Justin Haysbert, Alex Olteanu, Mauryan Uppalapati

Tables

Identify and summarize issues of this course in the following tables. Some cells have already been filled to serve as an example.

1. Interesting terms and concepts (aka new fancy words you learned or revisited in this course)

Term and/or brief summary	**Computer system examples**	**Examples from different academic areas (including other branches of CS)**	**Examples from everyday life**
Caching (storing a copy of recently accessed data in a storage that's faster than the primary storage).	· Processor cache maintains small subset of data from main memory. · Flash drive is used as cache for HDD.	- Html pages are copied and stored on the disk - Upon accessing the file again the cache is compared to the internet file	
Trap(An instruction that temporarily switches a process from user mode to kernel mode)	· In a computer system, when a system call is made , The OS then issues the "trap" instruction,which enables kernel mode, which is what then allows the process to then be privy to sensitive information that it didn't have access to previously		

Kernel Mode(the os mode that gives a program full access to the machines hardware)	In a computer system , the kernel mode keeps the OS intact , most programming is done in user mode because otherwise if some program crashes it only affects that specific program and couldn’t theoretically affect the whole OS and brick the computer	The administrative access to academic research instruments like a large telescope , which lets you actually move the telescope to where you want to look rather than the guest access which only allows you access to data at where it is currently pointed	An abstract example could be , when instead of giving a person read only access to your assignment you give them read and write access , wherein they technically have full access.
System Call(system call is a way through which the a process can directly interface with the operating system)	In a computer system,using a system call , a process can then request information from the kernel, a part of the OS to which it does not have access to	Building on the above example , the request the research fellow could make to the department for information could be an example of a system call.	Building upon the same example from above , the request a user makes to gain access to the google doc could be an example of a system call.
Trap table(a table created by the machine at boot that contains the locations of trap handlers)	In a computer system, during a system call when a trap instruction is issued , the OS then looks up the memory address for the specific trap that the system call asked for , this arrangement of addresses in the table is always scrambled at boot which allows for greater security against outside threats		In banking , we only can see account numbers , but when the bank receives a transfer to a specific account it then looks up the specificities of the account and then does the necessary transaction

Policy(policies in operating systems are a method of determining what to do in a specific situation.)	In the computer system ,a multitude of policies are implemented for various situations. For example; in thread scheduling , which thread should run? Has policies implemented like first come first serve or the round robin policy for specific situations	In universities and even classes like these , are are polices that are standard : like the late submission policy for this class and it is a method of determining what to do in a specific situation	Most of our monetary system works solely due to policies. Such as the fiscal policy or the monetary policy
Interrupt(a signal that alerts the processor that immediate action is needed)	When the interrupt signal occurs , the OS stops the process and then waits on the action before proceeding		An example could be when my roomba gets entangled on a phone charger , it stops i.e “an interrupt” occurs and doesnt start until the cable has been removed
API(A software interface)	Api is basically a software interface wherein two applications can interact with one another without any need for user intervention		

2. Useful problem-solving approaches

Brief statement	**Computer system examples**	**Examples from different academic areas (including other branches of CS)**	**Examples from everyday life**

Periodically check the system status in order to detect errors early.	Periodically check that locks are available to eventually take them and enter the critical section	Periodically check the status of a biological experiment to catch obvious problems early.	Periodically check the status of the food in the oven to detect overcooking early
Virtualizing resources by making making every process think it has unlimited access	Every process thinks it has unlimited access to the cpu when in reality all process share a limited set of CPUs	Network virtualization allows for a view of the network that can be manipulated without altering the actual network	Creating a virtual scan of a document so that you can format and edit it without changing the 1 of 1 physical version
Using a different modes to enable access to certain controls	Using a privileged mode to execute system calls then returning to user mode and giving controls back to user programs	Higher ranking military officers having access to locations of certain units that aren't available to other ranks	Using parental controls to make changes to a kids gaming console then returning it to child mode where they can play games under supervision
Using Interrupts to halt a running processes and run a more important process	Timer interrupts are used to ensure that the cpu has time to run whatever processes it wants	When certain conditions within a program are reached the running code is interrupted and other code is executed Ex. while loop	When watching tv and my baby starts crying i can pause the tv and check on my baby because it is more important
Using forks to create two identical separate operations	The fork function allows us to create a parent and child process both execute the same things separately	In chemistry we may carry out the exact same experiment multiple times to ensure we get the same result	- We make multiple copies of a document in case something happens to the original - We solve a math problem twice to ensure we

			get the same outcome
Using multilevel feedback queues to change priorities	The multilevel feedback queue is used in cpu scheduling to track the time execution of processes and decides what process holds priority and can run	When designing an experiment we may order the experiment based on the amount of time certain steps take.	When going throughout our day we may mentally order tasks based on time they will take to complete. We may put shorter tasks in the forefront of our schedule
Using directories in the file system to store file data	Directories are used in the file system in order to store file data within the disk. We use files, directories and inodes to organize the disk and optimize its usage	We may file lab documents in different folders that pertain to the information we are storing allowing us to maintain high levels of organization	I may create a playlist on my computer containing several songs that I feel need to be in that playlist. For example, I'll put sad songs in a playlist titled sad songs. This also makes these songs easier to find

3. Neat solutions, hacks, design tricks

Brief statement	**Computer system examples**	**Examples from different academic areas (including other branches of CS)**	**Examples from everyday life**
Devising separate solutions for initial system startup, system resume, and normal system operation.	Different types of actions are performed at OS booting, during standard OS operations, and OS shutdown.		Similarly, different types of actions are performed at starting a car, continuous running of a car, and upon car shutoff.
Within certain operating systems, the process manager will favor interactive threads the user is directly engaging with such as the mouse, keyboard and visual windows. This ensures that the components of the computer that the user is directly interacting with appear smooth, sleek and responsive.	This is usually facilitated by the Kernel, which allows for multiple devices and the reading and handling of their input. Windows 7 notably employs this tactic.	This reminds me of practices in business where maintaining appearances is sometimes the most important driver of success. Although a company may be faced with internal disturbances, maintaining the image of a healthy, well-oiled organization will preserve consumer confidence in the business. Thus, these customers will continue to shop despite these hypothetical internal delays that would otherwise deter them away.	I am also vaguely reminded of interviews and job applications, where an employer may only be presented with the best qualities of an applicant. The applicant may in fact be less qualified below the surface, but maintaining a facade may ensure that the employer will continue to 'interact' with them. This is analogous to a computer and the user where the user is constantly judging the performance of the machine.
Virtualizing the resources of a computer and allowing the OS to serve as the 'brain' that mediates	The OS virtualizes the CPU, memory and disk, allowing programs to run in isolated and protected environments.	In Psychology experiments, the subjects of a study will often be unaware of the greater scope of the research. Each	Sometimes ignorance can be a good thing. If two employees are told that they are most essential to the company they work

between them is a clever way of safely and effectively managing how the machine operates.	This way, they do not exchange or exchange memory unless explicit permission is granted. Should one program be freely allowed to overwrite the memory of another, then everything would fall into chaos!	individual acts on his own accord, unaware of what they may actually be observed on within the trial. If participants could talk or exchange information, then the entire experiment could be ruined.	for, then each will be imbued with confidence and positive self-esteem. They may then work harder since they believe that they "run things around here." Programs are similar in that each believes that they have complete access to the resources of a machine, when in reality they are many many other programs just like them.
An operating system is considered robust if the implementation of a new feature does not interfere or disturb already existing components of the machine. However, a neat loophole can allow virtually any OS to be considered as robust if it simply blocks most editions of the feature entirely. Therefore, it is still technically robust because you cannot break the OS with an update! (This is moreso an abstract concept but I still thought it was interesting and worth including).	An operating system will assess the potential impact of a software update instead of blindly installing it.	Theories in Physics, Biology, Chemistry or Math all build upon the work of another. In a very, very abstract sense then, the entire study of these subjects can be considered a robust system in itself. If a new theory emerges that quickly proves to be problematic or incorrect, it will simply be ignored by the greater community and the study of these fields will live on. Useful "additions" or "features"(software updates) to the system may either be adopted if useful, or discarded and ignored if limiting.	The internet is a perfect metaphor of a robust system. As stated in class earlier, it is a network that has been built upon for decades and scaled to an unbelievably all-encompassing global scale. Operating systems must similarly be able to build upon themselves. However, one lackluster website added to the internet will not bring the entire thing down, even if it is poorly coded. The rest of the network is able to operate

			independently of new additions.
In recent years, modern computing device design has begun to test the physical limits of resources and materials in our world. We represent bits with billions of electrical charges inside of our processors. However, as mentioned we are exceeding the physical limit of how many transistors we can fit into a single chip. This is still a fascinating solution to forever increasing demand for more and more bits.	In 1956, Gordon Moore theorized that the growth in average transistors per chip would exponentially grow, doubling every year. However, recently experts have begun discussing the possibility that this will end within the coming years.	The study of physics has arguably seen a select period of stagnation during the 20th century only to be revived in recent years given the advent of modern technology. In a similar way, sometimes the next breakthrough in technology is necessary to further a field, discipline or already existing technology.	As people grow and gain experience with time, we learn new skills, lessons and insights to deal with the existing problems in our lives.

4. Other (optional, bonus points)

If there are other interesting ideas from the course that don't fit into the above categories, name them and provide examples here. Essentially, you're looking to create a cheat-sheet that you could use as a reference for most essential course content in the future.

Fast file system design - When designing a fast file system its faster to access blocks that are closer together that way we don't have to wait for an abundant amount of spins of the disk

- Avoid seek times because this is what makes the process take so long
- Organize these inodes closer for faster performance
- We should always take the shortest path similar to how we would use the shortest gps path to the store

Writing new blocks - we have to update the block bitmap, then the data block, then the inode

- We can't just add a new block without writing it to the bitmap.
- This would be like creating a new city and not adding it to the map

    - Then how would we know where it is?

When building locks - we have to create locks for various reasons but mostly for managing processes.

- When building a lock we have to be able to initialize it, lock it, and unlock it
- If im lacking any of these capabilities the lock just wouldn't work
- We also have to ensure that multiple processes take the same lock and enter the critical section
- If the purpose of a lock on my bird's cage was to keep my dog from getting in, why would I lock my dog in the cage too?