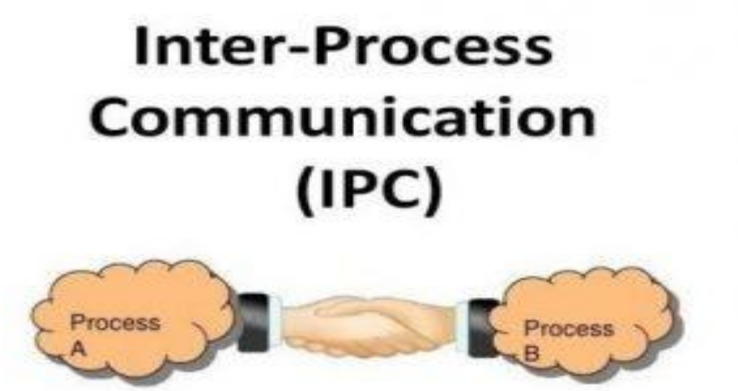
AMIRAJ

COLLEGE OF ENGINEERING & TECHNOLOGY

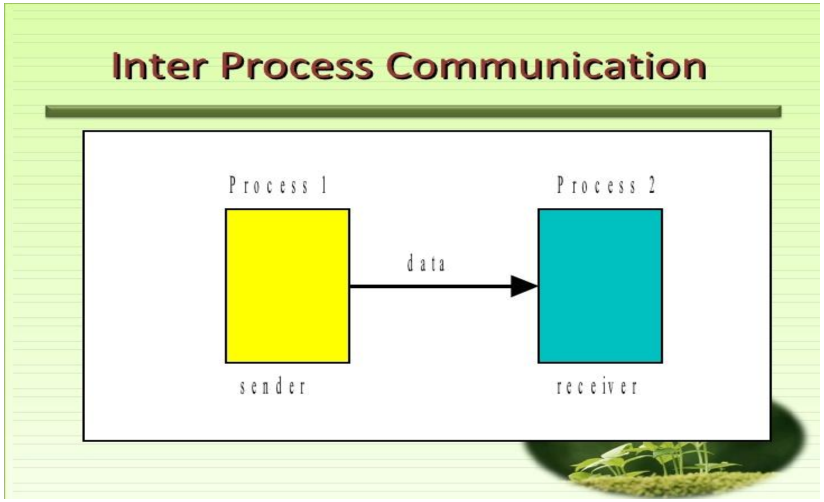
CHAPTER -4
INTER PROCESS COMMUNICATION

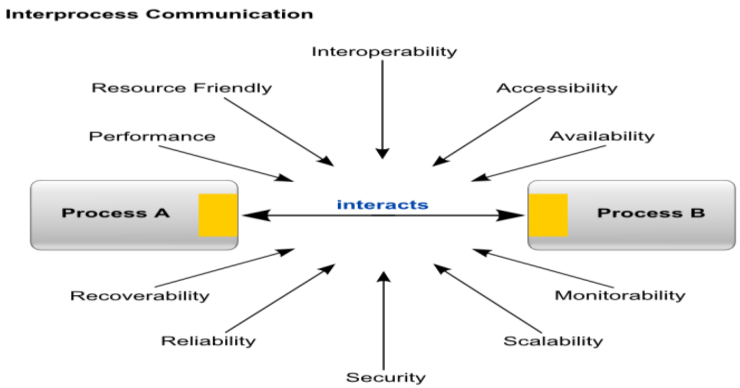


Subject:- OS Code:-3140702	Prepared by: Asst.Prof.Foram Patel (Computer Department,ACET)	AMIRAJ COLLEGE OF ENGINEERING & TECHNOLOGY

INTER PROCESS COMMUNICATION

- To allow one process to communicate with another process.
- Processes may be running on one or more computers connected by network.
- Client server computing always uses IPC.
- IPC facility provides two operation: send & receive
- A process can be of two types:
- Independent process, co-operating process.

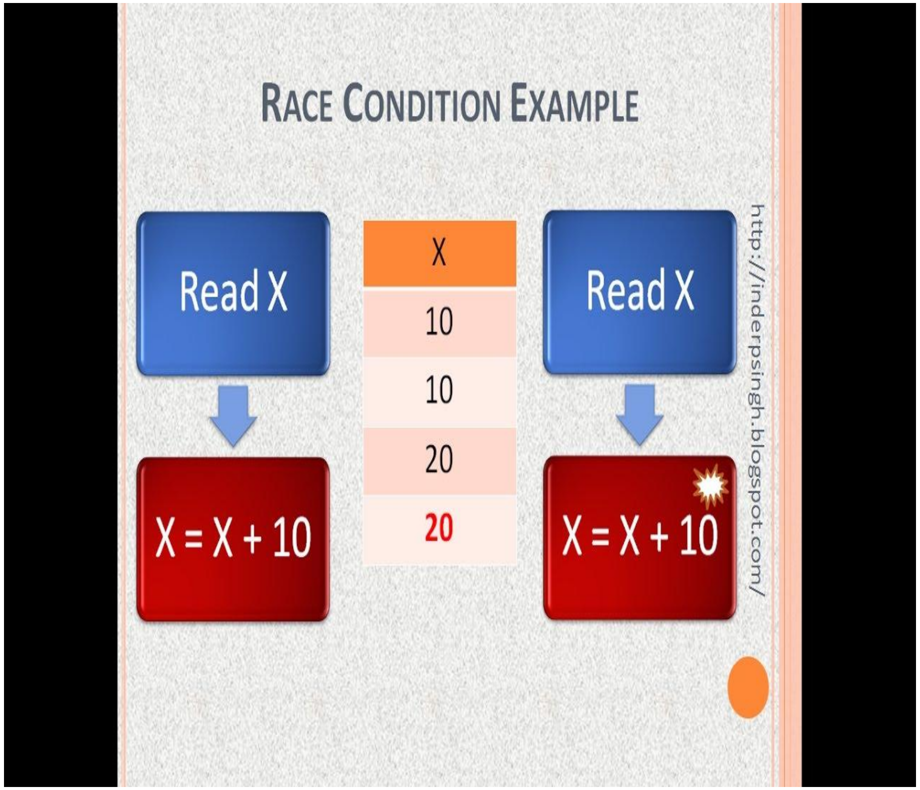




AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

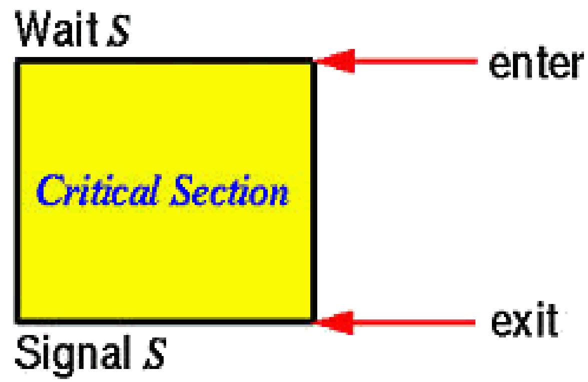
RACE CONDITION

- A race condition is a situation that may occur inside a critical section.
- It is a situation in which concurrently executing processes accessing the same shared data
- A race condition is an undesirable situation that occurs when a device or system attempts to perform two or more operations at the same time.
- Access and manipulate shared data concurrently.



CRITICAL SECTION

- When one process is in a critical section , all other processes are excluded from their critical section.
- Each process must ask permission to enter critical section in entry section , may follow critical section with exit section , then reminder section.




```
do {
      entry section
            critical section
      exit section
            remainder section
   } while (TRUE);
```

STRICT ALTENATION

- Strict alternation using a variable turn that can take the values 0 or 1.
- Turn variable is a synchronization mechanism that provides synchronization among two processes.
- Initial flag=0, if flag value is 0 then process P0 enters its critical section & P1 is waiting at this time.
- Po finish its critical section come out from it, set flag 0 to 1.
- flag=1 then p1 enter into critical section , come out from it , set 1 to 0.

```
while(TRUE) {
  while(turn != 0);
  critical_region();
  turn = 1;
  noncritical_region();
}
```

```
while(TRUE) {
  while(turn != 1);
  critical_region();
  turn = 0;
  noncritical_region();
}
```

PETERSON'S SOLUTION

Process 0

```
#define  N   2
int turn;  /* should be shared */
int interested[N];

void main()
{

        while(1)
        {
                noncritical_region();

                enter_region(0):
                critical_region();
                leave_region(0);

                noncritical_region();
        }
}
```

Process 1

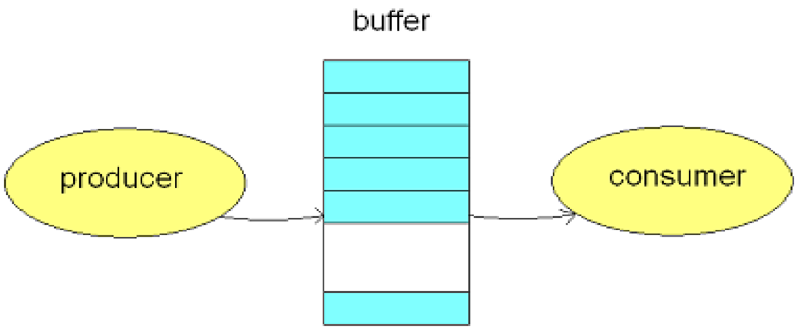
```
#define  N   2
int turn; /* should be shared */
int interested[N];

void main()
{

        while(1)
        {
                noncritical_region();

                enter_region(1):
                critical_region();
                leave_region(1);

                noncritical_region();
        }
}
```

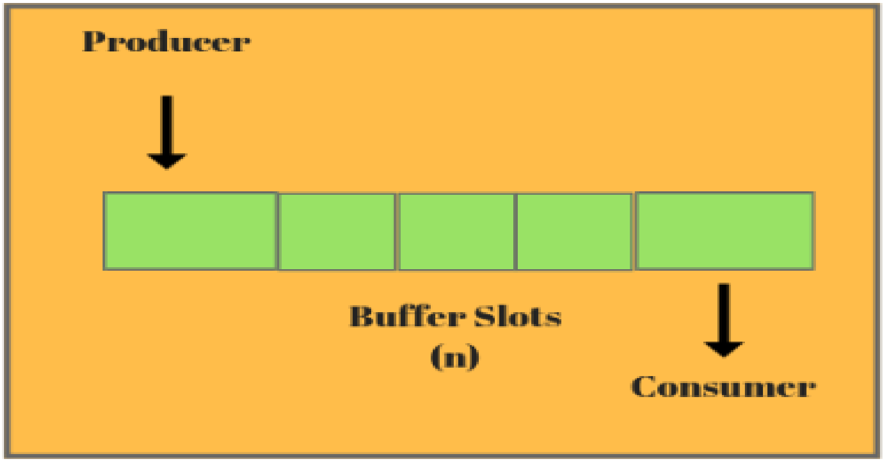
- Peterson's algorithm is a concurrent.
- It uses only shared memory for communication.
- It is a classic software based solution to the critical section problem.
- Peterson's algorithm is a concurrent programming algorithm for mutual exclusion that allows two or more processes to share a single-use resource without conflict

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

PRODUCER CONSUMER PROBLEM

- The producer consumer problem is a synchronization problem.
- There is a fixed size buffer and the producer produces items and enters them into the buffer.
- The consumer removes the items from the buffer and consumes them.
- The producer consumer problem can be resolved using semaphores.
- The producers and consumers share the same memory buffer that is of fixed-size.





AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

PRODUCER CONSUMER PROBLEM

Producer:

- Producer first waits until there is atleast one empty slot.
- There will now be one empty slot, producer is going to insert data in to buffer.
- After performing the insert operation, the lock is released and the value of full
- When producer is full, then producers go to the sleep.

```
while (true) {
    /* produce an item in next produced */

    while (counter == BUFFER SIZE) ;
        /* do nothing */
    buffer[in] = next produced;
    in = (in + 1) % BUFFER SIZE;
    counter++;
}
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

PRODUCER CONSUMER PROBLEM

Consumer:

•The consumer waits until there is atleast one full slot in the buffer.

•consumer completes the removal operation so that the data from one of the full slots is removed.

•Then, the consumer releases the lock.

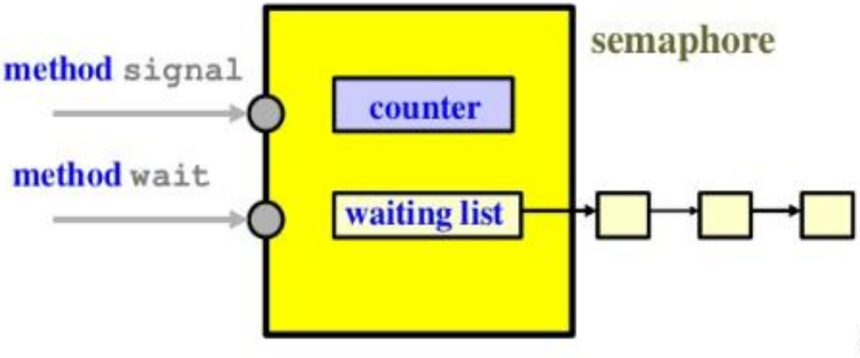
•consumer has just removed data from an occupied slot, thus making it empty.

```
while (true) {
    while (counter == 0); /*do nothing*/
    next consumed = buffer[out];
    out = (out + 1) % BUFFER SIZE;
    counter--;
    /*consume the item in next consumed*/
}
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

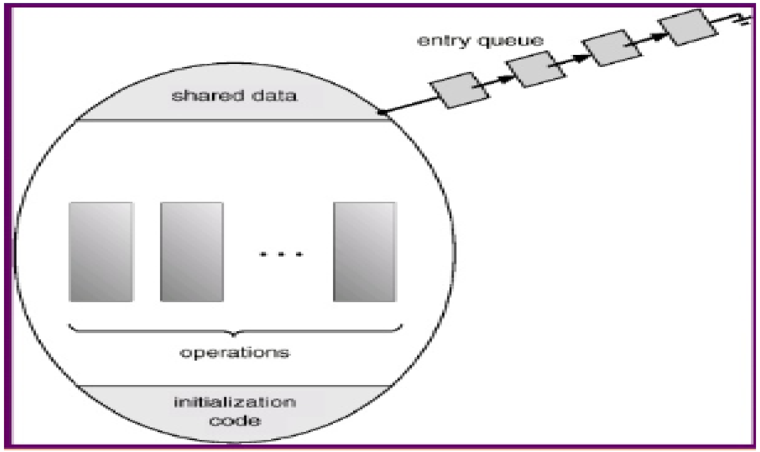
SEMAPHORES

- A semaphore is an object that consists of a counter, a waiting list of processes and two methods : signal & wait.
- Semaphore is a synchronization tool which can be used to deal with the critical section problem.
- It is a protected variable whose value can be accessed and altered only by the operation P & V.



MONITORS

- Monitor is a highly structured programming language construct.
- Only one process may be active within the monitor at a time.
- Private variables and Private procedures - Use within a monitor.
- Monitors have no public data.




```
monitor Monitor-Name
{
    local variable declarations;

    Procedure1 (...)
    { // statements };
    Procedure2 (...)
    { // statements };
    // other procedures
    {
      // initialization
    }
}
```

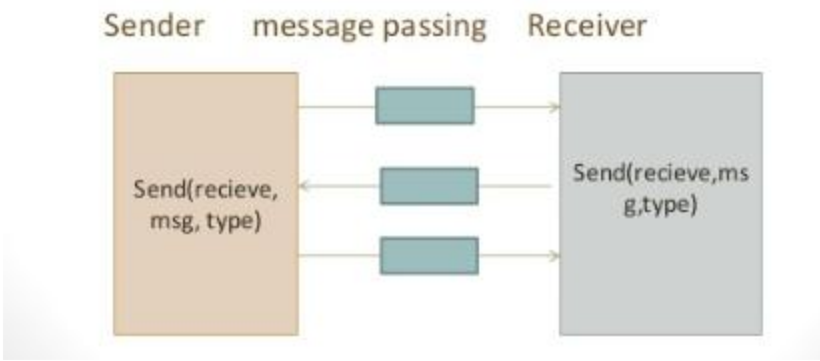
AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

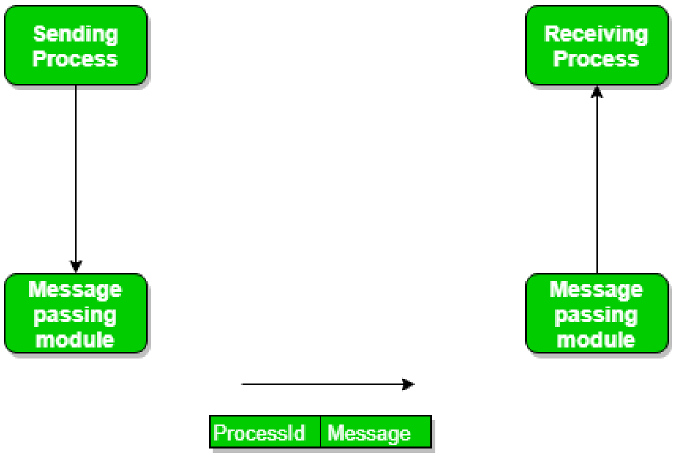
MESSAGE PASSING

- Message passing is the basis of the most inter-process communication in distributed system.
- It requires the programmer to know

1) Message
2) Name of source
3) Destination process

- OS send() system call to pass message to kernel. After execution user process waits for result with receive().





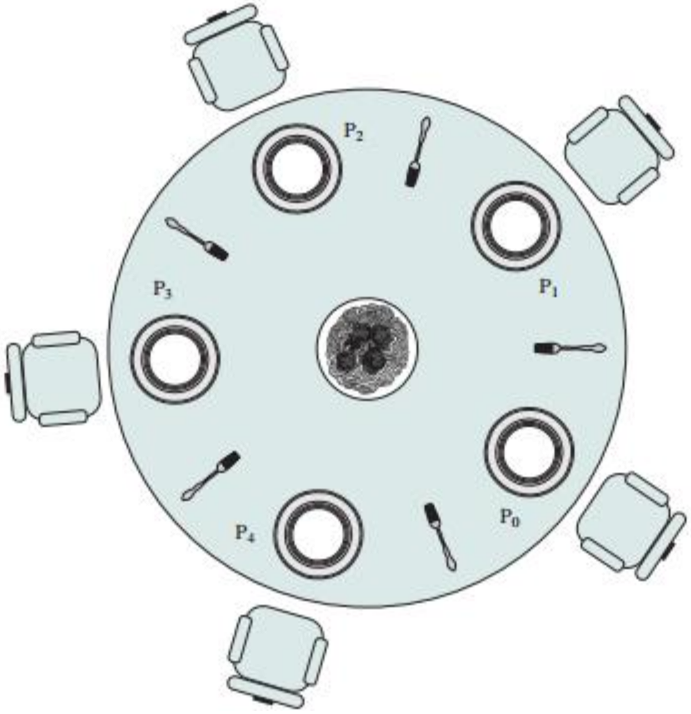
AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

READER'S & WRITER PROBLEM

- If one of the people tries editing the file, no other person should be reading or writing at the same time, otherwise changes will not be visible to him/her.
- However if some person is reading the file, then others may read it at the same time.
- When data items shared among the reader & writer single read and write operation perform at the same time then problem accure.
- **Reader:** Only read data they do not perform any update.
- **Writer:** Can both read and write.

DINNING PHILOSOPHER PROBLEM

- 5 philosopher are sitting around a circular table.
- Dinning table has 5 forks & bowl of rice in the middle.
- Philosopher either eat or think.
- Philosopher can pick up only one fork at a time.
- When philosopher want to think he/she keeps down both fork.



DINNING PHILOSOPHER PROBLEM

- Solution:
- No two neighbouring philosophers can eat at the same time.
- There should be at most four philosophers on the table.
- An even philosopher should pick the right chopstick and then the left chopstick while an odd philosopher should pick the left chopstick and then the right chopstick.

```
#define N 5

void philosopher(int i){
 while(TRUE){
     think();
     take_fork(R_i);
     if (available(L_i){
       take_fork(L_i);
       eat();
       put_fork(L_i);
       put_fork(R_i);
     }else{
       put_fork(R_i);
       sleep(random_time);
   }
}
```

AMIRAJ COLLEGE OF ENGINEERING & TECHNOLOGY