CS350 Final Review

For what reasons might a process fail to open a file?

1. The file does not exist.
2. The user process does not have the correct permissions.
3. The filename format is incorrect.
4. The file is empty.

For what reasons might a process fail to open a file?

1. The file does not exist.
2. The user process does not have the correct permissions.
3. The filename format is incorrect.
4. The file is empty.

A 5400rpm disk has 2^10 tracks with capacity 2^32 bytes. Each track has 2^8 sectors. Maximum seek time is 20ms.

1. What is the capacity of each track?
2. What is the capacity of each sector?
3. How long does it take for a full rotation of the disk?
4. How long does it take to read 10 consecutive sectors?
5. How long does it take to seek 400 tracks away from the current head position?

A 5400rpm disk has 2^10 tracks with capacity 2^32 bytes. Each track has 2^8 sectors. Maximum seek time is 20ms.

1. What is the capacity of each track? 2^32/2^10 = 2^22 bytes
2. What is the capacity of each sector? 2^22/2^8 = 2^14 bytes
3. How long does it take for a full rotation of the disk?

    5400/60 = 1/x -> x = 11.1ms

4. How long does it take to read 10 consecutive sectors?

    (10/2^8)11.1ms = 0.436ms

5. How long does it take to seek 400 tracks away from the current head position?

    400/2^10*20ms = 7.8125ms

Enumerate the disk operations when executing “ls /usr/bin/local”. Assume local is a directory.

Enumerate the disk operations when executing “ls /usr/bin/local”. Assume local is a directory.

1. Read / i-node
2. Read / data block to retrieve usr i-number
3. Read usr i-node
4. Read usr data block to retrieve bin i-number
5. Read bin i-node
6. Read bin data block to retrieve local i-number
7. Read local i-node
8. Read local data block

Give a pseudocode implementation of a function that adds two global arrays (A+B=C) of the same length such that there is one thread for each element of the array.

Give a pseudocode implementation of a function that adds two global arrays (A+B=C) of the same length such that there is one thread for each element of the array.

```
void AddKernel( void * data, long int idx )

{

        C[idx] = A[idx] + B[idx];

}

void SumArrays()

{

        For i = 1 to length(A)

                thread_fork( “add”, null, AddKernel, null, i );

}
```

Suppose MIPS had an atomic instruction SwapNotEqual such that:

```
int SwapNotEqual( int &x, int y )
{
    If ( x != y )
       Int tmp = x;
       x = y;
       return tmp;
    return y;
}
```

Implement a spinlock using SwapNotEqual.

Suppose MIPS had an atomic instruction SwapNotEqual such that:
Implement a spinlock using SwapNotEqual.

```
int lock = 1; // lock is free, let 0 mean the lock is NOT free.

While ( !SwapNotEqual( lock, 0 ) ) {}
```

Reason:

If lock = 1; then SwapNotEqual(1, 0) sets lock to 0 and returns 1. !SNE = 0, so no more looping.

If lock = 0; then SwapNotEqual(0, 0) returns 0. !SNE = 1, so continue looping.

The Linux Completely Fair Scheduler assigns each thread a weight that is proportional to what share of CPU time the thread should receive. Suppose two threads A and B have weights 10 and 11 respectively.   A has an actual runtime of 5 and B has an actual runtime of 10. If the total thread weight is 100, which of thread A or B will run next?

The Linux Completely Fair Scheduler assigns each thread a weight that is proportional to what share of CPU time the thread should receive. Suppose two threads A and B have weights 10 and 11 respectively. A has an actual runtime of 5 and B has an actual runtime of 10. If the total thread weight is 100, which of thread A or B will run next?

Virtual Runtime Thread A = AR(TOTAL/W) = 5(100/10) = 50

Virtual Runtime Thread B = AR(TOTAL/W) = 10(100/11) = 90.9

Since VR(A) < VR(B) Thread A will run next.

Long-running, non-interactive computation threads that do not block will filter down to the lowest priority queue in the multi-level feedback queue scheduling algorithm. If there are significantly more high-priority and blocking threads running, these lower priority threads may not run. How does the multi-level feedback queue method ensure these threads do not starve?

Long-running, non-interactive computation threads that do not block will filter down to the lowest priority queue in the multi-level feedback queue scheduling algorithm. If there are significantly more high-priority and blocking threads running, these lower priority threads may not run. How does the multi-level feedback queue method ensure these threads do not starve?

Periodically takes all threads in all queues and puts them into the highest-priority queue.

Solaris does this once per second.

A prankster placed an easter egg in a major OS. If you type “this os sucks” at the command prompt a kernel-privilege program executes and V’s every semaphore in the kernel exactly once. Does the kernel program succeed in V’ing every semaphore, or, does it panic when it tries to V? Why? (For the sake of this question, do not consider the side-effects of V’ing random semaphores).

A prankster placed an easter egg in a major OS. If you type “this os sucks” at the command prompt a kernel-privilege program executes and V’s every semaphore in the kernel exactly once. Does the program succeed in V’ing every semaphore, or, does it panic when it tries to V? Why?

The code succeeds in V’ing every semaphore because semaphores do not store/check if the thread that P’d is the thread that V’d.

A device driver writes some data to a data register and then writes to a command register. When the device is done the requested operation the OS needs to acknowledge the completion before any other threads can use the device. Give the pseudocode for the driver and handler.

A device driver writes some data to a data register and then writes to a command register. When the device is done the requested operation the OS needs to acknowledge the completion before any other threads can use the device. Give the pseudocode for the driver and handler.

```
Driver
    P( device semaphore )
    Write to data register
    Write to command register
Handler
    Acknowledge completion to device
    V( device semaphore )
```

On-demand paging only loads program data into the address space when it is needed. When is this detected?

If we use the Clock Replacement Algorithm, what must the kernel do to frame entries when choosing a victim?

On-demand paging only loads program data into the address space when it is needed. When is this detected?

If we use the Clock Replacement Algorithm, what must the kernel do to frame entries when choosing a victim?

- Detected on TLB miss

- Until a victim is found (an entry whose use bit is not set); kernel sets the use bit of non-victims to 0

A system uses 48bit virtual addresses and 64bit physical addresses. Pages are 64kb (2^16) in size.

1. What is the maximum size of physical and virtual memory?
2. How many pages are there in virtual memory? How many frames in physical memory?
3. How many bits are needed for the page offset?
4. If a PTE is 16 bytes (2^4), how many fit on a page?
5. How many levels are needed for the multi-level page table if each table must fit onto a page?

A system uses 48bit virtual addresses and 64bit physical addresses. Pages are 64kb (2^16) in size.

1. What is the maximum size of physical and virtual memory?

   2^64 bytes, 2^48 bytes

2. How many pages are there in virtual memory? How many frames in physical memory?

   Pages = 2^48/2^16 = 2^32; Frames = 2^64/2^16 = 2^48

3. How many bits are needed for the page offset? 16
4. If a PTE is 16 bytes (2^4), how many fit on a page? 2^16/2^4 = 2^12
5. How many levels are needed for the multi-level page table if each table must fit onto a page?

   (48-16)/12 = 32/12 -> 3 levels [8 bit directory index][12 bit page index][12 bit page index][16 bit offset]

OS/161 raised an exception. EPC = 0x80001024, VADDR = 0x0 (read).
What code was executing when this exception was raised? What happened?
How should the OS respond?

OS/161 raised an exception. EPC = 0x80001024, VADDR = 0x0 (read). What code was executing when this exception was raised? What happened? How should the OS respond?

Kernel code was executing.

Attempt to dereference NULL.

The OS should panic because it's a bug in the kernel code.

A TLB miss occurs while executing sys_getpid. Draw the kernel stack at this point.

A TLB miss occurs while executing sys_getpid. Draw the kernel stack at this point.

trapframe
mips_trap
syscall
sys_getpid
trapframe
mips_trap
vm_fault

Why is shortest-job-first not a feasible scheduling algorithm?

Why is shortest-job-first not a feasible scheduling algorithm?

It requires that job length is known.   Job length cannot usually be known and is subject to machine performance and state.

Which of the following could NOT be valid virtual addresses in OS/161 and why?

1. 0x0000 0000
2. 0x8000 8900
3. 0xB346 0020
4. 0xFFFF FFFF

Which of the following could NOT be valid virtual addresses in OS/161 and why?

1. 0x0000 0000 invalid user address; out-of-segment
2. 0x8000 8900 valid kernel address; kseg0
3. 0xB346 0020 valid kernel address; kseg1
4. 0xFFFF FFFF invalid kernel address; kseg2 (not used)

In VSFS, why can two files not have the same file identifier?

In VSFS, why can two files not have the same file identifier?

File identifiers correspond to i-numbers, which are the index of the files i-node in the i-node array. If two unique files accidently shared the same i-number, then both hard links (for file A and B) would point to the same file.

In a CV, why does the thread sleep on the CV’s wait channel and not the locks?

In a CV, why does the thread sleep on the CV's wait channel and not the locks?

The thread is waiting on a condition to be met, it is not waiting to acquire the lock. If it DID sleep on the locks wait channel, how could it be woken without waking those threads that ARE waiting for the lock?

Why does disk block size match page size?

Why does disk block size match page size?

To minimize I/O service request time for on-demand paging. It would require only one I/O operation per Page Fault.

Why is defragmentation not useful for SSDs?

Why is defragmentation not useful for SSDs?

No performance advantage for random I/O vs. sequential I/O.

Defragmentation performs extra, unnecessary writes. Also, affects wear-leveling, causing premature disk failure (conversion to read-only mode).

What are barrier semaphores used for?

What are barrier semaphores used for?

When a thread needs to wait for a set of other threads to complete before proceeding. Example: when a parent thread needs to wait for child-threads to complete before cleaning up memory and tabulating results.

List all of the disk reads/writes for opening existing file /usr/bin/foo.a

List all of the disk reads/writes for opening existing file /usr/bin/foo.a

1. R / inode
2. R / data block to find usr hardlink
3. R usr inode
4. R usr data block to find bin hardlink
5. R bin inode
6. R bin data block to find foo.a hardlink
7. R foo.a inode

How to study

1. Read over your assignments
2. Re-do the midterm
3. Try previous 2-3 midterms
4. Do review questions posted to course website
5. Read “extra” notes and guides posted to piazza
6. Do extra sample problems and quizzes posted to piazza
7. Become fast at doing virtual memory calculations and disk drive calculations
8. Study with a group; ask each other questions on piazza/facebook/blah, etc.