**Tutorial 4: Memory Management**

Q1. Differentiate between the followings:

1. Fixed partition and dynamic partition

|  |  |
| --- | --- |
| **Fixed partition** | **Dynamic partition** |
| * a memory allocation scheme with portions assigned to each job. The size of each partition remains static once the system is in operation. * Each partition can only be reconfigured when the computer system is shut down, reconfigured and restarted. * If the partition sizes are too small, larger jobs will be rejected. If partition sizes are too big, memory can be wasted leading to internal fragmentation | * A memory allocation scheme creating their own partitions in main memory in which jobs are given as much memory as they request when they are loaded for processing * external fragmentation occurs as new jobs enter the system, fragments of free memory are created between blocks of allocated memory |

1. Internal fragmentation and external fragmentation

|  |  |
| --- | --- |
| **Internal fragmentation** | **External fragmentation** |
| A situation in which a partition is only partially used by a job; the remaining space within the partition is unavailable to any other job | a situation in which the dynamic allocation of memory creates unusable fragments of free memory between blocks of busy, or allocated, memory. |

1. Best Fit algorithm and First Fit algorithm.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | ***First-Fit*** | ***Best-Fit*** | | Faster to implement but may not be using memory space efficiently. | Uses memory efficiently but slower to implement because the entire free list table needs to be searched before allocation can be made. | | Algorithm is less complex. | Algorithm is more complex because it needs to find smallest block of memory into which the job can fit. | | Memory list organized according to memory locations, low-order | Memory list organized according to memory size, smallest to largest. | |

Q2. Consider a system is using fixed *memory partition* techniques and 4 processes waiting in a queue as show in the figure 1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Free Space List** | | | **Partition Number** | **Size (KB)** | | A | 350 | | B | 550 | | C | 750 | | D | 500 | | |  |  | | --- | --- | | **Queuing Processes** | | | **Process** | **Size (KB)** | | 1 | 225 | | 2 | 500 | | 3 | 540 | | 4 | 360 | |  |  | |

**Figure 1**

Show how the four processes, namely P1, P2, P3 and P4, are allocated memory partitions by using the following memory allocation algorithms.

1. First-fit
2. Best-fit
3. Worst-fit

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **First fit**   |  |  |  | | --- | --- | --- | | Partition | Process | Fragmentation | | A | P1 | 125 | | B | P2 | 50 | | C | P3 | 210 | | D | P4 | 140 | | Total | | 525 |  * No job need to wait. Total fragmentation is 525kb. |
| **Best fit**   |  |  |  | | --- | --- | --- | | Partition | Process | Fragmentation | | A | P1 | 125 | | B | P3 | 10 | | C | P4 | 390 | | D | P2 | 0 | | Total | | 525 |  * No job need to wait. Total fragmentation is 525kb. |
| **Worst fit**   |  |  |  | | --- | --- | --- | | Partition | Process | Fragmentation | | A |  |  | | B | P2 | 50 | | C | P1 | 525 | | D | P4 | 140 | | Total | | 715 |   Process 3 will need to wait as there is no available block large enough (540kb) to store. Total fragmentation is 715kb. |

Q3. A system is using variable-size partitions (*dynamic memory partition* techniques are used); partitions can be allocated on the basis of first-fit, best-fit and worst-fit. In a contiguous memory allocation system, the free space list contains 6 entries in the following order: 190KB, 550KB, 220KB, 420KB, 650KB, 110KB.

Given the following requests in the input queue: A=210KB, B=430KB, C=100KB, D=430KB, determine how these requests can be satisfied based on each of the allocation schemes listed below.

* First-fit
* Best-fit
* Worst-fit

For the scenario mentioned above, which allocation scheme uses the memory more efficiently? Explain why.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **First-fit** | **Best-fit** | **Worst-fit** | | A in block 550  B in block 650  C in block 190  D cannot be allocated | A in block 220  B in block 550  C in block 110  D in block 650 | A in block 650  B in block 550  C in block 420 (remaining space in hole 650 after allocated 210 to A)  D cannot be allocated |   Best-fit uses the memory most efficiently, as all the requests can be satisfied with the least amount of unused memory due to external fragmentation |

Q4. Page replacement is fundamental in demand paging. Some popular page replacement algorithms are *First-In-First-Out (FIFO)*, *Optimal and Least-Recently-Used (LRU)*.

1. Illustrate how each of the above page replacement algorithm works, by using the page reference string of a process as given below. You may assume that 3 frames are allocated to the process and they are initially empty. For each algorithm, determine also the number of page fault.

2, 3, 2, 1, 5, 2, 4, 5, 3, 2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **FIFO**:  Present   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 2 | 2 | 2 | 2 | 5 | 5 | 5 | 5 | 3 | 3 | |  | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | |  |  |  | 1 | 1 | 1 | 4 | 4 | 4 | 4 | |  |  | H |  |  |  |  | H |  | H |   No. of Page fault = 7  **Optimal:**  Future   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 2 | 2 | 2 | 2 | 2 | 2 | 4 | 4 | 4 | 4 | |  | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | |  |  |  | 1 | 5 | 5 | 5 | 5 | 5 | 5 | |  |  | H |  |  | H |  | H | H |  |   No. of Page fault = 6  **LRU: Past**   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | |  | 3 | 3 | 3 | 5 | 5 | 5 | 5 | 5 | 5 | |  |  |  | 1 | 1 | 1 | 4 | 4 | 4 | 2 | |  |  | H |  |  | H |  | H |  |  |   No. of Page fault = 7 |

1. Highlight **ONE** advantage and **ONE** disadvantage of the optimal page replacement algorithm.

|  |
| --- |
| * Disadvantages – Complex, less easy to implement. Assistance needed is low * Advantages – OPR is perfect, but not possible in practice as the operating system cannot know future requests. Error handling is tough. |

1. Explain the concept of thrashing in systems which support virtual memory via demand paging.

|  |
| --- |
| * Cause by under allocation of the minimum number of pages required by a process, forcing it to continuously page fault. * That is the process is busy swapping pages in and out between main memory and secondary storage. * As a result, the system spends an excessive amount of time on paging, compared to the execution of processes. |

Q5. Support a page size of 256 bytes is used in demand paging system. Given the following sequence of addresses:

321, 150, 700, 510, 1031, 400, 350, 150, 842, 910

1. Translate the given virtual addresses inti a page reference string.

|  |
| --- |
| 1, 0, 2, 1, 4, 1,1,0,3,3 |

1. Prepare a page trace analysis and count the number of page faults by First-In-First-Out (FIFO), Optimal and Least-Recently-Used (LRU) page replacement algorithms, assuming 3 page frames to be allocated. Then, compute the hit ratio for each algorithm.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **FIFO**   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **1** | **1** | **1** | **1** | **4** | **4** | **4** | **4** | **3** | **3** | |  | **0** | **0** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | |  |  | **2** | **2** | **2** | **2** | **2** | **0** | **0** | **0** | |  |  |  | **H** |  |  | **H** |  |  | **H** |   7 page faults. Hit ratio = 3/10 = 0.3  **LRU**   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | |  | **0** | **0** | **0** | **4** | **4** | **4** | **4** | **3** | **3** | |  |  | **2** | **2** | **2** | **2** | **2** | **0** | **0** | **0** | |  |  |  | **H** |  | **H** | **H** |  |  | **H** |   6 page faults. Hit ratio = 4/10 = 0.4  Optimal   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **3** | **3** | |  | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | |  |  | **2** | **2** | **4** | **4** | **4** | **4** | **4** | **4** | |  |  |  | **H** |  | **H** | **H** | **H** |  | **H** |   5 page faults. Hit ratio = 5/10 = 0.5 |

Q6. CPU generates a logical address that is mapped to physical memory location. This is implemented by the operating system which maintains page and segment tables for the mapping.

1. Why are page sizes always powers of 2?

|  |
| --- |
| The bytes within a page are addressed using the last N bits of a virtual address, for some value of N. Since the number of addresses that can be expressed with N bits is 2N, the page size is a power of 2. |

1. Consider a logical address space of 32 pages of 1024 words each, mapped onto a physical memory of 64 frames. How many bits are there in the logical address and physical address respectively?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Addressing within a 1024 word page requires 10 bits because 1024 = 210. Since the logical address of 32 = 25 pages, the logical address must be 10 + 5 = 15 bits.  Logical address  15   |  |  | | --- | --- | | Page number | Page offset |   5 10  Similarly, since there are 64 = 26 frames, physical addresses are 6 + 10 = 16.  Physical address  16   |  |  | | --- | --- | | Frame number | Frame offset |   6 10 |

Q7. (i) Memory management using paging is more common than segmentation. Highlight **TWO** advantages that paging systems have over segmentation systems

|  |
| --- |
| Advantages of paging over segmentation   * Allocation of memory space for processes is easy as the size of the page and frame are the same * Eliminate external fragmentation |

(ii) In a particular segmentation system, a process P has a segment table shown below:

|  |  |  |
| --- | --- | --- |
| **Segment** | **Base** | **Length (Bytes)** |
| 0 | 1250 | 700 |
| 1 | 300 | 90 |
| 2 | 2400 | 550 |

Figure 2: Segment Table

Explain how the system establishes the corresponding physical address for a logical address of <1, 35> through the use of the segment table.

What will happen during address translation for a logical address of <2,600>?

|  |
| --- |
| Logical address *<1,35>* where *1* represents the segment number and *35* the displacement/offset in the segment. During address translation, the segment number is used as index to the segment table to retrieve the <*base, length*>. The offset *35* is checked against the *length* to ascertain its validity (must be less than *length*). As *35 < 90*, the offset is then added to the *base* ie *300 + 35* to obtain the physical address ie 335  As for the logical address *<2, 600>*, since the offset is larger the segment length (*550*) thus a trap occurs due to addressing error. |

**Self-Review**

Q1. Given the following information in **Table 1**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Free Space List | |  | Queuing Processes | |
| Partition | Size (KB) |  | Process | Size (KB) |
| A | 650 |  | 1 | 600 |
| B | 600 |  | 2 | 650 |
| C | 250 |  | 3 | 250 |
| D | 300 |  | 4 | 300 |
| E | 650 |

**Table 1**

Show how the four processes are allocated into fixed memory partitions when the ***first-fit*** and ***best-fit*** file allocation algorithms are applied.

*Note: Show your answers using the following table format.*

Table format:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Partition | Partition Size | Process | Process Size | Internal Fragmentation |
|  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. **First fit**  |  |  |  |  |  | | --- | --- | --- | --- | --- | | Partition | Partition Size | Process | Process Size | Internal Fragmentation | | A | 650 | 1 | 600 | 50 | | B | 600 | 3 | 250 | 350 | | C | 250 |  |  |  | | D | 300 | 4 | 300 | 0 | | E | 650 | 2 | 650 | 0 |  1. **Best fit**  |  |  |  |  |  | | --- | --- | --- | --- | --- | | Partition | Partition Size | Process | Process Size | Internal Fragmentation | | A | 650 | 2 | 650 | 0 | | B | 600 | 1 | 600 | 0 | | C | 250 | 3 | 250 | 0 | | D | 300 | 4 | 300 | 0 | | E | 650 |  |  |  | |

Q2. Consider the following page reference string:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 4 | 3 | 5 | 6 | 6 | 3 | 5 | 1 | 4 | 3 | 7 | 2 | 4 |

Assume that 5 frames are initially empty in the memory. Perform the page faults trace to determine the number of page faults that will occur for the following page replacement algorithms.

1. First-In-First-Out (FIFO)
2. Least Recently Used (LRU)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **FIFO**     |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1 | 2 | 4 | 3 | 5 | 6 | 6 | 3 | 5 | 1 | 4 | 3 | 7 | 2 | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1 | 1 | 1 | 1 | 1 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | |  | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | |  |  | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 7 | 7 | 7 | |  |  |  | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 3 | |  |  |  |  | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | |  |  |  |  |  |  | H | H | H |  | H | H |  |  |  |  * Page fault =10/15   **LRU**   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1 | 2 | 4 | 3 | 5 | 6 | 6 | 3 | 5 | 1 | 4 | 3 | 7 | 2 | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1 | 1 | 1 | 1 | 1 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 7 | 7 | 7 | |  | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | |  |  | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | |  |  |  | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | |  |  |  |  | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 2 | 2 | |  |  |  |  |  |  | H | H | H |  | H | H |  |  | H |  * Page fault = 9/15 |