SHIVANI PATEL(OPERATING SYSTEM-NOTES)

(AT INTERVIEW TIME)

SNO.	TOPICS
1.	OPERATING SYSTEM
2.	FUNCTIONALITY OF OPERATING SYSTEM
3.	MICROPROCESSOR
4.	PROCESS
5.	THREAD
6.	TYPES OF THREAD
7.	DIFFERENCE BETWEEN MULTIPROCESSING AND MULTITHREADING
8.	JOB SCHEDULING
9.	CPU SCHEDULING
10.	SEMAPHORES AND MUTEX
11.	PRODUCER CONSUMER PROBLEM
12.	DEADLOCK
13.	PAGING
14.	PROCESS SYNCHRONIZATION

Date __/__/_____

otes

OPERATING SYSTEM (OS)

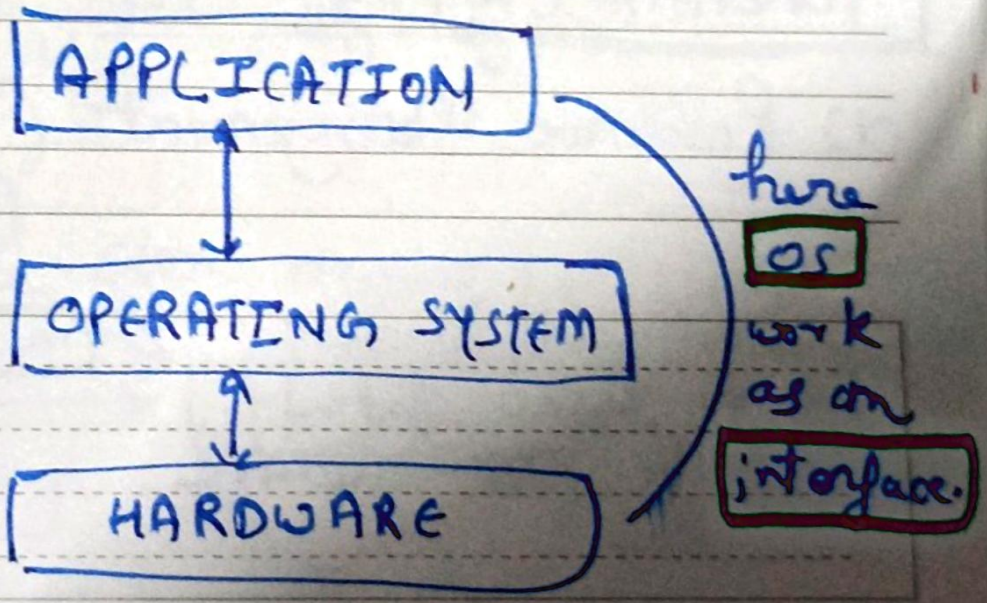
What is ??

Operating System is a **system** software

It is **inside** your system.

It works / providing **interaction** between **user** and **hardware.**

user1 user2 .... usern



Remarks

Scanned by Scanner Go

Date__/__/_____ Notes

Example of OS: Windows, Linux, Android etc.

⇒ Primary goal of OS:

↓

provide convenience to user

⇒ for maximum throughput

↓

Use linux

Functionality of OS

(1) Resource Management (require in parallel access)

Remarks multiple user try to access data with same device ← means

Scanned by Scanner Go

Notes Date __/__/____

(2) Process Management (CPU Sheduling algorithm)

means

If we want to execute the process then how to execute that on CPU

for that

uses CPU scheduling algo.

(3) Store management (Secondary device)

that is

Hard-disk

how execute store management

through → file system

Remarks

Scanned by Scanner Go

Date__/__/_____ Notes

(4) Memory Management (RAM)

has constraints

that is limitation of size

↓

when it comes in the **RAM** then we **transfer** it to CPU one by one.

↓ we did here

Allocation / deallocation

↓ reserve some memory for the execution of program.

↓ way to free the RAM of finished processes.

Remarks

Scanned by Scanner Go

Notes

Date__/__/_____

(5) Security (authenticate user can access it)

↓

window uses Kerberos security protocol.

Operating System

↓ works through

System Call

→ that are

↓

Open, write, read etc.

Remarks _____

Scanned by Scanner Go

Date__/__/_____ Notes

Microprocessor (OPERATING SYSTEM)

↓ What is ??

It is a program that enables the computer hardware to communicate and operate the computer software.

↓

It is the brain of computer, which does all the work. It is a processor that incorporates all the fn of CPU on a single IC (Integrated circuit)

How It works → ??

Remarks: First, it takes in input perform some arithmetic and logical

Scanned by Scanner Go

otes Date__/__/_____

operations over it and produces the desired output.

Secondly, it is a digital device on a chip so it can fetch instructions from memory, decode and execute them

give output as result.

Components of microprocessor —

- Control unit
- I/O units
- Arithmetic logic unit
- Registers
- Cache

Remarks

Scanned by Scanner Go

Date__/__/_____ Notes

Microprocessor

↓

data processing logic + control unit

= Single integrated chip.

Example of microprocessor

- VCRs
- DVD players
- ovens
- stoves etc.

Note: For more depth, of microprocessor that come under computer of architecture so I will cover there as in depth, in OS it is sufficient.

Remarks

Scanned by Scanner Go

Notes Date__/__/_____

Process in Operating System

It is a heavy weight task.

It is an entity which represents the basic unit of work to be implemented in the system.

Process states

Primary state

- New
- Ready
- Running
- Terminated
- Wait/Block

Remarks

Scanned by Scanner Go

Date__/__/_____ Notes

New → means creating a process let's suppose that you wrote a program in C lang.

↓ store it in Secondary memory

after store we pop it into Ready state

↓

Store inside RAM

dispatch

Running state (CPU) ← our process

↓ still in RAM but its status change.

after execution its terminated the process.

↓ means

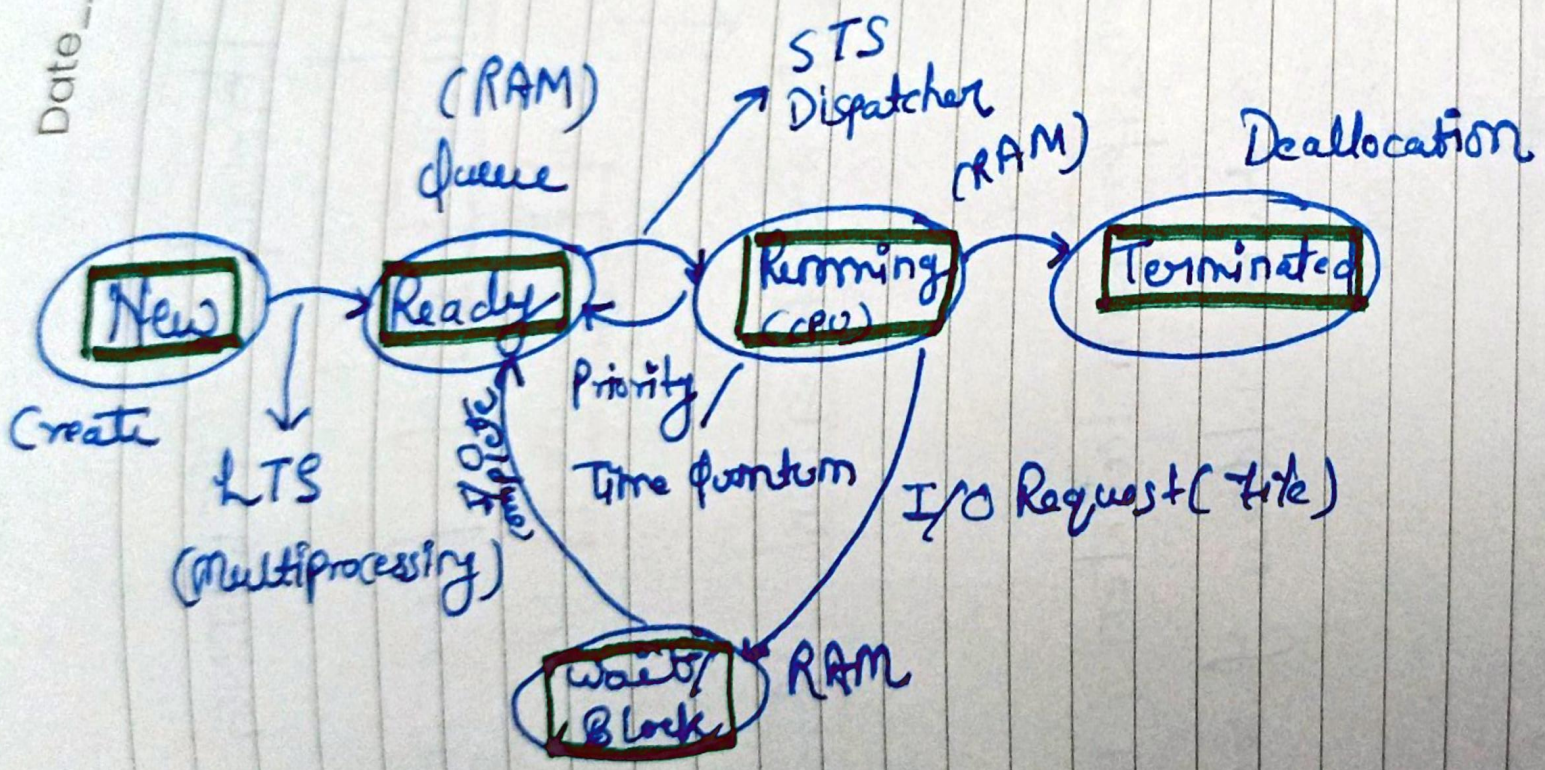
deallocation

Remarks

Scanned by Scanner Go

LTS (long term scheduler)

STS (short term scheduler)



Date __/__/____

Remarks

www.lawrencemayo.ac.in

Scanned by Scanner Go

Date__/__/_____ Notes

Thread (OPERATING SYSTEM)

↓

Smallest unit of processing that can be performed in OS.

↓

Thread exists within a process.

↓

has its own program counter (like it contain address of next instruction)

↓

Advantages of Thread.

- Responsiveness
- Faster Context Switch
- effective utilization
- Resource Sharing

Remarks

Scanned by Scanner Go

Types of Thread

User Thread

- Implemented by **users.**
- implementation is **easy**
- designed as **independent** thread.

Imp

Example — Java thread, POSIX thread.

Kernel Thread

- Implemented by **OS.**
- recognised by **OS**
- it is **complicated**
- Hardware support is needed
- If one kernel thread perform **blocking operation** then other thread can **continue** their execution.

example window solaris

Remarks

Scanned by Scanner Go

Date__/__/_____ Notes

Multiprocessing And Multithreading

Multiprocessing

↓

CPUs are added for increasing computing power.

↓

many processes are executed simultaneously

↓

Classified into Symmetric and Asymmetric.

↓

every process owned a separate address space.

Multithreading

↓

many threads are created of a single process for increasing computing power.

↓

many thread of a process are executed simultaneously.

↓

not classified in any categories.

↓

common address space is shared by all the threads.

Remarks

Scanned by Scanner Go

otes Date__/__/_____

Job Scheduling (OPERATING SYSTEM)

↓

Process of allocating system resources to many different tasks by an (OS).

↓

Different tasks get executed at pre-determined time or when the right event happens.

↓

This type of scheduling makes sure that all jobs are carried out fairly and on time.

Remarks

Job scheduling

↓

Performed using job schedulers.

Scanned by Scanner Go

Date__/__/_____ Notes

CPU Scheduling Algo

↓

what is cpu scheduling ?

↙

that allows one process to use the CPU while the execution of another process is on hold

↘

Due to unavailability of any resource like I/O etc.

↙

make the system efficient, fast and fair.

Remarks

Scanned by Scanner Go

Notes Date__/__/_____

CPU Scheduling

- Arrival time
- Burst time
- Completion Time
- Turn Around time
- Waiting time
- Response time

Arrival time: time at which process enter the ready queue or state.

Burst time: Time required by a process to get execute on CPU.

Completion time: the time at which process complete its execution.

Remarks Turn Around Time:

Complete Time − Arrival Time

Scanned by Scanner Go

Date__/__/_____ Notes

waiting time = T.A.T - Burst Time

Response time ⇒ (time at which a process get CPU first time) - (A.T)

Types of CPU scheduling

Non-preemptive

↓

Used when a **process terminates.**

↓

Switches from **running** to the waiting state.

Remarks

↓

ex: SJF (shortest Job first)

Preemptive

↓

Used when process **switches** from **waiting** to **ready** state.

↓

resources are **allocated** to the **process** for a limited amount of **time.**

ex → RR, SRTF

Scanned by Scanner Go

Notes

Date__/__/_____

Scheduling Algorithm

↓

decide which process to execute first and which process to execute last to achieve maximum utilization,

↓ ↓ ↓ ↓

FCFS SJF SRTF RR

these four are mainly used.

FCFS: first come First Serve

SJF: shortest - Job - first

SRTF: shortest Remaining time first

RR: Round Robin

Remarks

Scanned by Scanner Go

Date__/__/_____ Notes

FCFS:

↓

Just like **queue** data structure.

↓

used in **batch** systems.

↓

easy to **understand** and **implement**

where a **new** process enters through the **tail** of the queue, and the **scheduler** selects process from the **head** of the queue.

SJF:

works on the process with the **shortest burst time**.

↓

Best approach to **minimize** waiting time.

Remarks

Scanned by Scanner Go

Notes Date__/__/_____

Round Robin Scheduling

↓

designed for time-sharing systems.

↓

this algo is similar to FCFS scheduling but in Round Robin (RR).

↓

time is alotted to each process called a quantum.

SRTF

↓

is a scheduling method that is a preemptive version of shortest job next scheduling.

↓

smallest amount of time remaining until completion is selected to execute.

Remarks

Scanned by Scanner Go

PAGE NO.
DATE : / /

Semaphore (OPERATING SYSTEM)

It is a method or it is tool which is used to present race condition.

race → means when we synchronize processes or run multiple processes at a time

→ specially with co-operative process.

Something is common means

→ due to race condition, the problem occur may be loss of data, deadlock etc.

for remove — OR — for prevent

One of the method for that is semaphore.

Scanned by Scanner Go

PAGE NO.
DATE: / /

Semaphore

It is a **integer** variable which is used in a **mutual** exclusive manner by various **concurrent** co-operative processes.

Two types of Semaphore

- **Counting**
  - value can vary from (−∞ to ∞)
- **Binary**
  - Value can vary from (0,1)

Working of ~~counting~~ Semaphore —??

critical section (CS) → whatever is **common** in **piece of code** that in **critical** section.

Scanned by Scanner Go

PAGE NO.
DATE : / /

Entry code run first

CS

↓

Exit section

Operation on entry code and exit section

↓

P(), Down, Wait → Entry code

V(), up, signal, Post, Release → Exit code

Mutex (OPERATING SYSTEM)

↓

It is a mutual exclusion object that synchronize access to a resource.

↓

locking mechanism used to synchronize access to a resource.

↓

only one process can work on a particular

Scanned by Scanner Go

PAGE NO.
DATE : / /

code section at a time.

Use of mutex —— ??

↓

if multiple threads want to access the critical section, mutex allows only one thread at a time to be in critical section.

↓ means —— ??

P1 P2 P3 P4 ——→ not X

↓

P1 P2 P3 P4 ——→ not X

P1 P2 P3 P4 ——→ this is critical section

Key Points: With the help of mutex, solve {"Producer Consumer Problem"}

Scanned by Scanner Go

PAGE NO.
DATE : / /

Difference b/w mutex and semaphore

⇒ Mutex is used for thread but semaphore is used for process.

⇒ mutex is work in userspace but semaphore is work in kernel space.

⇒ mutex is locking mechanism ownership method but semaphore is signalling mechanism.

⇒ thread to thread mutex is used but process to process locking mechanism semaphore is used.

Scanned by Scanner Go

PAGE NO.
DATE : / /

Producer Consumer Problem

(OPERATING SYSTEM)

⇓

It is a standard problem of multi-process synchronization.

⇓

means,

in this, we have two processes here, one is producer and one is consumer.

assumption we take that is

both process occur at same time

both are parallel processes
And they are sharing something.

CODE

Producer

Consumer

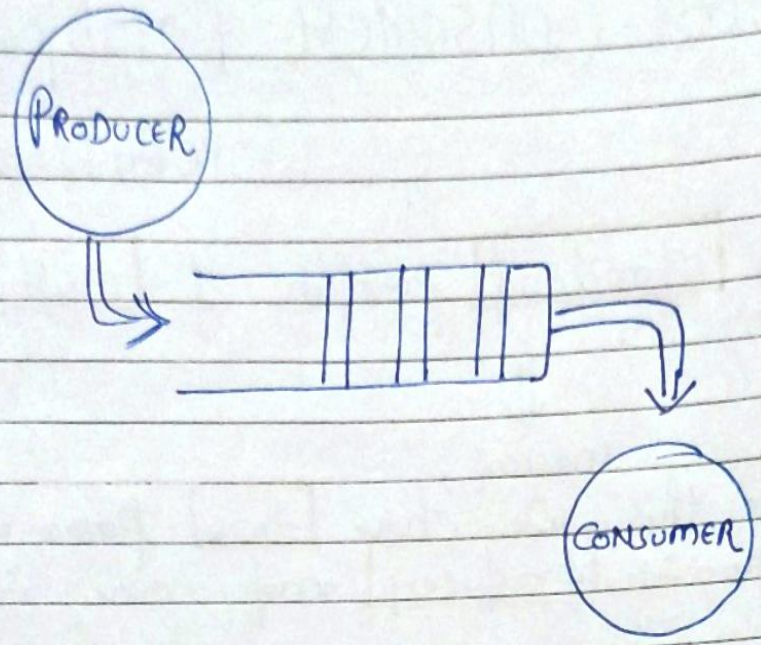
It produces an item, place it in a buffer.

it consumes the item, and it will execute this whole code.

And finally bring out item from buffer

Scanned by Scanner Go

PAGE NO.
DATE : / /



Conditions must be met by the producer and the consumer processes —

⇓

(i) Producer not necessary to produce an item if the shared buffer is full.

(ii) Consumer process must not consume an item if the shared buffer is empty.

(iii) Shared buffer must be exclusive; this means → at any given instance

↓

Only one process should be able to access the shared buffer and makes changes.

Scanned by Scanner Go

PAGE NO.
DATE : / /

Solution of Producer - Consumer Problem

⇓

Solve using **semaphores**

types

Binary

Similar to **mutex** lock but not the same.

↓

have two **value - 0 and 1**

↓

val **initialized = 1**

↓

used to **implement** the **solution** of **critical** section problem with **multiple processes**.

Counting

value **not** fixed.

↓

Used to **control** access to a resource that has **multiple** instances.

Solution for producer —

when producer produces an item $\xrightarrow{\text{then}}$ value of empty, reduced by 1.

Scanned by Scanner Go

PAGE NO.
DATE : / /

because one slot will be filled now.

⇒ value of mutex is also reduced to Prevent consumer to access the buffer.

⇒ Now, the producer has placed the item and thus the value of "full" is increased by 1.

⇓

the value of mutex also increased by 1 because the task of producer has been completed and consumer can access the buffer.

Solution for consumer —

Consumer is removing an item from buffer, → value of "full" → reduced by 1.

⇒ And the value of mutex is also reduced so that producer cannot access the buffer at this moment.

Scanned by Scanner Go

PAGE NO.
DATE : / /

Now,

the item consumed by consumer

↓

increase value of "empty" by 1.

↓

value of mutex also increased so

producer can access the buffer now.

Scanned by Scanner Go

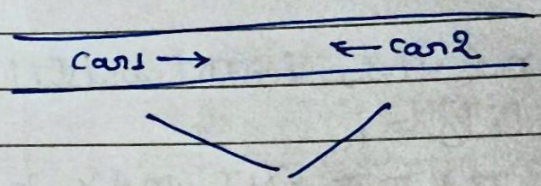
PAGE NO.
DATE : / /

Deadlock (OPERATING SYSTEM)

what is ??

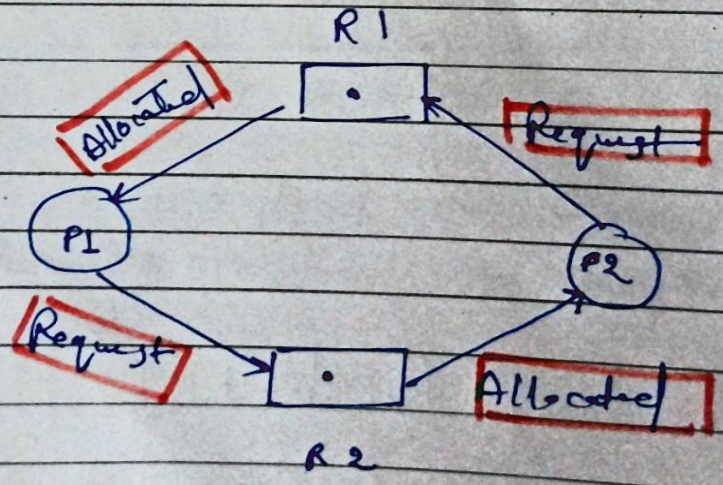
If **two** and **more** processes are **waiting** on **happening** of **some** event, but that **event** does **not** happen, that is **deadlock**

Example :



both are stuck
no one can pass and
waiting for some event
so that is deadlock situation.

example :



Scanned by Scanner Go

PAGE NO.
DATE : / /

Condition of deadlock

- **mutual exclusion** → means → the resource you are using in the scenario that must be used in mutual exclusive way. That means more than one processes cannot used the same ~~pro~~ resource.
- **No preemption** → means like P1 is using R1, R1 is allocated to P1 and P1 need R2 and P2 need R1 so preemption not there. → Preemption also not occur.
- **Hold & wait** → P1 is holding resource that is R1 and waiting for R2 And P2 is holding one resource (R2) and waiting for R1 that is hold and wait.
- **Circular wait** → if their is loop then condition of deadlock arise.

Deadlock handling methods

- (i) Deadlock ignorance (Ostrich method)
- (ii) Deadlock prevention
- (iii) Deadlock avoidance (Banker's algo)
- (iv) Deadlock detection and recovery

Scanned by Scanner Go

PAGE NO.
DATE: / /

Paging (OPERATING SYSTEM)

In **paging** we **split** a process into **equally** sized pages and insert it into the **frames** of **main memory**.

→ Process size = 4B
Page size = 2B
No. of Pages = $\frac{4B}{2B}$ = 2

Note: **Frame size** and **page size** should be **same**.

m/m size = 16B
frame size = 2B
No. of frame = $\frac{16B}{2B}$ = 8 frames

What is **frame**??

→ **Physical Address** space is conceptually divided into number of fixed

Scanned by Scanner Go

PAGE NO.
DATE : / /

size blocks called frames.

What is pages??
↳ logical address space is splitted into fixed-size blocks called pages.

Translation of logical Address into physical address —

Some points to note:

- CPU always generates a logical address.
- In order to access the main memory always a physical address is needed.

↳ logical address generated by CPU always consists of two parts:

1. Page number (p) ⟶ used to specify the specific page of the process from which the CPU wants to read the data and also used as an index to to page table.
2. Page offset (d) ↓ used to specify word on the page that the CPU wants to read.

Scanned by Scanner Go

PAGE NO.
DATE : / /

Process Synchronization (Operating System)

What is ??

If multiple processes runs in our system then there could be two modes, one is serial mode and the other is parallel mode.

Serial mode

Processes will get executed one by one.

means when one process get started, other process can't start until it can't get terminated

like an ATM machine.

Parallel mode

uses multiple environment, means now a days our system use time sharing environment that there will be more than one process at a time

when this much process runs at a time then we get a very big problem

in this case some processes are cooperative and other could be independent

PAGE NO.
DATE : / /

Process

co-operative

Processes are those where **execution** of one **affects** other

there **shares** something **common** between them

use **common variable** so they can share some **memory** // **buffer**

independent

Processes means **execution** of one is **not** putting any **effect** on **other**

nothing common like if I am doing a transaction on online server other I can perform on other server also.

Where **co-operative** process create an **issue** ??

⇒ We have two process **P1** and **P2**

they are **running** on **same** system. In **P1** the **code** is like **x**, there's variable a

Scanned by Scanner Go

PAGE NO.
DATE : / /

P1	P2
1. x = shared;	y = shared;
2. x++;	y--;
3. sleep(1);	sleep(1);
4. shared = x;	shared = y;

Shared variable
int shared = 5

↓

here both process run at same time.

sleep(1) → pause process for 1 sec

here, talking about [uni processor] like our [laptop], so here [P1 and P2] run [parallel] but [one] get [CPU] first.

⇒ Let say [P1] get [CPU first], then it will execute [instruction] no. 1

P1	P2
1. int x = 5	
2. x++ = 6	
3. process pause for 1 sec.	

↓

So here CPU can't stop

In this case [P2] will come in [running queue] → means [context switching]

Scanned by Scanner Go

PAGE NO.
DATE : / /

We saved [P1's] value in [PCB] and we loaded [P2]

↓

now [P2] start instruction — ①

① . y = 5
② . y = ~~5~~ 4 (y--)
③ . sleep(1)

↓

here also process [P2] will [pause]

↓

but [CPU] can't [pause], if it [can't] pause so find [next] process and goes [with P1].

↓

from [P1] we [executed] from [4th] instruction, here [x = 6]

↓

shared into load

↓

now shared = 6

↓

here process will terminated.

now CPU again goes to P2 and start from 4th instruction. here y = 4 and shared = 4

Scanned by Scanner Go

PAGE NO.
DATE: / /

final value of shared = 4

↓

But it is wrong because

for P1 $x = 5$ ↓ $x++$ ↓ $\boxed{x = 6}$

for P2 $y = 5$ ↓ $y--$ ↓ $\boxed{y = 4}$

but here answer is $\boxed{4}$ so this is the issues in $\boxed{\text{co-operative}}$ process.

↓

this problem called $\boxed{\text{Race condition.}}$

??

It is an $\boxed{\text{undesirable}}$ $\boxed{\text{situation}}$ that occurs when a $\boxed{\text{device}}$ or system $\boxed{\text{attempts}}$ to $\boxed{\text{perform}}$ $\boxed{\text{two}}$ or $\boxed{\text{more operations}}$ at the $\boxed{\text{same}}$ time.

Scanned by Scanner Go