Toby Chappell

CPSC 380

Homework #3

Part III – Memory Management

1. Explain the difference between internal and external fragmentation. (4 pts)

Internal Fragmentation: allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used

External Fragmentation: total memory space exists to satisfy a request, but it is not contiguous

1. When Given six memory partitions of 300 KB, 600 KB, 350 KB, 200 KB, 750 KB, and 125 KB (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of size 115 KB, 500 KB, 358 KB, 200 KB, and 375 KB (in order)? Rank the algorithms in terms of how efficiently they use memory. (6 pts)

First-fit:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 300 | 600 | 350 | 200 | 750 | 125 |
| 185 | 600 | 350 | 200 | 750 | 125 |
| 185 | 100 | 350 | 200 | 750 | 125 |
| 185 | 100 | 350 | 200 | 392 | 125 |
| 185 | 100 | 150 | 200 | 392 | 125 |
| 185 | 100 | 150 | 200 | 17 | 125 |

Best-fit:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 300 | 600 | 350 | 200 | 750 | 125 |
| 300 | 600 | 350 | 200 | 750 | 10 |
| 300 | 100 | 350 | 200 | 750 | 10 |
| 300 | 100 | 350 | 200 | 392 | 10 |
| 300 | 100 | 350 | 0 | 392 | 10 |
| 300 | 100 | 350 | 0 | 17 | 10 |

Worst-fit:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 300 | 600 | 350 | 200 | 750 | 125 |
| 300 | 600 | 350 | 200 | 635 | 125 |
| 300 | 600 | 350 | 200 | 135 | 125 |
| 300 | 242 | 350 | 200 | 135 | 125 |
| 300 | 100 | 150 | 200 | 135 | 125 |
| E | R | R | O | R | ! |

Unable to place processes using worst-fit.

Best-fit is more efficient than first-fit.

1. On a system with paging, a process cannot access memory that it does not own. Why? How could the operating system allow access to other memory? Why should it or should it not? (6 pts)

The reason that a process cannot access memory that is not its own is because the page will not be in the page table. To allow access, the OS needs to allow entries for non-process memory to be added to the process's page table. This is useful when two or more processes need to exchange data (easy to read/write to same physical address vs varying logical addresses).

1. Assuming a 1-KB page size, what are the page numbers and offsets for the following address references (provided as decimal numbers): (10 pts)

2^n = 1KB = 1024 = 2^10 => # of bits for offset = 10

* 1. 3085

= 110000001101 =>

Page number: 11 = 3

Page offset: 0000001101 = 13

* 1. 42095

= 1010010001101111 =>

Page number: 101001 = 41

Page offset: 0001101111 = 111

* 1. 215201

110100100010100001 =>

Page number: 11010010 = 210

Page offset: 0010100001 = 161

* 1. 650000

10011110101100010000 =>

Page number: 1001111010 = 634

Page offset: 1100010000 = 784

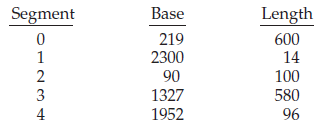
* 1. 2000001

111101000010010000001 =>

Page number: 11110100001 = 1953

Page offset: 0010000001 = 129

1. Consider the following segment table: Segment Base Length (5 pts)



What are the physical addresses for the following logical addresses?

1. 0,430

219 + 430 = 649

1. 1,10

2300 + 10 = 2310

1. 2,500

Illegal reference

1. 3,400

1327 +400 = 1727

1. 4,112

Illegal reference

1. Assume that a program has just referenced an address in virtual memory. Describe a scenario in which each of the following can occur. (If no such scenario can occur, explain why.) (8 pts)

* TLB miss with no page fault

has been brought into memory, but has been removed from the TLB

* TLB miss and page fault

Address doesn’t exist in page table so address is not in memory.

* TLB hit and no page fault

is in memory and in the TLB. Most likely a recent reference  
TLB hit and page fault cannot occur. The TLB is a cache of the page table.

* TLB hit and page fault

Not possible, if address in TLB then address is in memory (therefore cannot have page fault).

1. What is the copy-on-write feature, and under what circumstances is its use beneficial? What hardware support is required to implement this feature? (5 pts)

The copy-on-write feature creates a copy of a set of values that are being accessed by two programs simultaneously. This allows the two programs to access the different copies without interfering with each other in a write-protected manner. The hardware required is some functionality so that, on each memory access, the page table is consulted to check whether the page is write-protected. If it is, a trap occurs and the operating system should resolve the issue.

1. Consider the following page reference string: (6 pts)

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 7 | 7 | 1 |  | 1 | 3 | 3 | 3 | 7 |  | 7 | 7 | 5 | 5 | 5 | 2 | 2 | 2 | 1 |
|  | 2 | 2 | 2 |  | 2 | 2 | 4 | 4 | 4 |  | 1 | 1 | 1 | 4 | 4 | 4 | 3 | 3 | 3 |
|  |  | 3 | 3 |  | 5 | 5 | 5 | 6 | 6 |  | 6 | 0 | 0 | 0 | 6 | 6 | 6 | 0 | 0 |

18 page faults

* FIFO replacement

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 7 | 7 | 1 |  | 1 |  | 1 | 6 | 6 |  | 6 | 0 | 0 | 0 | 6 | 6 | 6 | 0 | 0 |
|  | 2 | 2 | 2 |  | 5 |  | 5 | 5 | 7 |  | 7 | 7 | 5 | 5 | 5 | 2 | 2 | 2 | 1 |
|  |  | 3 | 3 |  | 3 |  | 4 | 4 | 4 |  | 1 | 1 | 1 | 4 | 4 | 4 | 3 | 3 | 3 |

17 page faults

* Optimal replacement

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 7 | 7 | 1 |  | 1 |  | 1 | 1 | 1 |  |  | 1 |  | 1 | 1 | 1 | 1 |  |  |
|  | 2 | 2 | 2 |  | 5 |  | 5 | 5 | 5 |  |  | 5 |  | 4 | 6 | 2 | 3 |  |  |
|  |  | 3 | 3 |  | 3 |  | 4 | 6 | 7 |  |  | 0 |  | 0 | 0 | 0 | 0 |  |  |

1. age faults
2. What is the cause of thrashing? How does the system detect thrashing? Once it detects thrashing, what can the system do to eliminate this problem? (5 pts)

Thrashing is caused by under allocation of the minimum number of pages required by a process, forcing it to continuously page fault. The system can detect thrashing by evaluating the level of CPU utilization compared to the level of multiprogramming. To eliminate thrashing, the system should reduce the level of multiprogramming.

1. Consider a demand-paging system with the following time-measured utilizations:

(10 pts)

CPU utilization 20%

Paging disk 97.7%

Other I/O devices 5%

For each of the following, indicate whether it will (or is likely to) improve

CPU utilization. Explain your answers.

1. Install a faster CPU.

No, since the CPU is underutilized and it spends a lot of time accessing disk, the system is thrashing. Therefore, the CPU is not the issue so a faster CPU would not help.

1. Install a bigger paging disk.

This may or may not help since thrashing is more likely caused by the main memory capacity and disk accessing speed. As such, the page disk size might not be the issue.

1. Increase the degree of multiprogramming.

No, increasing the degree of multiprogramming would worsen thrashing.

1. Decrease the degree of multiprogramming.

Yes, decreasing the degree of multiprogramming would allow for more frames per process, therefore reducing thrashing.

1. Install more main memory.

Yes, installing more main memory would allow for more frames per process, therefore reducing thrashing.