COP 4600 – Final examination

Date: Dec 9, 2013

**Name: ………………………………………………………………………………………………………….**

Instructions:

* This exam is open book and open notes. Allotted time is 180 minutes.
* Note that the points add up to 100 + 20 bonus points.

# Problem 1 (10 pts)

1. Explain what is an access matrix in computer protection. (2 sentences)

ANSWER: An access matrix is an abstract model of protection where the rows represent domains and the columns represent objects. Each entry in the matrix consists of a set of access rights.

1. The simplest approach to implement an access matrix is to just store it in an Excel spreadsheet in a file. Give two reasons why this is a bad idea. (2 sentences)
	* It will be too big.
	* Whenever you add the file, you need to update its right for every user.
	* Whenever you add a user, you need to update its rights for every file
2. Describe how the Unix system is approximating the functionality of an access matrix. (3 sentences)

ANSWER: It uses access lists associated with users, but only considers 3 categories of users for each file (owner, group members and others). This is an approximation, because not all possible combinations can be expressed with it.

# Problem 2 (20 points)

A process contains eight virtual pages on disk and is assigned a fixed allocation of four page frames in the main memory.

## a) Show the successive pages residing in the four frames using the LRU (least recently used) policy.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 4 | 6 | 1 | 3 | 2 | 4 | 6 | 1 | 3 | 2 | 4 | 6 | 1 | 3 | 2 | 4 | 6 | 1 | 3 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 1 | 1 | 1 | 1 | 6 | 6 | 6 | 6 | 4 | 4 | 4 | 4 |
|  | 4 | 4 | 4 | 4 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 1 | 1 | 1 | 1 | 6 | 6 | 6 |
|  |  | 6 | 6 | 6 | 6 | 4 | 4 | 4 | 4 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 1 | 1 |
|  |  |  | 1 | 1 | 1 | 1 | 6 | 6 | 6 | 6 | 4 | 4 | 4 | 4 | 2 | 2 | 2 | 2 | 3 |

* How many page table hits were there? ………0…………..
* How many misses? ……20……………..
* How many misses were unavoidable? ………4…………..
* Explain the result (1 sentence)

Very bad pattern for LRU: you always replace the one which will come next.

## c) Repeat for the optimal algorithm

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 4 | 6 | 1 | 3 | 2x | 4x | 6x | 1 | 3x | 2x | 4x | 6 | 1x | 3x | 2x | 4 | 6x | 1x | 3x |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 4 | 4 | 4 | 4 |
|  | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
|  |  | 6 | 6 | 6 | 6 | 6 | 6 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|  |  |  | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |

* How many page table hits were there? …12………………..
* How many misses? ……8……………..
* How many misses were unavoidable? ……4…………..
* Explain the result (1 sentence)

The optimal algorithm does not fall for the trick in the pattern.

# Problem 3 (20 pts)

Suppose the page table for a process A currently executing on the processor looks like the following. All numbers are decimal, everything is numbered starting from zero, and all addresses are memory byte addresses. The page size is **4096** bytes.

|  |  |  |  |
| --- | --- | --- | --- |
| Virtual page number | Valid bit (1 =valid) | Modify bit(1 = modified) | Page frame number |
| 0 | 1 | 0 | 10 |
| 1 | 1 | 1 | 11 |
| 2 | 0 | 0 | - |
| 3 | 1 | 0 | 3 |
| 4 | 0 | 0 | - |
| 5 | 1 | 1 | 1 |

## What **physical address**, if any, would each of the following **virtual addresses** correspond to:

77

Page 0, Frame 10, 40960 + 77 = 41037

12328

Page 3, disp 40, Frame 3 = 12328

10000

Page 2, no physical address

## What **virtual address**, if any, would each of the following **physical addresses** correspond to:

40970

Frame 10, disp 10, Page 0 = 10

70

Frame 0: not present

# Problem 4 (20 pts)

A type of hard disk has a failure probability of 0.05 / year. We equate “failure” with “data loss”.

1. What is the failure probability of 2 disks assembled in a RAID-0 configuration?
2. What is the failure probability of 2 disks assembled in a RAID-1 configuration?
3. Is the failure probability of a RAID-5 configuration with 4 disks higher or lower than the same 4 disks arranged in two RAID-1 pairings? Explain.
4. 1 – 0.95 \* 0.95 = 0.0975
5. 1 – (1-0.05 \* 0.05) = 0.0025
6. In both cases, if you have one disk failing, you won’t loss data. If two disks fail, RAID-5 will loose data for sure, but the two RAID-1 pairings might survive if appropriate disks fail..

# Problem 5 (10 pts)

* What are the problems associated with linked allocation of disk space routines?
* Which disadvantage is solved by the file allocation table?

Ans: The major problem is that a linked allocation can be used effectively only for sequential-access files. Another disadvantage is the space required for the pointers. Yet another problem of linked allocation is the decreased reliability due to lost or damaged pointers.

File allocation tables solve the problem of sequential access.

# Problem 6 (10 pts)

A page fault must be preceded by a TLB miss, but a TLB miss does not necessarily mean a page fault. True or false? Explain why (2-3 sentences)

Ans: True. If you have true TLB hit, the page is in the memory. It is possible that you have a TLB miss, but the page is in the memory, as the TLB is smaller than the page table.

# Problem 7 (10 pts)

Assume a system has a TLB hit ratio of 99%. It requires 10 nanoseconds to access the TLB, and 90 nanoseconds to access main memory. What is the effective memory access time in nanoseconds for this system?

Answer: 0.99 (10 + 90) + 0.01 (10 + 90 + 90) = 99 + 1.9 = 100.9

# Problem 8 (20 pts)

You have a hard drive with 150 tracks. Assume that the disk head’s last movement was from 90 to 100 and is currently on track 100. Consider the following series of disk track requests:

10, 30, 50, 90, 102, 120

Trace the following disk scheduling algorithms and for each of them:

1. draw a small bar graph of the distances the head must travel for each request (starting from beginning, so the second one will be longer, the third one even longer)
2. calculate the average distance the head must travel (numerical value required, don’t leave it as an expression)

# SSTF

Order: 100 , 102 (2), 90 (14), 120 (44), 50 (114), 30 (134), 10 (154)

Avg: (2 + 14 + 44 + 114 + 134 + 154) / 6 = 462 / 6 = 77

# SCAN

Order: 100, 102 (2), 120 (20), 90 (20 + 30 + 60 = 110, through 150), 50 (150), 30 (170), 10 (190)

Avg: (2 + 20 + 110 + 150 + 170 + 190) / 6 = 642 / 6 = 107

# C-LOOK

Order: 100, 102 (2), 120 (20), 10 (20+110 = 130), 30 (150), 50 (170), 90 (210)

Avg: (2+20 + 130 + 150 + 170 + 210) / 6 = 682 / 6 = 113.66