DATE

Assignment # 5

Aagya Sharma
HCE074BCT001

1 How can deadlock be prevented. Explain

A set of processes is said to be deadlocked if each process in the set is waiting for an event that only another process in the set can cause.

Deadlock can be prevented by using the techniques to attack the four conditions that may cause deadlock

a. Prevention from Mutual condition

If no resource were assigned exclusively to a single process, no deadlock may occur. To any mutual exclusion spooling can be done. The resource should not be assigned to a single process until absolutely necessary and convert all non sharable resources to sharable. For example: instead of assigning a printer to a process, we can create a spool, so that several processes can generate output at the same time. Although, spooling can be an effective approach to violate mutual exclusion but it suffers from two kinds of problems

1. This cannot be applied to every resource
2. After some point of time, there may arise a race condition between the processes to get space in that spool

b. Prevention from wait and hold condition

Hold and wait condition lies when a process holds a resource and waiting for some other resource to complete its task. Deadlock occurs because there can be more than

DATE

one process which are holding one resource and waiting for the other in cyclic order. However, we have a mechanism by which a process doesn't hold any resource or doesn't wait. This means, all processes must request all resources before execution. If everything is available the process would be allowed with all required resources and run to completion. Otherwise, nothing would be allocated and the process must wait

The problem with the approach is

1. Practically not possible
2. Possibility of getting starved will be increases due to the fact that some process may hold a resource for a very long time

c. Preventing the non-preemption condition

Deadlock arises due to the fact that a process can't be stopped once it starts. However, if we take the resource away from the process which is causing deadlock then we can prevent deadlock. This is not a good approach at all since if we take a resource away which is being used by the process then all the work which it has done till now can become inconsistent

Consider a printer is being used by any process. If we take the printer away from that process and assign it to some other process then all the data which has been printed can become inconsistent and ineffective and also the fact that the process can't start printing again from where it has left which causes performance inefficiency.

PAGE

DATE

d. Preventing the circular wait

To violate circular wait, we can assign a priority number to each of the resource - A process can't request for a lesses priority resource. This ensures that not a single process can request a resource which is being utilized by some other process and no cycle will be formed. Among all the method, violating circular wait is the only approach that can be implemented practically

2. Write short notes on.

1. Integrated Deadlock Strategy

The Integrated deadlock strategy for deadlock ~~is to~~ involves grouping resources into number of different resource classes. The ordering is imposed on the resource classes and linear ordering Strategy as defined for the prevention of circular wait is used to prevent deadlocks between resource classes. Within a resource class, an ~~ree~~ algorithm that is most appropriate for that class is used

As an example of this technique, consider the following classes of resources

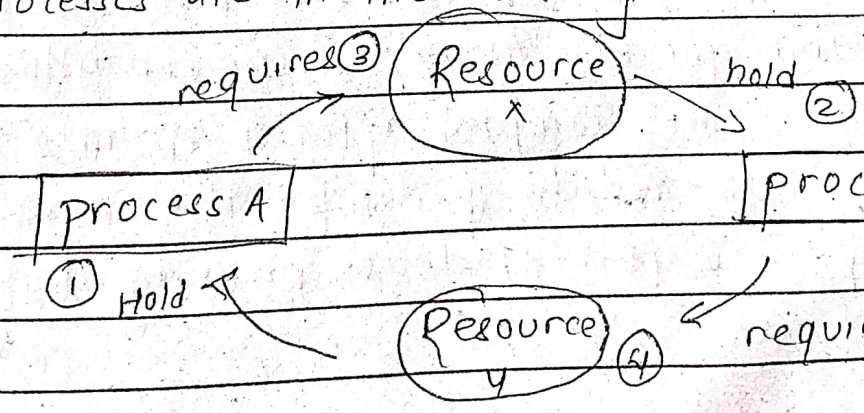
- swappable space
- process Resources
- Main memory
- Internal Resources

DATE

class	description	Strategy
swappable space	blocks of memory on secondary storage for use in swapping processes	Allocate all required resources at one time
process resources	Assignable devices, such as tape drivers, and files	Deadlock avoidance since it is expected that processes declare in advance their needs
main memory	Assignable in pages or segments to processes	Prevention by preemption
Internal resources	such as I/O channels	Prevention by means of resource ordering

2 Livelock

Livelock is a condition that takes place when two or more processes change their state continously, with neither process making progress. In other words, it occurs when two or more processes continually repeat the same interaction in response to changes in the other processes without doing any useful work. These processes are not in the waiting state, and they are running concurrently. This is different from a deadlock because in a deadlock all processes are in the waiting state



PAGE

DATE

In the above example, each of the two given processes needs two resources and they use the primitive polling enter register to try to acquires the locks necessary for them. If the attempts fail, the method works again

1. Process A hold Y resource
2. Process B hold resource X
3. Process A require X resource
4. Process B requires Y resource

Assuming, process A runs first and acquires resource X and the process B runs and acquires resource Y, no matter runs first, none of them further progress.

However, neither of the two processes are blocked. They use up CPU resources repeatedly without any progress being made but also stop any processing block.

Therefore, this situation is not that of a deadlock because there is not a single process that is blocked, but we face the situation, something equivalent to deadlock, which is LIVELOCK.

3. Starvation and aging

Starvation is the name given to the indefinite postponement of a process because it requires some resource before it can run, but the resource, though available for allocation, is never allocated to this process. For example, after a deadlock is detected, it is decided to preempt the resources held by a process (P_i) of least priority. Later, again a deadlock occors with the same process (P_i), its

resources are again preempted. That is P1 faces starvation

Starvation or indefinite blocking is phenomenon associated with the priority scheduling algorithms, in which a process ready to run for CPU can wait indefinitely because of low priority. Starvation can occur in CPU scheduling algorithms like SJF, FCFS, preemptive priority, etc

Aging is a technique to avoid starvation in a scheduling system. It is a technique of gradually increasing the priority of processes that wait in the system for a long time. For example, if priority range from 127 (low) to 0 (high), we could increase the priority of a waiting process by 1 every 15 minutes. Eventually even a process with an initial priority of 127 would take no more than 32 hours for priority 127 process to age to a priority - 0 process.

4. Two phase locking

Two phase locking (2PL) is a concurrency control method that guarantees serializability. Two phase locking is a process used to gain ownership of shared resources without creating the possibility for deadlock. The modificatio of data, and the subsequent release of the locks that protected the data generally are grouped together and called the second phase. Two phase locking prevents deadlock from occurring in systems by releasing all the resources / lock it has acquired, if it is not possible to obtain all the resources required without waiting for another process to finish using a lock. This means that no process is ever in a state where it

is holding some shared resources, and waiting for another process to release a shared resource which it requires.

The resource (or lock) acquisition phase of a "two" shared data access protocol is usually implemented as a loop within which all the locks required to access the shared data are acquired one by one. If any lock is not acquired on the first attempt the algorithm gives up all the locks it had previously been able to get, and starts to try to get all the locks again. This is often call "back-off and re-try" strategy.

5. Communication Deadlock

Process ~~com~~ may communicate with each other by passing messages to each other. Communication deadlock occors if each process in the set is waiting to communicate with another process in the same set and no process ever initiates any communication until it receives communication for which it is waiting.

Communication deadlock is when process A is trying to send a message to process B, then blocks until B sends a reply. If we are using synchronous communication scheme, both A and B are blocked as A is block till B sends reply, and B is blocked till message is received from A. Also, B may be trying to send a message to process C, which is trying to send a message to A using synchronous communication scheme. Here also communication deadlock occurs.