MIT World Peace University

Computer Networks
Second Year B.Tech Semister 3
Academic Year 2022-23

Operating Systems

Notes from Tananbaum and Classes

Prepared By

P34. Krishnaraj Thadesar

Batch A2

September 13, 2022

Contents

1 Concurrency Control - Waht to learn

P rocess Synchronization: Principles of Concurrency, Requirements for Mutual Exclusion: Hardware Support, OS Support

C lassical Synchronization Problems: Readers Writers Problem, Producer and Consumer Problem

P rinciples of Deadlock, Deadlock modelling, prevention, avoiance and stuff.

When 2 processes are running at the same time, when they are interdependent on each other through inputs and outputs, then you would call that concurrency. We arent doing multiprocessing by choice, here we just gotta do it at the same time somehow.

2 Design Issues in concurrency

1. Communication among processes
2. Sharing and Competition for resources
3. Synchronization of activities of multiple processes
4. Allocation of processor time to processes.

3 Contexts of Concurrency

1. Multiple Aplications like Multiprogramming(diff programs on a single processor) and Multiprocessing(on diff processors)
2. Strucctured Applications: Some applictions can be effectively programmed as a set of concurrent proesses. (Principles of modular design and structured programming)
3. OS Structure: OS often implemented as a set of processes or threads.

4 Key terms related to concurrency

1. Atomic Operation: A sequence of one or more statements that appear to be inivisible that is no other process can see an intermediate state or interrupt the operations
2. Critical Section: A section of code within a process that requires access to shared resources and that must not be executed while another process is in corresponding section of code.
3. Deadlock: A situation in which two or more processes are unable to proceed because each is waiting for one of the others to do something.
4. Mutual Exclusion: The requirement that when one process is in critical section that access shared resources, no other processes may be in critical section that accesses any of those shared resources.
5. Race Condition: A situation in which multiple threads/Processes read and write a shared data item and final result depends on relative timing of their execution.
6. Starvation: A situation in which a runnable processes is overlooked indefinitely by the scheduler, although it is able to proceed, it is never chosen.

4.1 Race Condition

A race Condition occurs when multiple competing processes or threads read and write data items so that final result depends on the order of eecution of instructions in multiple processes. so 2 processes p1 and p2, say they share global variable 'a'. P1 updates a to 1, and p updates it to 2. Thus two tasks are in a race to write variable 'a'. Loser of race is the one that determines the value of 'a'.

5 Difficulties due to concurrency

1. Sharing of global resources: Eg. Two processes both make use of global variable and both perfrom read and write on that variable, in which read and write are done is critical.
2. Management of resources optimally. Eg. Process has gained the ownership of IO devices but is suspended before using it, thus locking IO device and preventing its use by other processes.
3. Error locating in Program: Pesults are not deterministic are reproducible.

5.1 Example

```
void echo()
{
    chin = getchar();
    chout = chin;
    putchar(chout);
}
```

1. Uniprocessor multiprogramming, single user environment
2. Many applications can call this procedure repeatedly to accept user ip and display on screen.
3. User can jump from one applciation to other.
4. Each application needs or uses procedure echo.
5. Echo is made shared procedure and loaded into a portion of memory global to all applications
6. Single copy of echo procedure is used, saving space.
7. When echo procedure is invoked by some process, an dif fthe process gets suspended for any reason beffore completing it, then no other process can invoke echo till prcess that was suspended gets resumed, and completes echo. Thus other processes are not allowed to access it.
8. It would make more sense if you visualize this in terms of a multi user operating system. So if many people are using a server kinda thing at the same time.

5.2 Example

Say you have global variables that hvae values b = 1, c = 3 If P1 executes b = b + c
P2 does c = c + b

Now if P1 executes first, you would get b = 3, c = 5;
If P2 was to Execute first, then b = 4, c = 3

This is a problem.

5.3 Process Interaction

- Processes that are unaware of each Other (Competition)
    1. Here Multiprogramming of Multiple indepent Processes
    2. OS Needs to know about competition for resources such as printer, disk, file etc.
    3. Potential Problems: Mutual Exclusion, Deadlock, starvation
- Processes that are indirectly aware of each Other (Cooperation by sharing)
    1. Shared Access to some object such as shared variable.
    2. Cooperation by sharing.
    3. Potential Probelm: Mutual Exclusion, Deadlock, starvation, data coherence etc.
- Processes that are directly aware of each other (Cooperation by communication)
    1. Cooperation by communicatio, communication primitives are available.
    2. Potential control problems: Deadlock and starvation.
    3. Mutual exclusion not a probelm, as both are aware of each other.
    4. DeadLock is possible, and so is starvation.

5.4 Three Control problems

- The need for mutual Exclusion: Two or more processes require access to single non sharable resources such as printer, say. Such a resource is called a critical resource, and the portion of code using it is called as critical section of the program. Mutual Exclusion is when you allow only process to work when its in its critical section.
- A common problem would be deadlock, where say P1 is using the printer, and P2 is using a file, Now P1 has to wait for P2 for using the file, and P2 has to wait for P1 to use the printer.
- Starvation would be like how we schedule the processes, and so periodic access would be given. While one process is using some resource, others have to patiently wait, and so they are starving.

Remember to Examples a whole lot while answering questions of OS

5.4.1 Cooperation among processes by sharing

- Processes that interact with other processes without begin explicitly aware of them, access to shared variables, or files or databases is what is being talked about here.
- Processes may use and update shared data without reference to other processes but know that other processes may have access to same data.

- Processes must cooperate to ensure that the data that they share are properly managed.
- control problems of mutual exclusion, deadlock and starvation are again present.
- Data Items are accessed in 2 Modes: Reading and Writing.
- Writing operations must be mutually exclusive.

5.4.2 Requirements for Mutual Exclusion

- Any facility or capability providing support for mutual exclusion should meet following requirements:
- Mutual Exclusion must be **enforced**. Only one process at a time in the CS, among all processes that have CS for same resource or shared object.
- A process that **halts in its non CS** must do without interfering with other processes.
- A Process requiring access to CS must not be **denied or delayed** indefinitely. So no deadlock or starvation.
- When no process is in its CS, any process that requests an entry to its CS, ust be permitted to enter without delay.
- No Asusmptions are made about the relative speed of the process.

5.4.3 Approaches to satisfy the requirements of Mutual Exclusion

1. Hardware Approaches
2. Support from the OS or programming language itself.
3. Software Approach (No support from OS or programming language) You have to write it yourself.