Unit 3- QB- Selected Answers

1 Explain the difference between internal and external fragmentation.

Answer:

a. Internal fragmentation is the area in a region or a page that is not used by the job occupying that region or page. This space is unavailable for use by the system until that job is finished and the page or region is released.
b. External fragmentation is unused space between allocated regions of memory. Typically external fragmentation results in memory regions that are too small to satisfy a memory request, but if we were to combine all the regions of external fragmentation, we would have enough memory to satisfy a memory request.

2. Given five memory partitions of 100 KB, 500 KB, 200 KB, 300 KB, and 600 KB (in order), how would each of the first-fit, best-fit, and worst-fit algorithms place processes of 212 KB, 417 KB, 112 KB, and 426 KB (in order)? Which algorithm makes the most efficient use of memory?

Answer:

a. First-fit:
b. 212K is put in 500K partition
c. 417K is put in 600K partition
d. 112K is put in 288K partition (new partition 288K = 500K − 212K)
e. 426K must wait
f. Best-fit:
g. 212K is put in 300K partition
h. 417K is put in 500K partition
i. 112K is put in 200K partition
j. 426K is put in 600K partition
k. Worst-fit:
l. 212K is put in 600K partition
m. 417K is put in 500K partition
n. 112K is put in 388K partition
o. 426K must wait

In this example, best-fit turns out to be the best.

4. Compare the main memory organization schemes of contiguous memory allocation, pure segmentation, and pure paging with respect to the following issues:

a. external fragmentation
b. internal fragmentation
c. ability to share code across processes

Answer:

The contiguous memory allocation scheme suffers from external fragmentation as address spaces are allocated contiguously and holes develop as old processes die and new processes are initiated. It also does not allow processes to share code, since a process's virtual memory segment is not broken into noncontiguous fine grained segments. Pure segmentation also suffers from external fragmentation as a segment of a process is laid out contiguously in physical memory and fragmentation would occur as segments of dead processes are replaced by segments of new processes. Segmentation, however, enables processes to share code; for instance, two different processes could share a code segment but have distinct data segments. Pure paging does not suffer from external fragmentation, but instead suffers from internal fragmentation. Processes are allocated in page granularity and if a page is not completely utilized, it results in internal fragmentation and a corresponding wastage of space. Paging also enables processes to share code at the granularity of pages.

6 Assuming a 1 KB page size, what are the page numbers and offsets for the following address references (provided as decimal numbers):

a. 2375
b. 19366
c. 30000
d. 256
e. 16385

Answer:

a. page = 1; offset = 391
b. page = 18; offset = 934
c. page = 29; offset = 304
d. page = 0; offset = 256
e. page = 1; offset = 1

7. Consider a logical address space of 32 pages with 1024 words per page; mapped onto a physical memory of 16 frames.

a. How many bits are required in the logical address?
b. How many bits are required in the physical address?

Answer:

a. 2^5 = 32 + $2^1$0 = 1024 = 15 bits.
b. 2^4 = 32 + $2^1$0 = 1024 = 14 bits.

10. Why are segmentation and paging sometimes combined into one scheme?

Answer:

Segmentation and paging are often combined in order to improve upon each other. Segmented paging is helpful when the page table becomes very large. A large contiguous section of the page table that is unused can be collapsed into a single-segment table entry with a page-table address of zero. Paged segmentation handles the case of having very long segments that require a lot of time for allocation. By paging the segments, we reduce wasted memory due to external fragmentation as well as simplify the allocation.

15.Assume we have a demand-paged memory. The page table is held in registers. It takes 8 milliseconds to service a page fault if an empty page is available or the replaced page is not modified, and 20 milliseconds if the replaced page is modified. Memory access time is 100 nanoseconds.

Assume that the page to be replaced is modified 70 percent of the time. What is the maximum acceptable page-fault rate for an effective access time of no more than 200 nanoseconds?

Answer:

$$0.2\ \mu\text{sec} = (1 - P) \times 0.1\ \mu\text{sec} + (0.3P) \times 8\ \text{millisec} + (0.7P) \times 20\ \text{millisec}$$
$$0.1 = -0.1P + 2400\,P + 14000\,P$$
$$0.1 \simeq 16{,}400\,P$$
$$P \simeq 0.000006$$

16.When a page fault occurs, the process requesting the page must block while waiting for the page to be brought from disk into physical memory. Assume that there exists a process with five user-level threads and that the mapping of user threads to kernel threads is many to one. If one user thread incurs a page fault while accessing its stack, will the other user threads belonging to the same process also be affected by the page fault—that is, will they also have to wait for the faulting page to be brought into memory? Explain.

Answer:

Yes, because there is only one kernel thread for all user threads, that kernel thread blocks while waiting for the page fault to be resolved. Since there are no other kernel threads for available user threads, all other user threads in the process are thus affected by the page fault.

17.Consider the following page reference string:

7, 2, 3, 1, 2, 5, 3, 4, 6, 7, 7, 1, 0, 5, 4, 6, 2, 3, 0 , 1.

Assuming demand paging with three frames, how many page faults would occur for the following replacement algorithms?

- LRU replacement
- FIFO replacement
- Optimal replacement

Answer:

- 18
- 17
- 13

18 Assume that you are monitoring the rate at which the pointer in the clock algorithm (which indicates the candidate page for replacement) moves. What can you say about the system if you notice the following behavior:

a. pointer is moving fast
b. pointer is moving slow

Answer:

If the pointer is moving fast, then the program is accessing a large number of pages simultaneously. It is most likely that during the period between the point at which the bit corresponding to a page is cleared and it is checked again, the page is accessed again and therefore cannot be replaced. This results in more scanning of the pages before a victim page is found. If the pointer is moving slow, then the virtual memory system is finding candidate pages for replacement extremely efficiently, indicating that many of the resident pages are not being accessed.

19. Discuss situations in which the LFU page-replacement algorithm generates fewer page faults than the LRU page-replacement algorithm. Also discuss under what circumstances the opposite holds.

Answer:

Consider the following sequence of memory accesses in a system that can hold four pages in memory: 1 1 2 3 4 5 1. When page 5 is accessed, the least frequently used page-replacement algorithm would replace a page other than 1, and therefore would not incur a page fault when page 1 is accessed again. On the other hand, for the sequence “1 2 3 4 5 2,” the least recently used algorithm performs better.

20.Discuss situations in which the MFU page-replacement algorithm generates fewer page faults than the LRU page-replacement algorithm. Also discuss under what circumstances the opposite holds.

Answer:

Consider the sequence in a system that holds four pages in memory: 1 2 3 4 4 4 5 1. The most frequently used page replacement algorithm evicts page 4while fetching page 5, while the LRU algorithm evicts page 1. This is unlikely to happen much in practice. For the sequence “1 2 3 4 4 4 5 1,” the LRU algorithm makes the right decision.