[3]=1, s[4]=1

P0	S0	S1
P1	S1	S2
P2	S2	S3
P3	S3	S4
P4	S0	S4

For Nth Process:

```
wait (takefork(i+1%5));
// Right Fork
wait (takefork(i));//Left Fork
```

Thank you