CMSC 412 Midterm #2 (Spring 2016) - Solution

1.) (20 points) Define and explain the following terms:

a) Negative ACL

In a file system a listing of who is not allowed to perform some action on a file or directory.

b) Set uid bit

A bit that indicates that a program should be run as the owner of the program's executable file rather than the user who runs the program (useful to run programs at higher privilege)

c) Bankers Algorithm

An algorithm to prevent (avoid) deadlock by ensuring that there exists a safe sequence of resource allocations should a specific request be granted.

d) Dirty bit

A bit set by the hardware to indicate that a page frame has been written to. It is cleared by the OS when the OS writes the page frame to disk.

2.) (20 points) Memory Systems

a) (8 points) Consider an architecture with a two-level page table, but without a TLB. Accessing a memory location takes 100ns. Accessing a disk block takes an average of 10ms. What percentage of accesses can result in page faults if the average access time needs to be less than 350ns?

10ms = 10,000,000 ns. Each ref (page fault or not) needs 300ns, so need miss rate such that 10,000,000ns = x * 50ns. thus x= 0.0005%

b) (6 points) Why is having a processor that supports a single "large reach" map of an arbitrary numbers of page frames from a contiguous region of physical memory to a virtual range useful? Explain what performance gains are possible, and one type of application that can use this type of mapping?

The purpose is to eliminate TLB misses (especially with random accesses to memory). Examples of uses include HPC systems (where one user process is all that runs on a node) and virtualization (i.e. VMware).

c) (6 points) Name two parts of the OS kernel that can't be made pageable

Page out code
Page fault handler code
Dispatcher
APIC/IOAPIC

3.) (20 Points) Synchronization: Use binary semaphores to implement a solution to the standard readers/writers problem (i.e. readers preferred) that allows either a single writer or at most five readers at a time. Show all variable and semaphore declarations and initial values.

Semaphore wsem =1, mutex = 1, rsem = 0

Int readCount = 0 wReader = 0

```
READER: while (1) {
        P(mutex)
        If (readCount == 5)
                wReader++
                V(mutex)
                P(rSem)
        else if (readCount == 0)
                P(wsem)
                readCount++
                V(mutex)
        else
                readCount++
                V(mutex)

        // READ

        P(mutex)
        If (readCount == 5 && wReader)
                wReader—
                V(rsem)
        Else
                readCount—
                if (readCount == 0) V(wsem)
        V(mutex)


WRITER: while (1)

        P(wsem)
        // WRITE
        V(wsem)
```

4.) (20 points) File Systems

a) (7 points) When hard links are added to a tree based file system it can then be a DAG which can make tools such as disk usage harder to write. Does adding symbolic links instead allow for DAGs? Are there any additional concerns for these types of tools with symbolic links?

Yes, DAGs are possible (in fact any use of a symbolic link results in a DAG structure vs. a tree).

Additional problems with symbolic links include links to files that no longer exist and cycles caused by a symbolic link pointing to a directory along its path to the file system root.

b) (7 points) Describe two reasons a first fit policy for allocating disk blocks might result in lower performance or available storage for files.

Leads to fragmentation which can result in space not being useable (i.e. can't store a fie since no continuous blocks are available even though total blocks free exceeds requested file size)

Additional problem is the time to search the free list to find a first fit allocation.

c) (6 points) Consider a disk with 512 byte blocks. If you used a single block to implement a bit vector free list, how big of disk could you support?

One sector can hold 512 bytes/sector x8 bytes/bit = 4096 bits/sector. Each bit references one 512 byte sector so total storage is 4096 x 512 = 2x 1024 x 1024 = 2MB

5.) (20 points) Project

a) (6 points) If you leave the 0^{th} entry in the 0^{th} page table (i.e. 0^{th} page dir entry) as invalid, what happens to the physical memory associated with this page?

It is not used (it is left un-allocated).

b) (6 points) Why do you need to pin the page while waiting for the disk to read the data for that page form the mapped file?

It needs to be pinned so that the OS can't remove this page from memory. This is a problem since DMA used to read bytes from a disk won't know the page has been removed from memory and would overwrite whatever is there now.

c) (8 points) In the project, writes to the memory region of a mapped file resulted in changes the file. If you wanted to allow files to be mapped into memory, but that any modifications to memory should not result in changes to the file, how could you do this (you may not use a second copy of the file)?

Use a COW (copy on write approach). Keep the modified copy in memory only. If it needs to be paged out, put any dirty pages into the paging file.