CS & IT ENGINEERING

Operating Systems

Miscellaneous Topics

Lecture No. 2

By- Dr. Khaleel Khan Sir

- Monitors
- Shared Pages

Monitors

B. Hansen

(CLASS): Abs/Encap.

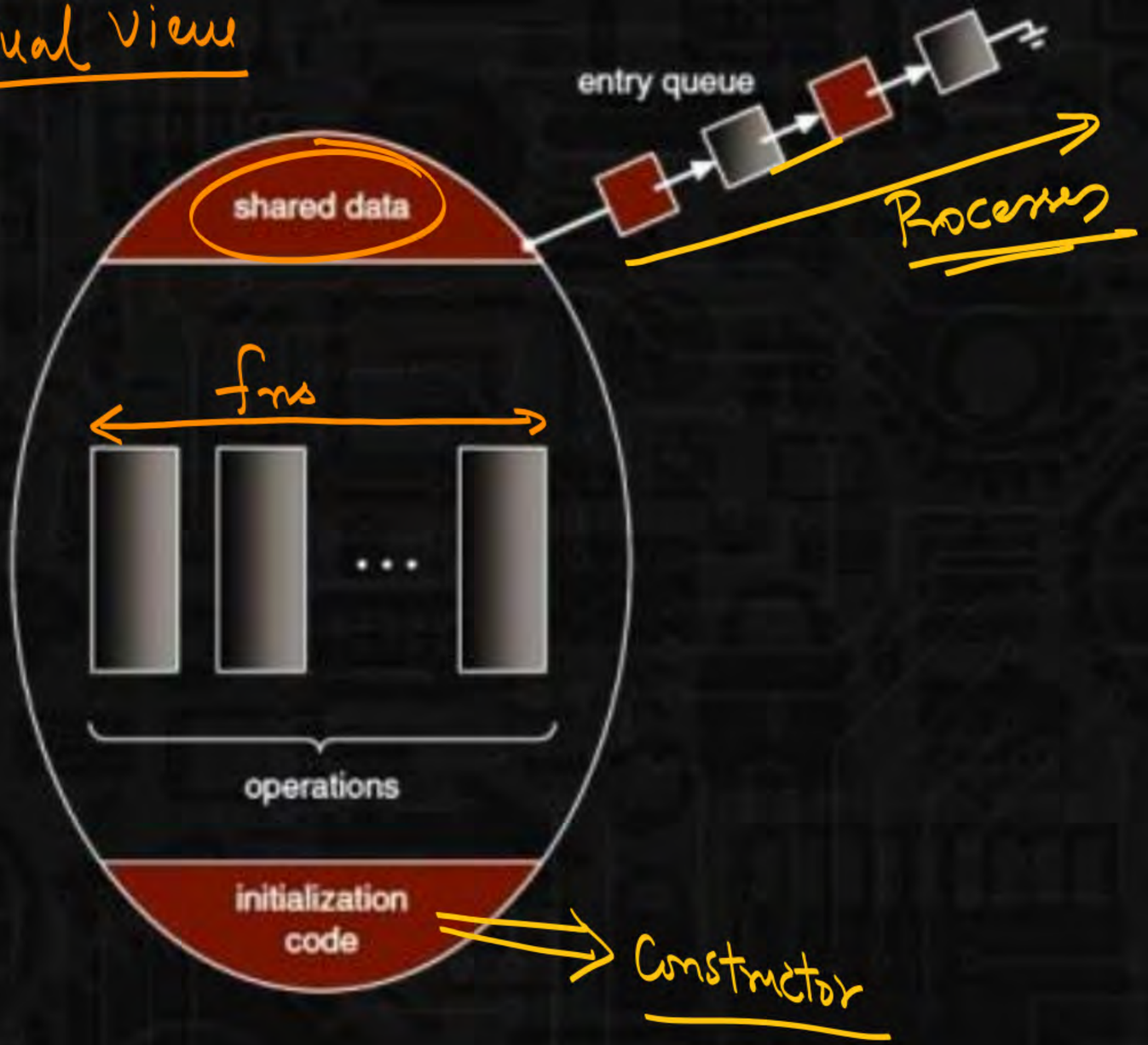
① : is a collection of Procedures, variables and data Structures that are all grouped together in a special kind of Module / Package

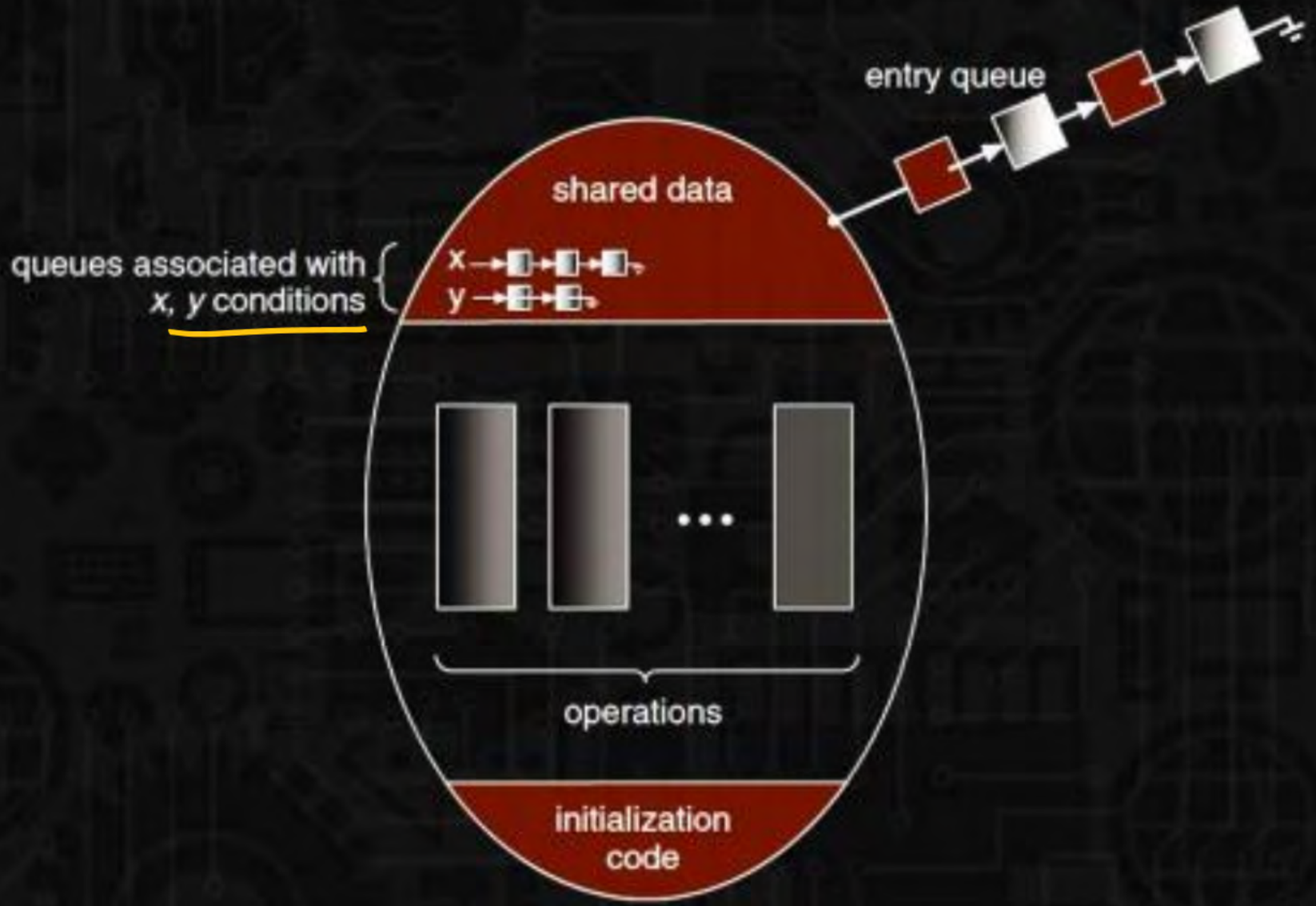
→ Procedures that runs outside the monitor cannot access Monitor Member Variables (Data) (PVT), but they can access Monitor Member functions;

→ Monitors Support Spl. Condition variables, that has Wait & Signal opn's to meet the Synch. requirements

→ only one Process can be active in Monitor at any-time

Conceptual View




entry queue
shared data
queues associated with
x, y conditions
x
y
operations
initialization
code

```
monitor monitor_name
{
   /* shared variable declarations */
   function P1 (. . .) {
   } . . .
   function P2 (. . .) {
   } . . .
                .
                .
                .
   function Pn (. . .) {
   . . .
   }
   initialization_code (. . .){
   } . . .
}
```

DINING - PHILOSOPERS : MONITORS

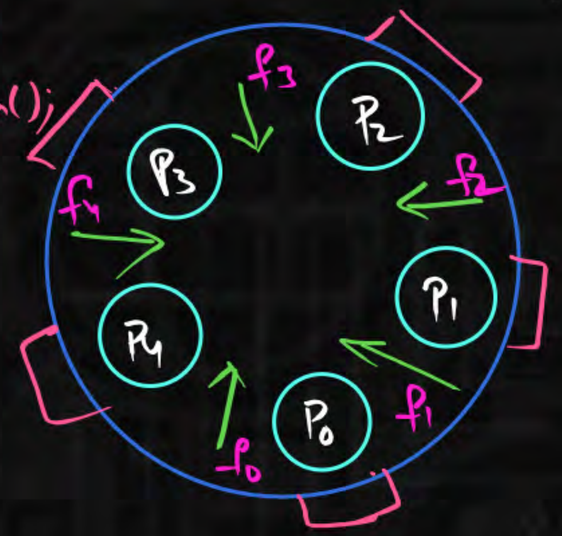
```
monitor DiningPhilosophers
{
        enum {THINKING, HUNGRY, EATING} state[5];
        condition self [5];
        void pickup(int i)
        {
                state[i] = HUNGRY;
                test(i);
                if (state[i] != EATING)
                self[i].wait();
        }
        void putdown(int i) {
                state[i] = THINKING;
                test((i + 4) % 5);
                test((i + 1) % 5);
        }
        void test(int i) {
                if ((state[(i + 4)% 5]!= EATING) &&
                (state[i] == HUNGRY) &&
                (state[(i + 1) % 5]!= EATING)) {
                        state[i] = EATING;
                        self[i].signal();
                }
        }
  initialization_code() {
 for(int i = 0; i < 5; i++)
 state[i] = THINKING;
  }
}
```

Enum{THINKING, HUNGRY, EATING}state[5];
Condition self[5];

wait/Signal

N=5

```
Pickup();
<eat>
putdown();
```



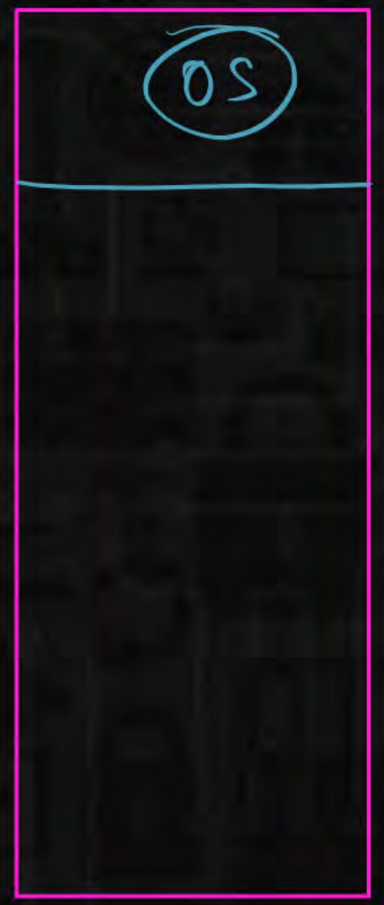
Producer-Consumer Using Monitors

```
Monitor ProducerConsumer
{
    int N=100, Count, Buffer[N], in=0, out=0;
    Condition Empty, full;

    Procedure Enter(itemp);
    {
        a) if (Count == N)
                full.wait();
        b) Buffer[in] = itemp;
        c) in = (in+1) % N;
        d) Count = Count+1;
        e) if (Count == 1) Empty.signal();
    }

    Procedure Remove(itemc)
    {
        a) if (Count == 0)
                Empty.wait();
        b) itemc = Buffer[out];
        c) out = (out+1) % N;
        d) Count = Count-1;
        e) if (Count == N-1) full.signal();
    }

    init:
        Count = 0;
}
```

```
Procedure Producer()
{
    int itemp;
    while(1)
    {
        itemp = Produce();
        ProducerConsumer.Enter(itemp);
    }
}
```

Shared Pages : for Memory optimization

M.Pr. Time Shared O.S (UNIX) → V_i



P_1 P_2 . . P_n

(Each user/appl. needs to have its own copy of Editor)

→ (If the code Pages are Read only/Non-Modifiable is Re-Entrant)

Shared Pages

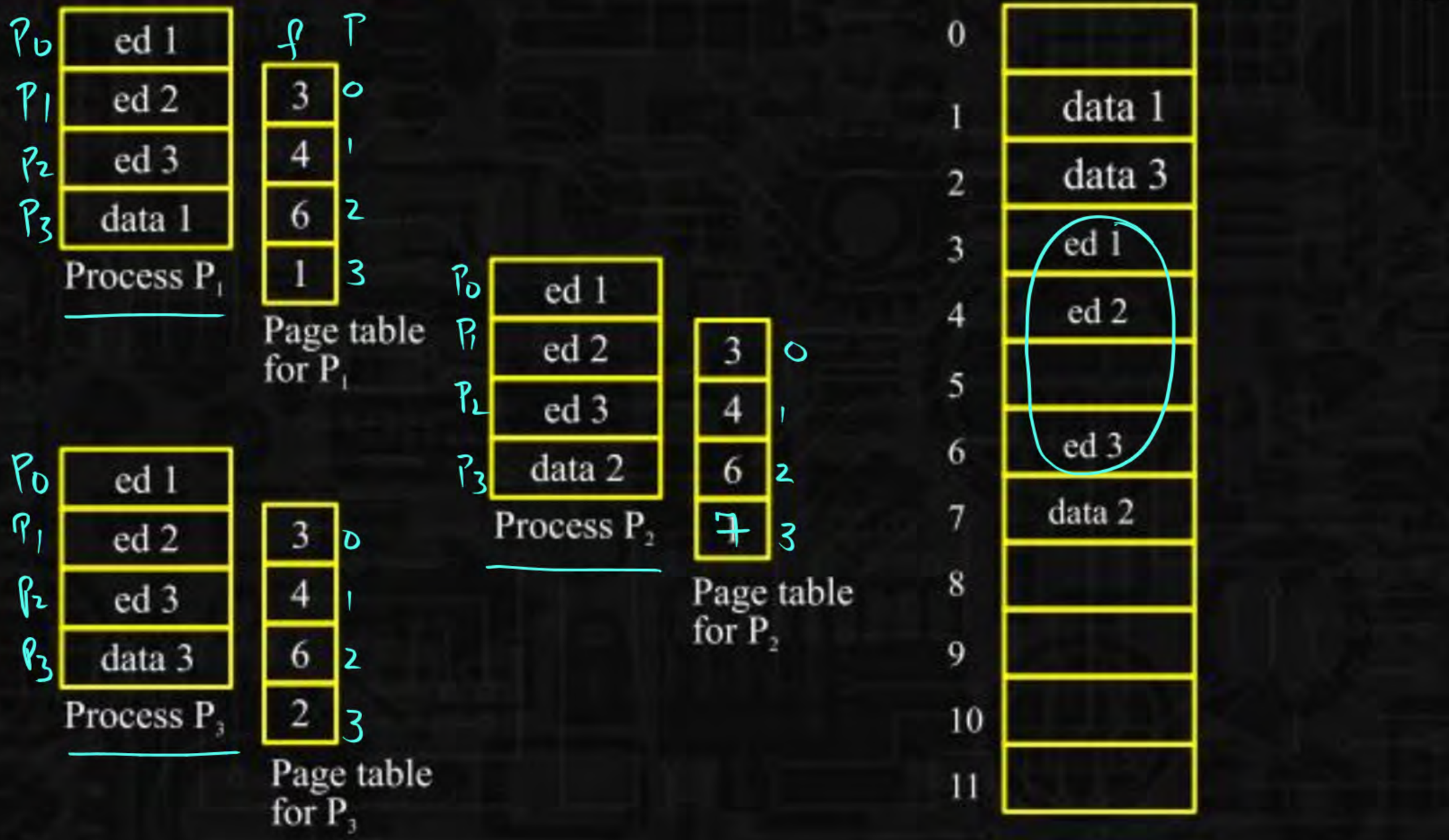
Shared code

- One copy of read-only (reentrant) code shared among processes (i.e., text editors, compilers, window systems).
- Shared code must appear in same location in the logical address space of all processes

Private code and data

- Each process keeps a separate copy of the code and data
- The pages for the private code and data can appear anywhere in the logical address space

Shared Pages Example



1. Prepare the Long / Short Notes (Read one Text-Book)
2. Important Formulas / Points
3. Typical Problems
4. PYQ's (leftover)

5. Revision on regular basis
6. Chapter exercise questions

7. Test Series
   - Time Mgmnt.
   - Coverage of Syllabus

Don'ts
1) NO Comparism
2) Unauthentic videos / Content


THANK YOU GW
SOLDIERS !