CSS-430 : Operating Systems : Mid-Term-Exam

David Liu

The Mid-Term Exam will be held online on 04-May-2020, starting at 5:45 pm, for 2 hours. Deadline for uploading answers to Canvas is 8:00 pm (allowing an extra 15 minutes for any upload glitches).

During the exam you may consult notes, books, lectures or the Internet.

Please answer all 20 questions.

Good luck!

1. 5 points.   
   **What is the difference between user mode and kernel mode?**

Kernel mode allows for code execution to have complete and unrestricted access to the underlying software.

User mode has no ability to directly access hardware or reference memory

**How does kernel mode prevent Apps from taking control of the computer?**

There really isn’t an effective strategy. You can try your best by verifying all processes or isolating the problem. You can also make sure apps run in user mode so they don’t have kernel access and elevating individual applications required privilege access. (Least privilege)

**Can we build a secure Operating System without kernel mode?**

Yes, but it would be extremely difficult to provide sufficient security within the software that operates the custom OS. The OS and anything underlaying it can be hacked using user mode.

**Do popular processors support any modes in addition to user and kernel?**

Nope 😊

1. 5 points.  
   **What is the Von Neumann bottleneck?**

It is a limitation of throughput caused by the computer architecture from the lack of processors or the general ability of those processors. The limited throughput caused by the CPU transfer rate and the amount of memory.

**The term was invented many decades ago. Is it still relevant?**

Yes, it serves as the very basis for stored-program computer concepts. It is used in computers, laptops, and high performance computers.

**What strategies do modern computers follow to avoid any Von Neumann bottlenecks?**

You can use CPU caches, branch predictor logic, and parallel computing to compensate for the lack of processing power or distribute it evenly.

1. 5 points.  
   **What is an interrupt?**

Generated by hardware, it stops a running program and continues it as if nothing ever happened. They are used for multi-tasking operating systems and are typically used for hardware clocks.

**What is a trap?**

Synchronous exception that handles invalid memory access within a user process.

**We describe one as synchronous, the other as asynchronous. What does this mean? And which is which?**

A synchronous operation blocks a process until the process is complete and asynchronous is vice versa (non-blocking). Interrupt – Asynchronous and Trap - Synchronous  
**What is each used for in an operating system such as Linux?**   
 An interrupt can be used as a hardware clock.

A trap can be an arithmetic error - division / zero.

**Can a user program cause a trap?**

Yes, they can be caused by catching arithmetic errors or invalid system routines.

1. 5 points.  
   **What challenges are faced by an Operating System for mobile devices, compared with a deskside PC?**

A mobile device has limited hardware within a small compact device.

Lack of storage within a mobile device.

Hardware upgrades are limited.

**Name two popular Operating Systems used in mobile devices.**

Android and iOS

1. 5 points.  
   **What is a “system call” or “syscall”?**

When a program requests a service from the kernel (they are run in kernel mode)

**Name any 3 system calls in an Operating System.**

Pipe() fork() sleep()  
**Does a syscall execute in user mode, or kernel mode?**

Kernel mode  
**Does the calling process undergo a context switch when it makes a syscall?**

No, a context switch occurs whenever the kernel transfers control of the CPU from an executing process. An example would be doing a syscall to get the date and time.  
**Does the calling thread undergo a context switch when it makes a syscall?**

No, a context switch occurs whenever the kernel transfers control of the CPU from an executing process. An example would be doing a syscall to get the date and time.

A full context switch would involve swapping both the process and the thread via

* Exit
* Using up its time slice
* Requires another resource that isn’t available
* A resource has become available for a sleeping process
* Encounters a semaphore

1. 5 points.  
   **There are two main techniques used for IPC (“Inter Process Communication”). Name them.**

Shared memory and Message passing

**Which technique is best for large transfers? Why?**  
 Cooperating process b/c it **shares memory** between systems allowing for an unbounded buffer and the ability to break up the process into multiple threads.

**Which technique is used to communicate with remote PCs?**

Communication within client/server architectures use **sockets.**

**Explain briefly the differences in synchronization needs of the two techniques**

Synchronous and asynchronous synchronization where synchronous is blocking the send so that the receiver can receive the message. (Shared Memory) Non-blocking (asynchronous) allows for the sender to send the message and continue. (Message Queues)

1. 5 points.  
   **Explain the difference between a Monolithic kernel and a Micro kernel?**

* Monolithic kernels:
  + Large Kernel size
  + Needs to compile the entire source code
  + No crash recovery
  + More secure
  + Old
  + Difficult to bug fix
  + Fast
* Micro kernel:
  + Small kernel size
  + Extensible and easy to troubleshoot
  + Slower
  + Subsystems can be compiled independently

**Which one is “better”, and why?**

In my opinion, the microkernel is better because it allows for easier management of code b/c it’s split up into smaller user space services. It’s also more secure as most of its processes don’t run in kernel mode. If one user space fails, it doesn’t effect the entire system as the monolithic structure does.

**Is Linux monolithic or micro?**

monolithic

**Is Windows monolithic or micro?**

Monolithic

1. 5 points.  
   **The memory for a process is divided into 4 sections. What are they called?**

Stack, heap, text, and data

**Which for the 4 sections are writeable?**

Data, Stack, heap

**Which for the 4 sections are executable?**

Stack, Heap

1. 5 points.  
   **What is a context switch?**

Kernel transfers control of the CPU from an executing process to another that is ready to run

**Describe the main steps involved in a Process context switch.**

Saves the context of the current process that is currently running on the CPU

Updates the PCB (Process Control Block)

Moves the process in the PCB to queue

Selects a new process for execution

Updates the PCB

Restores the context of the previous process  
**Describe the main steps involved in a Thread context switch.**

Initiated by interrupt

Core state is saved on TCB(Thread Control Block)

Saves the stack pointer

Chooses the next thread

Updates the TCB

Loaded back the core state of the selected thread.

**Which one is faster? And why?**

Thread context switch because they have the same virtual memory maps vs. you have to map and cache a whole process.

1. 5 points.  
   **40% of a certain process can be parallelized. Use Amdahl’s law to calculate the speedup on a 2-core system. A 3-core system? A 4-core system?**

2 core system:

* + 1 / (0.6 + (1.0 – 0.6) / 2) = 1.25 speedup gain

3 core system:

* + 1 / (0.6 + (1.0 – 0.6) / 3) = 1.36 speedup gain

4 core system:

* + 1 / (0.6 + (1.0 – 0.6) / 4) = 1.43 speedup gain

1. 5 points.  
   **Describe the difference between so-called user-mode threads, and kernel-mode threads.**

User mode’s executing code has no ability to directly access hardware or reference memory.

Kernel mode has complete and unrestricted access to the underlying hardware.

**Can both make use of multiple cores? Explain your answer.**

Yes, they both make use of multiple cores to take advantage of the high performance that comes with running multiple threads concurrently.

**Which are lighter-weight (in the sense the consume less CPU to start, and to context switch)?**

User mode is more light weight b/c kernel mode has a lot more overhead than user mode.

**Which kind are “green” threads?**

Threads that are scheduled by a runtime library or VM.

1. 5 points.  
   **Suppose our computer has only 1 core. Suppose we have 3 large jobs to run, and each takes an hour. How is it possible that running them side-by-side in the computer will take less than 3 hours.**

Multitasking allows for the core to schedule all 3 jobs simultaneously by splitting them up into separate tasks and scheduling them properly by which task needs CPU time.

1. 5 points.  
   **Explain the meaning of the following times, associated with CPU scheduling:**

* Arrival time
  + The time when a process arrives into the ready state to be executed
* CPU Burst time
  + Given amount of time for execution for that process
* Response time
  + Time it takes from a process’s first submission to initial execution.
* Execution time
  + The time spent executing that task
* Turn-around time (“TAT”)
  + The time it takes from submission to completion
* Wait time
  + The time it the process spent waiting for gaining access to the CPU

1. 5 points.  
   **In CPU scheduling, answer the following yes/no questions?**

* Can wait time ever exceed execution time?
  + Yes, it depends on the scheduling method. (FCFS P1 – CPU Burst is 1000, P2 – CPU Burst is 1)
* Can response time ever be zero?
  + Yes, FCFS will choose the first process.
* is TAT = wait time + execution time?
  + Yes
* Can TAT ever be less than execution time?
  + No
* If wait time is zero, is response time also zero?
  + Yes

1. 5 points.  
   Consider the following processes with arrival time and CPU burst times:

|  |  |  |
| --- | --- | --- |
| Process | Arrival Time | CPU Burst Time |
| P1 | 4 | 6 |
| P2 | 0 | 5 |
| P3 | 5 | 4 |
| P4 | 8 | 3 |

**Draw the Gannt chart for the FCFS (First Come First Served) schedule.**

Note: no need to draw a real Gannt chart with different processes on different lines. Just write a text version, like this:  
  
 1 1 3 3 2 2 4 4 etc, where “1” means P1, “2” means P2 and so on. Let’s call this a “simplified Gannt chart”.

2 2 2 2 2 1 1 1 1 1 1 3 3 3 3 4 4 4

1. 5 points.  
   **For the same processes P1 thru P4, draw the simplified Gannt chart for the SJF (Shortest Job First) schedule.**

Non – preemptive

2 2 2 2 2 3 3 3 3 4 4 4 1 1 1 1 1 1

1. 5 points.  
   **For the same processes P1 thru P4, draw the simplified Gannt chart for the STR (Shortest Time Remaining)** schedule. Reminder: STR is like SJF, but with pre-emption possible at each ‘event’

2 2 2 2 2 3 3 3 3 4 4 4 1 1 1 1 1 1

1. 5 points.  
   **Describe RR (Round Robin) scheduling**

Time slices assigned to each process in a circular order in equal partitions

**What is a “quantum” in RR?**

The size of the time slice given to each process

**What is a typical size for quantum?**

It can be anywhere between ~ 10 – 100 ms

**What happens if we make the quantum too short?**

The overhead between context switches won’t make up for the CPU time actually being done on each process resulting in large TAT with more processes coming in.  
  
**What happens to fairness if we make the quantum too long?**

You risk starving the other processes that need CPU time.

1. 5 points.  
   **Can a program have more than 1 critical section?**

No, critical sections can only be used by a single process at a time.

**Can different threads compete to execute the same critical section?**

No, this results in a race condition between two threads accessing the same resource at relatively similar times resulting in wrong information being given between the 2 threads.

**Can different processes compete to execute the same critical section?**

Each process has its own critical section.

1. 5 points.  
   **Explain your understanding of the terms “deadlock” and “starvation”**

Deadlock happens when each member in a system is waiting on for the other member as well as itself to take action. It causes a hang in the system.

Starvation is when a low priority program doesn’t get any CPU time b/c a higher priority program is utilizing all the resources.

**How would you know that a multi-thread program has become “deadlocked”? What are its symptoms?  
Same question, but for “starvation”?**

You can check is a program has become deadlocked by detecting cycles with the resource allocation. E.g. Resource 1 -> Process 1 -> Resource 2 -> Process 2 -> Resource 1

Starvation detection requires future detection since record keeping can’t determine if the process has made any progress. Aging (giving priority to processes over time if they haven’t been accessed) can break starvation.