|  |  |
| --- | --- |
| **1** | **Suppose the time to service a page fault is on the average 10 milliseconds, while a memory access takes 1 microsecond. Then a 99.99% hit ratio results in average memory access time of** |
|  | Average memory access time =  [(% of page hit)\*(memory access time)+(% of page miss)\*(time to service a page fault)]/100  So, average memory access time in microseconds is. (99.99\*1 + 0.01\*10\*1000)/100 = (99.99+100)/1000 = 199.99/1000 =1.9999 µs |
| 2 | **Consider a machine with 64 MB physical memory and a 32-bit virtual address space. If the page size is 4KB, what is the approximate size of the page table?** |
|  | Answer:- 2MB  Number of entries in page table =  (virtual address space size)/(page size)  Using above formula we can say that there will be 2^(32-12) = 2^20 entries in page table. No. of bits required to address the 64MB Physical memory = 26. So there will be 2^(26-12) = 2^14 page frames in the physical memory. And page table needs to store the address of all these 2^14 page frames. Therefore, each page table entry will contain 14 bits address of the page frame and 1 bit for valid-invalid bit. Since memory is byte addressable. So we take that each page table entry is 16 bits i.e. 2 bytes long.  Size of page table =  (total number of page table entries) \*(size of a page table entry)  = (2^20 \*2) = 2MB  For the clarity of the concept, please see the following figure. As per our question, here p = 20, d = 12 and f = 14. |
| 3 | **Consider Peterson’s algorithm for mutual exclusion between two concurrent processes i and j. The program executed by process is shown below.**  **repeat**  **flag [i] = true;**  **turn = j;**  **while ( P ) do no-op;**  **Enter critical section, perform actions, then exit critical**  **section**  **flag [ i ] = false;**  **Perform other non-critical section actions.**  **until false;**  **For the program to guarantee mutual exclusion, the predicate P in the while loop should be** |
|  | flag [j] = true and turn = j |
| 4 | **In a system with 32 bit virtual addresses and 1 KB page size, use of one-level page tables for virtual to physical address translation is not practical because of** |
|  | the large memory overhead in maintaining page tables  Since page size is too small it will make size of page tables huge.  Size of page table = (total number of page table entries) \*(size of a page table entry)  Let us see how many entries are there in page table Number of entries in page table = (virtual address space size)/(page size)  = (2^32)/(2^10) = 2^22  Now, let us see how big each entry is.  If size of physical memory is 512 MB then number of bits required to address a byte in 512 MB is 29. So, there will be (512MB)/(1KB) = (2^29)/(2^10) page frames in physical memory. To address a page frame 19 bits are required. Therefore, each entry in page table is required to have 19 bits.  Note that page table entry also holds auxiliary information about the page such as a present bit, a dirty or modified bit, address space or process ID information, amongst others. So size of page table  > (total number of page table entries) \*(size of a page table entry)  > (2^22 \*19) bytes > 9.5 MB  And this much memory is required for each process because each process maintains its own page table. Also, size of page table will be more for physical memory more than 512MB. Therefore, it is advised to use multilevel page table for such scenarios. |
| 5 | **Let the page fault service time be 10ms in a computer with average memory access time being 20ns. If one page fault is generated for every 10^6 memory accesses, what is the effective access time for the memory?** |
|  | **Let P be the page fault rate**  **Effective Memory Access Time = p \* (page fault service time) +**  **(1 - p) \* (Memory access time)**  **= ( 1/(10^6) )\* 10 \* (10^6) ns +**  **(1 - 1/(10^6)) \* 20 ns**  **= 30 ns (approx)** |
| 6 | **Two processes, P1 and P2, need to access a critical section of code. Consider the following synchronization construct used by the processes: Here, wants1 and wants2 are shared variables, which are initialized to false. Which one of the following statements is TRUE about the above construct?**  /\* P1 \*/  while (true) {  wants1 = true;  while (wants2 == true);  /\* Critical  Section \*/  wants1=false;  }  /\* Remainder section \*/  /\* P2 \*/  while (true) {  wants2 = true;  while (wants1==true);  /\* Critical  Section \*/  wants2 = false;  }  /\* Remainder section \*/ |
|  | It does not prevent deadlocks, but ensures mutual exclusion. |
| 7 | **Consider the following snapshot of a system running n processes. Process i is holding Xi instances of a resource R, 1 <= i <= n. currently, all instances of R are occupied. Further, for all i, process i has placed a request for an additional Yi instances while holding the Xi instances it already has. There are exactly two processes p and q such that Yp = Yq = 0. Which one of the following can serve as a necessary condition to guarantee that the system is not approaching a deadlock?** |
|  | (A) min (Xp, Xq) < max (Yk) where k != p and k != q  (B) Xp + Xq >= min (Yk) where k != p and k != q (C) max (Xp, Xq) > 1 (D) min (Xp, Xq) > 1  Answer (B) Since both p and q don’t need additional resources, they both can finish and release Xp + Xq resources without asking for any additional resource. If the resources released by p and q are sufficient for another process waiting for Yk resources, then system is not approaching deadlock. |
|  | **An operating system uses the Banker’s algorithm for deadlock avoidance when managing the allocation of three resource types X, Y, and Z to three processes P0, P1, and P2. The table given below presents the current system state.**  **Here, the Allocation matrix shows the current number of resources of each type allocated to each process and the Max matrix shows the maximum number of resources of each type required by each process during its execution.**  **There are 3 units of type X, 2 units of type Y and 2 units of type Z still available. The system is currently in a safe state. Consider the following independent requests for additional resources in the current state:**  https://www.geeksforgeeks.org/wp-content/uploads/gq/2014/04/GATECS2014Q42.png  **REQ1: P0 requests 0 units of X,**  **0 units of Y and 2 units of Z**  **REQ2: P1 requests 2 units of X,**  **0 units of Y and 0 units of Z** |
|  | Only REQ2 can be permitted  This is the current safe state.   |  |  |  | | --- | --- | --- | |  | AVAILABLE | X=3, Y=2, Z=2 | |  | MAX | ALLOCATION | |  | X Y Z | X Y Z | | P0 | 8 4 3 | 0 0 1 | | P1 | 6 2 0 | 3 2 0 | | P2 | 3 3 3 | 2 1 1 |     Now, if the request REQ1 is permitted, the state would become :   |  |  |  |  | | --- | --- | --- | --- | |  | AVAILABLE | X=3, Y=2, Z=0 |  | |  | MAX | ALLOCATION | NEED | |  | X Y Z | X Y Z | X Y Z | | P0 | 8 4 3 | 0 0 3 | 8 4 0 | | P1 | 6 2 0 | 3 2 0 | 3 0 0 | | P2 | 3 3 3 | 2 1 1 | 1 2 2 |     Now, with the current availability, we can service the need of P1. The state would become :   |  |  |  |  | | --- | --- | --- | --- | |  | AVAILABLE | X=6, Y=4, Z=0 |  | |  |  |  |  | |  | MAX | ALLOCATION | NEED | |  | X Y Z | X Y Z | X Y Z | | P0 | 8 4 3 | 0 0 3 | 8 4 0 | | P1 | 6 2 0 | 3 2 0 | 0 0 0 | | P2 | 3 3 3 | 2 1 1 | 1 2 2 |     With the resulting availability, it would not be possible to service the need of either P0 or P2, owing to lack of Z resource. Therefore, the system would be in a deadlock. ⇒ We cannot permit REQ1.   Now, at the given safe state, if we accept REQ2 :   |  |  |  |  | | --- | --- | --- | --- | |  | AVAILABLE | X=1, Y=2, Z=2 |  | |  |  |  |  | |  | MAX | ALLOCATION | NEED | |  | X Y Z | X Y Z | X Y Z | | P0 | 8 4 3 | 0 0 1 | 8 4 2 | | P1 | 6 2 0 | 5 2 0 | 1 0 0 | | P2 | 3 3 3 | 2 1 1 | 1 2 2 |     With this availability, we service P1 (P2 can also be serviced). So, the state is :   |  |  |  |  | | --- | --- | --- | --- | |  | AVAILABLE | X=7, Y=4, Z=2 |  | |  | MAX | ALLOCATION | NEED | |  | X Y Z | X Y Z | X Y Z | | P0 | 8 4 3 | 0 0 1 | 8 4 2 | | P1 | 6 2 0 | 5 2 0 | 0 0 0 | | P2 | 3 3 3 | 2 1 1 | 1 2 2 |     With the current availability, we service P2. The state becomes :   |  |  |  |  | | --- | --- | --- | --- | |  | AVAILABLE | X=10, Y=7, Z=5 |  | |  | MAX | ALLOCATION | NEED | |  | X Y Z | X Y Z | X Y Z | | P0 | 8 4 3 | 0 0 1 | 8 4 2 | | P1 | 6 2 0 | 5 2 0 | 0 0 0 | | P2 | 3 3 3 | 2 1 1 | 0 0 0 |     Finally, we service P0. The state now becomes :   |  |  |  |  | | --- | --- | --- | --- | |  | AVAILABLE | X=18, Y=11, Z=8 |  | |  | MAX | ALLOCATION | NEED | |  | X Y Z | X Y Z | X Y Z | | P0 | 8 4 3 | 0 0 1 | 0 0 0 | | P1 | 6 2 0 | 5 2 0 | 0 0 0 | | P2 | 3 3 3 | 2 1 1 | 0 0 0 |   The state so obtained is a safe state. ⇒ REQ2 can be permitted. So, only REQ2 can be permitted. Hence, B is the correct choice.   Please comment below if you find anything wrong in the above post. |
|  | **Considering a system with five processes P0 through P4 and three resources types A, B, C. Resource type A has 10 instances, B has 5 instances and type C has 7 instances. Suppose at time t0 following snapshot of the system has been taken:**  [safety](https://media.geeksforgeeks.org/wp-content/cdn-uploads/gq/2016/01/safety.png)  **Question1. What will be the content of the Need matrix?**  Need [i, j] = Max [i, j] – Allocation [i, j]  So, the content of Need Matrix is:  [unnamed](https://media.geeksforgeeks.org/wp-content/cdn-uploads/gq/2016/01/unnamed.png)  **Question2.  Is the system in safe state? If Yes, then what is the safe sequence?**  Applying the Safety algorithm on the given system,  [questionsolved](https://media.geeksforgeeks.org/wp-content/cdn-uploads/gq/2016/01/questionsolved.png)  **Question3. What will happen if process P1requests one additional instance of resource type A and two instances of resource type C?**  [allocation](https://media.geeksforgeeks.org/wp-content/cdn-uploads/gq/2016/01/Allocation.png)  We must determine whether this new system state is safe. To do so, we again execute Safety algorithm on the above data structures.  [Q31](https://media.geeksforgeeks.org/wp-content/cdn-uploads/gq/2016/01/Q31.png)  Hence the new system state is safe, so we can immediately grant the request for process **P1 .** |
|  | **Pr Max Allocation Available**  **A, B, C, D A, B, C, D A, B, C, D**  **P0 6 0 1 2 4 0 0 1 3 2 1 1**  **P1 2 7 5 0 1 1 0 0**  **P2 2 3 5 6 1 2 5 4**  **P3 1 6 5 3 0 6 3 3**  **P4 1 6 5 6 0 2 1 2**  **Using Banker’s algorithm, answer the following questions:-**  **i) How many resources of type A, B, C, D are there?**  **ii) What are the contents of need matrix?**  **iii) Find if the system is in safe state? If it is, find the safe sequence.** |
|  | 1)  A = 4+1+1+3 = 9  B = 1+2+6+2+2 = 13  C = 5+3+1+1 = 10  D = 1+4+3+2+1 = 11  **2) Need matrix** :  enter image description here  3) YES , system is in safe state  Safe Sequence: P0, P2, P3, P4, P1 |
|  |  |