Operating System Concepts

Sessional Assignment Spring 2020

Marks : 20

NAME	**=**	**FAYYAZ MUHAMMAD**
ID	**=**	**14294**
DEPARTEMENT	**=**	**BS(CS) 5TH SEMESTER**
SUBJECT	**=**	**OPERATING SYSTEM**

1. Explain the necessary conditions that may lead to a deadlock situation.

ANS:- A deadlock situation on a resource can arise if and only if all of the following conditions hold simultaneously in a system:

1. **Mutual exclusion:** At least one resource must be held in a non-shareable mode. Otherwise, the processes would not be prevented from using the resource when necessary. Only one process can use the resource at any given instant of time.
2. *Hold and wait* or *resource holding:* a process is currently holding at least one resource and requesting additional resources which are being held by other processes.
3. *No* **preemption:** a resource can be released only voluntarily by the process holding it.
4. *Circular wait:* each process must be waiting for a resource which is being held by another process, which in turn is waiting for the first process to release the resource. In general, there is a set of waiting processes, $P = \{P_1, P_2, \ldots, P_N\}$, such that P_1 is waiting for a resource held by P_2, P_2 is waiting for a resource held by P_3 and so on until P_N is waiting for a resource held by P_1

2. What are the various methods for handling deadlocks?

ANS:- **Methods for handling deadlock**

There are mainly four methods for handling deadlock.

1. Deadlock ignorance

It is the most popular method and it acts as if no deadlock and the user will restart. As handling deadlock is expensive to be called of a lot of codes need to be altered which will decrease the performance so for less critical jobs deadlock are ignored. Ostrich algorithm is used in deadlock Ignorance. Used in windows, Linux etc.

2. Deadlock prevention

It means that we design such a system where there is no chance of having a deadlock.

- **Mutual exclusion:**
  It can't be resolved as it is the hardware property. For example, the printer cannot be simultaneously shared by several processes. This is very difficult because some resources are not sharable.
- **Hold and wait:**
  Hold and wait can be resolved using the conservative approach where a process can start it and only if it has acquired all the resources.
- **Active approach:**
  Here the process acquires only requires resources but whenever a new resource requires it must first release all the resources.
- **Wait time out:**
  Here there is a maximum time bound until which a process can wait for other resources after which it must release the resources.
- **Circular wait:**
  In order to remove circular wait, we assign a number to every resource and the process can request only in the increasing order otherwise the process must release all the high number acquires resources and then make a fresh request.

- **No pre-emption:**
  In no pre-emption, we allow forceful pre-emption where a resource can be forcefully pre-empted. The pre-empted resource is added to the list of resources where the process is waiting. The new process can be restarted only when it regains its old resources. Priority must be given to a process which is in waiting for state.

3. Deadlock avoidance

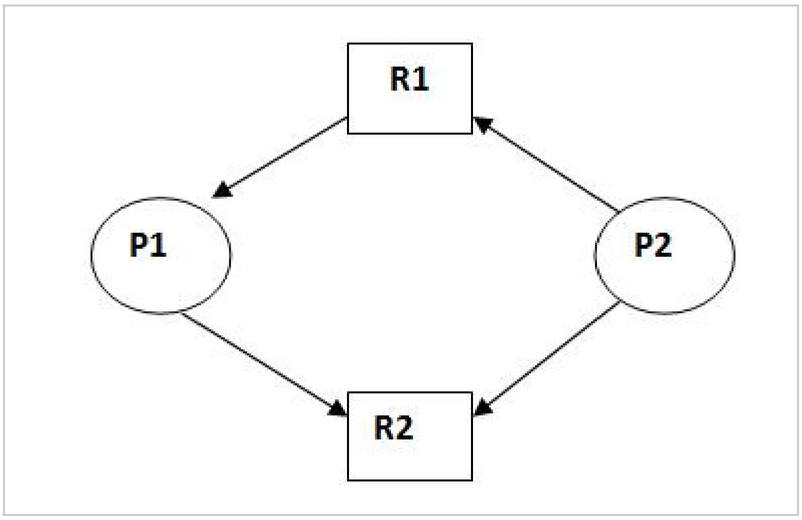
Here whenever a process enters into the system it must declare maximum demand. To the deadlock problem before the deadlock occurs. This approach employs an algorithm to access the possibility that deadlock would occur and not act accordingly. If the necessary condition of deadlock is in place it is still possible to avoid feedback by allocating resources carefully.

Methods for deadlock avoidance

:- Resource allocation graph

This graph is also kind of graphical bankers' algorithm where a process is denoted by a circle Pi and resources is denoted by a rectangle RJ (.dots) inside the resources represents copies.

Presence of a cycle in the resources allocation graph is necessary but not sufficient condition for detection of deadlock. If the type of every resource has exactly one copy than the presence of cycle is necessary as well as sufficient condition for detection of deadlock.



This is in unsafe state (cycle exist) if P1 request P2 and P2 request R1 then deadlock will occur.

4. Detection and recovery

When the system is in deadlock then one method is to inform the operates and then operator deal with deadlock manually and the second method is system will automatically recover from deadlock. There are two ways to recover from deadlock:

- **Process termination:**
  Deadlock can be eliminated by aborting a process. Abort all deadlock process. Abort is processed at a time until the deadlock cycle is eliminated. This can help to recover the system from file deadlock.

- **Resources preemption:**
  To eliminate deadlock using resources preemption, we prompt the same resources pas processes and give these resources to another process until the deadlock cycle is broken.
  Here a process is partially rollback until the last checkpoint or and hen detection algorithm is executed.

3. **Is it possible to have a deadlock involving only one single process? Explain your answer.**

ANS:- Deadlock with one process is not possible. Here is the explanation.

A deadlock situation can arise if the following four conditions hold simultaneously in a system.

- Mutual Exclusion.
- Hold and Wait.
- No Preemption.
- Circular-wait.

It is not possible to have circular wait with only one process, thus failing a necessary condition for Circular wait. There is no second process to form a circle with the first one. So it is not possible to have a deadlock involving only one process.

4. **Consider a system consisting of 4 resources of the same type that are shared by 3 processes, each of which needs at most 2 resources. Show that the system is deadlock free.**

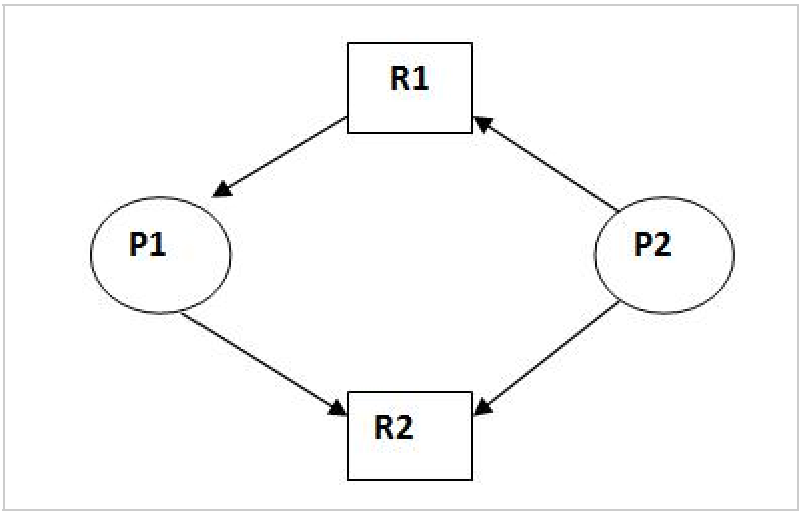
ANS:- Suppose the system is deadlocked. This implies that each process is holding one resource and is waiting for one more. Since there are three processes and four resources, on process must be able to obtain two resources. This process requires no more resources and therefore it will return its resources when done.

5. **What is a resource allocation graph? How do you obtain a wait-for graph from it? Explain their uses**.

ANS:- Resource allocation graph

This graph is also kind of graphical bankers' algorithm where a process is denoted by a circle Pi and resources is denoted by a rectangle RJ (.dots) inside the resources represents copies.

Presence of a cycle in the resources allocation graph is necessary but not sufficient condition for detection of deadlock. If the type of every resource has exactly one copy than the presence of cycle is necessary as well as sufficient condition for detection of deadlock.



This is in unsafe state (cycle exist) if P1 request P2 and P2 request R1 then deadlock will occur.

2) Bankers's algorithm

The resource allocation graph algorithms not applicable to the system with multiple instances of the type of each resource. So for this system Banker's algorithm is used.

Here whenever a process enters into the system it must declare maximum demand possible.

At runtime, we maintain some data structure like current allocation, current need, current available etc. Whenever a process requests some resources we first check whether the system is in a safe state or not meaning if every process requires maximum resources then is there ant sequence in which request can be entertaining if yes then request is allocated otherwise rejected.

Safety algorithm

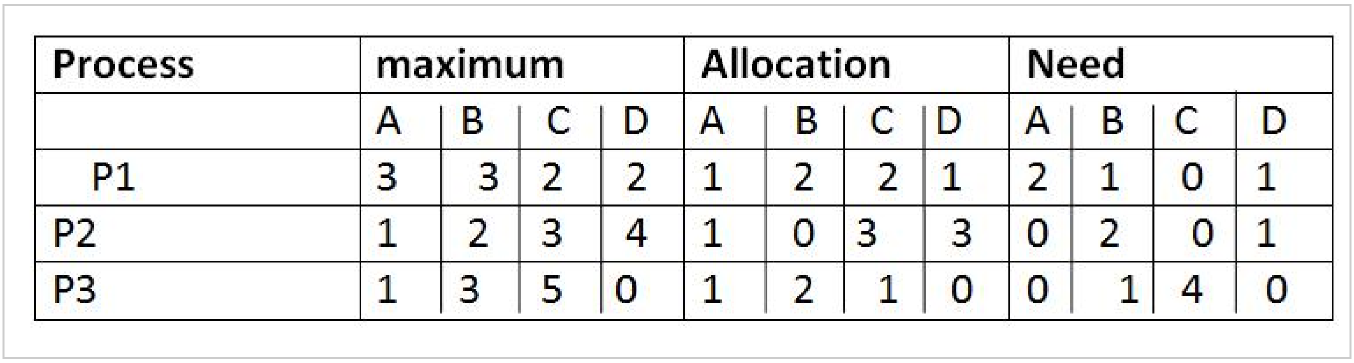
This algorithm is used to find whether system is in safe state or not we can find

```
Remaining Need = Max Need – Current allocation
Current available = Total available – Current allocation
```

Ad by Valueimpression

Let's understand it by an example:

Consider the following 3 process total resources are given for **A= 6, B= 5, C= 7, D = 6**



Process	maximum				Allocation				Need			
	A	B	C	D	A	B	C	D	A	B	C	D
P1	3	3	2	2	1	2	2	1	2	1	0	1
P2	1	2	3	4	1	0	3	3	0	2	0	1
P3	1	3	5	0	1	2	1	0	0	1	4	0

First we find the need matrix by **Need= maximum – allocation**

Then find available **resources = total – allocated**

```
    A   B   C  D( 6   5  7  6) -   A  B  C  D(  3  4  6  4
    Available resources A B C D(  3 1  1   2)
```

Then we check whether the system is in deadlock or not and find the safe sequence of process.

P1 can be satisfied

```
    Available= P1 allocated + available
    ( 1, 2, 2, 1) +( 3, 1, 1,2) = (4, 3, 3, 3)
```

P2 can be satisfied

```
    Available= P2 allocated + available
    (1, 0, 3, 3) + (4, 3, 3, 3) = (5, 3, 6, 6)
```

P3 can be satisfied

```
    Available= P3 allocated + available
    (1, 2, 1, 0) + (5, 3, 6, 6) = (6, 5, 7, 6)
```

So the system is safe and the safe sequence is **P1 → P2 → P3**

6. **Can a system detect that some of its processes are starving? If you answer “yes,” explain how it can. If you answer “no,” explain how the system can deal with the starvation problem.**

ANS:- Detection of starvation requires future knowledge since no amount of record-keeping statistics on processes can determine if it is making

'progress' or not. However, starvation can be prevented by 'aging' a
process. This means maintaining a rollback count for each process, and
including this as part of the cost factor in the selection
process for a victim for preemption/rollback.

7. **On a disk with 1000 cylinders, number 0 to 999, compute the number of tracks the disk arm must move to satisfy all the requests in the disk queue. Assume the last request serviced was at track 345 and the head is moving toward track 0. The queue in FIFO order contains requests for the following tracks: 123, 847, 692, 475, 105, 376. Perform the computations for the following disk scheduling algorithms:**

   - **FCFS**
   - **SSTF**

ANS:-

FCFS

The FCFS schedule is 345,123,874,692,475,105 and 376.

Total head movement = (345-123) + (874-123) + (874-692) + (692-475) + (475-105) + (376-105) =2013

SSTF

The SSTF schedule is 345,376,475,692,874,123 and 105.

Total head movement = (376-345) + (475-376) + (692-475) + (847-692) + (874-123) +(123-105) = 1298