

1. What controls whether or not a computer begins to thrash? How can we 'solve' thrashing?

A: Thrashing occurs because the system doesn't know when it has taken on more work than it can handle. Large amount of computing time is spent in paging overhead, response time goes down. Increase amount of memory, decrease degree of multiprogramming. Can also try changing paging algorithm.

2. What is a working set? How would knowing the working set be useful?

A: The set of pages a process is currently working with, the set needed to avoid thrashing. We can avoid scheduling a process if its working set does not fit in main memory, which will avoid thrashing.

3. Page Replacement: Consider a demand paging system with four pages of physical memory that executes the following reference stream.

1; 2; 3; 4; 1; 2; 3; 5; 1; 2; 3; 6

a) Assuming that memory starts empty and the system uses a FIFO page replacement strategy, determine the number and type of page faults that will occur and the final contents of memory.

A: There were nine page faults. The table below shows the pages in memory at each point in the reference stream. The memory contents are shown in FIFO order with pages at the front of the queue at the left. The page to be replaced is, thus, the page shown on the left.

Reference	Fault	Type	Memory			
1	yes	compulsory	1			
2	yes	compulsory	1	2		
3	yes	compulsory	1	2	3	
4	yes	compulsory	1	2	3	4
1	no	-	1	2	3	4
2	no	-	1	2	3	4
3	no	-	1	2	3	4
5	yes	compulsory	2	3	4	5
1	yes	capacity/policy	3	4	5	1
2	yes	capacity/policy	4	5	1	2
3	yes	capacity/policy	5	1	2	3
6	yes	compulsory	1	2	3	6

b) Determine the number and type of page faults that will occur and the final contents of memory, assuming the system uses the LRU page replacement strategy.

A: There were six page faults. The table below shows the pages in memory at each point in the reference stream. The memory contents are shown in LRU order with the most recently used page at the left. The page to be replaced is, thus, the page shown at the right.

Reference	Fault	Type	Memory			
1	yes	compulsory	1			
2	yes	compulsory	2	1		
3	yes	compulsory	3	2	1	
4	yes	compulsory	4	3	2	1
1	no	-	1	4	3	2
2	no	-	2	1	4	3
3	no	-	3	2	1	4
5	yes	compulsory	5	3	2	1
1	no	-	1	5	3	2
2	no	-	2	1	5	3
3	no	-	3	2	1	5
6	yes	compulsory	6	3	2	1

4. What is an Inode/File Descriptor? What do they store about files? Where would be the most efficient place to put them on a disk?

A: Records the files properties and the location of its blocks. How it records the location of its blocks depends on the allocation method used. For best results, want them to be stored close to the file data to minimize seeks. You can also consider placing them close to directory entries for the same reason.

5. Why is multi-level indexed file block allocation efficient for small files?

A: Because the location of the first 12 blocks of a file are recorded directly in the file inode in multi-level indexed allocation.