Deadlock in operating system tutorial pdf

The process in operating systems uses a variety of resources and uses resources as follows. 1) Resource Requests 2) Using Resource 2) Releases Resource Stupid is a situation where a set of processes are blocked because each process holds a resource and waits for another resource purchased by some other process. Consider an example where two trains go towards each other on the same track and there is only one track, none of the trains can move once they are in front of each other. A similar situation is observed in operating systems, where there are two or more processes that hold some resources and wait for resources held by others (s). For example, in the chart below, process 1 holds resource 1 and waits for Resource 2, which is acquired by Process 2, and Process 2 waits for resource 1. A lock may occur if the following four conditions hold at the same time (Necessary conditions) Mutual exception: One or more resources are not common (only one process can be used at a time) Hold and wait: the process holds at least one resource and waits for resources. There is no pre-emption: the resource cannot be taken from the process if the process does not release the resource. Circular waiting: a set of processes await each other in a circular form. The methods of handling gridlock there are three ways to handle the impasse 1) Preventing or avoiding gridlock: The idea is to not allow the system to be deadlocked. One can increase in each category individually, Prevention is done by denying one of the aforementioned necessary conditions for gridlock. Escape is a kind of futuristic character. Using the Avoidance strategy, we have to make an assumption. We must ensure that all information about the resources that the process will require is known to us before the process is complete. We use the Banker's algorithm (which in turn is a gift from Dijkstra) in order to avoid a dead end. 2) Detection and recovery of the impasse: Let the deadlock occur and then make a pre-emption to deal with it as soon as it happened. 3) Ignore the problem as a whole: If the impasse is very rare, then let it happen and restart the system. This is an approach that both Windows and UNIX are taking. Exercise: 1) Suppose n processes, P1, ... Pn shares m identical units of resources that can be reserved and released one at a time. The maximum resource requirement of the Pi process is Si, where Si is zgt; 0. Which of the following is sufficient to ensure that the impasse does not occur? (GATE CS 2005) (A) A (B) B (C) C (D) D For a solution, see Question 4 of See DEADLOCK for more questions. Links: box/os/ch07.pdf please write comments if you find something wrong or you want to share Information about the topic discussed above Is Attention Reader! Don't stop learning now. Get an all-important CS Theory concept for an SDE interview with the CS Theory course at a student-friendly price and become a ready industry. Recommended messages: Improved : BJP_supporting_Leftist, rithesharjun deadlocked situation that occurs in the OS when any process enters the waiting state because another waiting process holds the required resource. Congestion is a common problem in multiprocessing, where multiple processes have a certain type of mutually exclusive resources, known as soft locking or software. In this operating system tutorial, you'll learn: An example of a dead end in the real world example would be traffic that goes in just one direction. Here the bridge is considered a resource. So when there is a dead end, it can be easily resolved if one car is backed up (pre-emptive resources and rollback). Several cars may have to back up if a dead end occurs. So hunger is possible. Example of a dead endWhat is a circular expectation? One process awaits the resource, which is carried out by the second process, which is also waiting for the resource held by the third process, etc. This will continue until the last process waits for the resource held by the first process. This creates a circular chain. For example, Process A is allocated a B resource when you request Resource A. In the same way, Process B is allocated a resource A, and it requests resource B. This creates a circular waiting cycle. Example Circular Waiting For example, a computer has three USB drives and three processes. Each of the three processes is able to hold one of the USB drives. Thus, when each process requests a different drive, three processes will have a dead end, as each process will wait for the release of the USB drive that is currently in use. This will lead to a circular circuit. A circular example of waiting forDeadlock DetectionA deadlock can be detected by a resource planner. A resource planner helps the OS track all resources allocated to different processes. Thus, when a dead end is discovered, it can be resolved by using the following methods: Preventing deadlock: It is important to prevent gridlock before it can occur. The system checks each transaction before it is complete to make sure it does not lead to dead-end situations. So that even a small change occurs dead, that operation that could lead to gridlock in the future it also never allowed the process to perform. This is a set of methods to ensure that at least one of the conditions may not exist. No pre-emptive action: No pre-emption - Resource can only be released voluntarily by the process, withholding it after the process has completed its task which keeps some resources to request another resource that can't be immediately allocated to it, is that all resources will be released. Pre-emptive resources require a list of resources for the process that awaits. The process will be resumed only if it can restore its old resource and the new one it requests. If the process requests any other resource when it is available, it has been passed on to the request process. If it's done by another process that's waiting for another resource, we release it and give it to the request process. Mutual exception: Mutual exclusion is a complete form of Mutex. This is a special type of binary semaphore that is used to manage access to a shared resource. It includes a priority inheritance mechanism to avoid problems with inversion with an expanded priority. This allows you to keep current tasks with a higher priority in a locked state as soon as possible. Shared resources, such as read only files, never lead to dead ends, but resources such as printers and tape drives need exclusive access by one process. Hold and Wait: In this state, processes must be stopped from withholding one or more resources while waiting for one or more others. Circular expectation: it imposes a full order of all types of resources. Circular waiting also requires that each process request resources in order to increase the number of listings. Avoiding the blockage it is better to avoid a deadlock rather than take action after breaking the deadlock. It needed more information, such as how resources should be used. Avoiding a lock is the simplest and most useful model, each proclaiming the maximum amount of resources of each type that you may need. Algorithms to avoid deadlock-avoidance help to dynamically assess the state of resource allocation, so that there will never be a circular waiting situation. One instance of resource type. You need to use a resource allocation graph that is sufficient for rich resource-type instances. Cycles are necessary, but never sufficient for gridlock. Using the banker's algorithm The Difference Between Hunger and DeadlockHere, some important differences between gridlock and hunger: A deadlock situation occurs when one of the processes has been blocked. Hunger is a situation where all low-level processes are blocked and high-priority processes are carried out. Congestion is an endless process. Hunger is a long wait, but not an endless process. There is always hunger in every impasse. Every starvation is not necessarily at an impasse. The congestion occurs, then a mutual exception, keep and wait. Here, pre-emption and circular anticipation do not occur at the same time. This is due to uncontrolled priority and resource management. The benefits of gridlock here are the pros/benefits of using the Stupid method This situation is good for processes that perform one burst of activity there is no pre-emption required for The method when applied to resources whose condition can be saved and restored easily is feasible to ensure by compiling the verification time does not need to calculate the time of execution, as the problem solved in the design of the SystemDisadvantages of the DeadlockHere method, are minus /lack of use of the dead-end method delaying process initiation Processes should know the future need for Resources Pre-empts more often than necessaryDis-allows additional requests for resourcesIn fact. Description: Determining a dead end: This is a situation that occurs in the OS when any process enters into a waiting state, because another waiting process holds the required resourceCircular waiting time when one process waits for a resource that is held by a second process, which is also waiting for a resource held by a third process, etc. It is important to prevent deadlock before it can occur. The resource can only be released voluntarily by the process, withholding it after the process has finished its task. Mutual exclusion is a complete form of Mutex. This is a special type of binary semaphore that is used to manage access to a shared resource. Holding and waiting is a condition in which processes must be stopped from withholding one or more resources while waiting for one or more others. Avoiding a lock is the simplest and most useful model, each proclaiming the maximum amount of resources of each type that you may need. The deadlock prevention algorithm helps you dynamically assess the state of resource allocation so that there's never a circular waiting situation. Congestion is an endless process, while hunger is a long wait, not an endless process. Page 2First Come First Serve (FCFS) is an operating system planning algorithm that automatically performs queries and processes in the queue when they arrive. This is the simplest and easiest CPU scheduling algorithm. In this type of algorithm, the processes the processor asks for first get the processor distribution first. This is managed with a FIFO queue. The full FORM of FCFS is the first to come First Serve. When the process is in the ready turn, its PCB (Process Control Unit) is connected to the tail of the queue, and when the processor becomes free, it should be assigned to the process at the beginning of the queue. In this operating system tutorial, you'll learn: The characteristics of the FCFS method It supports a non-proactive and proactive planning algorithm. Jobs are always performed in order the first came. It's easy to implement and use. This method is poor in performance, and the overall waiting time is quite high. An example of FCFS an example of the FCFS method is to buy a movie ticket at the ticket office. In this planning algorithm, the person is serviced according to the queue image. The person who arrives first in line first buys a ticket and Next. This will continue until the last person in the queue buys a ticket. Using this algorithm, the processor process works in a similar way. How does FCFS work? Calculating the average waiting time here is an example of five processes arriving at different times. Each process has a different burst time. The arrival time of the P1 process 6 2 P2 3 5 P3 8 1 P4 3 0 P5 4 4 Using the FCFS planning algorithm, these processes are handled as follows. Step 0) The process begins with P4, which has an arrival time of 0 Step 1) During No. 1, P3 arrives. The P4 is still running. Thus, the P3 is in the queue. The Process Explosion Arrival Time P1 6 2 P2 3 5 P3 8 1 P4 3 0 P5 4 4 Step 2) During 2, P1 arrives, which is kept in line. Blast Time Arrival Time P1 6 2 P2 3 5 P3 8 1 P4 3 0 P5 4 4 Step 3) During No. 3, P4 process completes its execution. Step 4) During No.4, P3, which is first in line, begins running. Blast Time Arrival Time P1 6 2 P2 3 5 P3 8 1 P4 3 0 P5 4 4 Step 5) During No. 5, P2 arrives and it is kept in line. Blast Time Arrival Time P1 6 2 P2 3 5 P3 8 1 P4 3 0 P5 4 4 Step 6) During 11, P3 completes its execution. Step 7) During No.11, P1 starts running. It has a blast time of 6. It completes the run at the time interval of 17 Step 8) During 17, P5 begins running. It has a splash time of 4. It completes the run during Step 21 9) During 21, P2 begins running. It has a splash time of 2. It completes with a time interval of 23 Step 10) Let's calculate the average waiting time, for example. Waiting time - Start time - Arrival time P4 - 0-0 - 0 P3 - 3-1 - 2 PIs - 11-2 - 9 P5 17-4 - 13 P2 21-5 16 Average waiting time pluses /benefits of using fcFS planning algorithm: The simplest form of processor planning algorithm Easy to program First came the misuse of FCFsHere, are the cons/flaws of using the FCFS planning algorithm: It's not a proactive CPU planning algorithm, so once the process has been allocated to the processor, it will never release the processor until it finishes running. The average waiting time is high. Short processes that are at the back of the queue must wait for a long process at the front to finish. Not an ideal technique for time-sharing systems. Because of its simplicity, FCFS is not very effective. Description: Definition: FCFS is an operating system planning algorithm that automatically performs regular queries and processes in order of arrival, which supports a non-proactive and proactive planning algorithm. FCFS stands for First Come First ServeA a real example of the FCFS method is to buy a movie ticket at the ticket office. This is the simplest form of the planning algorithm not a proactive CPU planning algorithm, so once the process has been allocated to the processor, it will never release the processor until it is complete. Pages it's a storage mechanism that allows the OS to extract processes from the secondary store into the main memory in the form of pages. In the Paging method, the basic memory is divided into small fixed-size physical memory blocks called frames. The frame size should be the same as the size of the page to maximize the use of the underlying memory and avoid external fragmentation. Paging is used for faster access to data, and this is a logical concept. In this Paging tutorial you'll learn: Example, if the primary memory size is 16 KB and the frame size is 1 KB. Here the main memory will be divided into a collection of 16 frames of 1 KB each. There are 4 separate processes in the system which are A1, A2, A3, and A4 4 KB each. Here, all processes are divided into pages of 1 KB each, so that the operating system can store one page in one frame. At the beginning of the process, all frames remain blank, so that all pages of processes will be stored in an adjacent way. In this example, you can see that A2 and A4 move to a waiting state after a while. Thus, eight frames become empty, and so other pages can be loaded into these empty blocks. The 8 page (8KB) A5 process is waiting in the ready queue. In this example, you can see that there are eight non-contugugus frames in memory, and paging provides flexibility in storing the process in different locations. This allows you to download pages of the A5 process instead of A2 and A4. What is Paging protection? The paging process should be protected by the concept of inserting an additional bit called Valid/Invalid bit. Paging memory protection is achieved by linking bits of protection with each page. These bits are linked to each page table entry and show protection on the corresponding page. The benefits of PagingHere are the benefits of using the Paging method: An easy-to-use memory control algorithm is not necessary for external fragmentationSwapping easily between equal in size pages and frames pages. The drawbacks of PagingHere, are the flaws/minuses of Paging:May lead to internal fragmentation Complex memory control algorithmPage tables consume addittonal memory. Multi-level paging can result in overhead memory. What is segmentation? The segmentation method works in much the same way as paging, the only difference between them is that the segments have a variable length, while in the paging method, the pages are always fixed-sized. The program segment includes the basic function of the program, data structure, utility functions, etc. OS supports the segment map table for all processes. It also includes a list of free memory blocks along with its size, number of segments, and it places memory in the main memory or virtual memory. The benefits of the segmentation method are already there Protection of segmentation offers inside SegmentsY can achieve sharing by segments of links to multiple processes. Does not suggest internal fragmentationSegment tables use less memory than pagingDisadvantages SegmentationHere are cons/lack segmentation In the segmentation method processes are downloaded/removed from the underlying memory. Thus, the free memory space is divided into small pieces, which can create a problem of external fragmentation Of Costley's memory control algorithmSummary: Paging is a storage mechanism that allows the OS to extract processes from the secondary repository into the main memory in the form of pages. The paging process should be protected by the concept of inserting an additional bit called Valid/Invalid bit. The biggest advantage of paging is that it is easy to use a memory management algorithm Paging can lead to internal fragmentation of the Segmentation method working almost the same as paging, the only difference between them is that the segments have a variable length while, in the paging method, the pages are always fixed size. You can achieve segment sharing by reference to multiple processes. Segmentation is an expensive memory management algorithm, thePage 4A Livelock is a situation where the request for an exclusive lock is denied repeatedly, as many overlapping common locks continue to interfere with each other. Processes continue to change their status, which further prevents them from fulfilling the task. This further prevents them from doing the job. In this operating system tutorial, you'll learn: Examples of LivelockExample 1: The simplest example of Livelock will be two people who meet face-to-face in a hallway, and they both step aside to let others pass. They end up moving from side to side without making any progress as they move in the same way at the time. Here they never intersect. Example 2: You can see in the image above, each of the two processes needs two resources, and they use a primitive registry to enter the survey to try to get the locks they need. If the attempt fails, the method works again. The process of retention Y resource process B keeps the XProcess A resource require X Resource Process B to require A resource assuming the process works first and acquires the X data resource, and then Process B works and acquires resource Y, no matter what process works in the first place, none of them further progress. However, neither of these two processes is blocked. They use the CPU resources repeatedly without any progress, but also stop any processing unit. So this situation is not a dead end, because there is no process that is blocked, but we are faced with a situation that is equivalent to a dead end, LIVELOCK. What leads to Livelock? Livelock occurs when the total number of processes allowed in a particular system should be determined by the total number of entries in the process table. Therefore, slots for the process table should be as the ultimate resource. What is a dead end? Congestion is a situation that occurs in the OS, when any process enters the standby, because another waiting process holds the required resource. Congestion is a common problem in multiprocessing, where multiple processes have a certain type of mutually exclusive resources, known as soft locking or software. An example of a real deadlockA example is traffic that only goes

in one direction. Here the bridge is considered a resource. So when there is a dead end, it can be easily resolved if one car is backed up (pre-emptive resources and rollback). Several cars may have to back up if a dead end occurs. So hunger is possible. Example of a dead endWhat is hunger? Hunger is a situation where all low-level processes are blocked and high-priority processes continue. In any system, high/low priority resource requests continue to be dynamic. Therefore, some policies need to decide who gets support when. Using some algorithms, some processes may not get the service they want, even if they are not at a dead end. Hunger occurs when some streams make common resources inaccessible for a long period of time. Example of starvation: For example, an object offers a synchronized method that can take a long time to return. If one thread frequently uses this method, other threads that also need frequent synchronized access to the same object will often be blocked. The difference between gridlock, hunger and LivelockA deadlock is a situation that occurs in the OS when any process enters a waiting state, because the required resource is in a different waiting process. On the other hand, the livelock is almost like a dead end, except that the state processes that are involved in the livelock always continue to change to each other, no one progresses. Thus, Livelock is a unique case of resource starvation. Description: Definition: Livelock is a situation where an exclusive lock request is repeatedly denied, as many overlapping common locks continue to interfere with each other. Livelock occurs when the total number of processes allowed in a particular system must be determined by the total number of entries in the process tableA the impasse that occurs in the OS when any process enters the waiting state because another waiting process holds the required resource. A real example would be traffic that goes only in one direction. An example of Livelock is two people who meet face-to-face in a hallway, and they both step aside to allow the other to pass. Hunger is a situation where all low-level blocked and high-priority processes continue. Continue. Continue. deadlock in operating system tutorialspoint. deadlock in operating system tutorial pdf. deadlock characterization in operating system tutorial. deadlock avoidance in operating system tutorial