**CH 1-2 and APP A**

- explain why operating systems (OS) need to exist and/or what OS do

  The OS is needed to manage a computer's hardware. It must provide a basis for application programs and acts as an intermediary between the user and hardware.

- identify and describe three or more issues that drove the development of the earliest operating systems

  1. Modifying tasks required a great deal of effort and manual labor.
  2. Job setup time too long, as few resources available + expensive
  3. CPU often sat idle

- explain the relationship between the user and CPU, and OS and the CPU, and how they differ

  The OS emphasizes the need to control various I/O devices and user programs. OS controls/allocates resources. The user is who uses the application programs that run on the CPU. In which, they are not in charge of the specifics of how that program will run on the CPU.

- explain the relationship between the OS and the computer hardware, including how it protects hardware from potential misuse

The OS manages computer hardware, provides a platform for software applications, and protects hardware from misuse by controlling access to system resources and limiting software actions.

- identify and briefly explain the difference between a system service and a system call

- A system call is the way a program requests a service from the kernel. The service is what is performed by executing a system call

- briefly explain the difference between a “dumb” terminal and a computer

- a dumb terminal can only display data

- briefly explain the difference between CLI, GUI, and batch operations

-CLI: uses text commands. Faster performance since directly using commands.
-GUI: window system with a mouse that serves as a pointing device to direct I/O, menu of options, etc., and a keyboard.
-Batch operations: Commands and directives to control the commands that are entered into files and then those files are executed (Queue jobs). Good for direct line to files and folders.

- explain what a virtual machine is and why it might be necessary

- a virtual machine is an OS that is installed on software that can imitate hardware. It can edit and test without harming actual hardware and create cross platform programs without having to account for the system's instruction set.

**Ch 3**

- define and explain what a process is, and how a process is used by Operating Systems (OS)

a program in execution. A program is passive while a process is active. A process is used by the OS to run a list of instructions from a file.

- define and explain what a Process Control Block (PCB) is and what it does

Serves as the repository for all the data needed to start, or restart, a process/

- identify and briefly explain the states in which a process may reside

A process can be in the new, running, waiting, ready or terminating state. New is when a process is created. Running is when the instructions are being executed. Waiting is when the process is waiting for an event to happen (I/O).

- explain the relationship between the long-term scheduler and the degree of multiprogramming?

The degree of multiprogramming is the number of processes currently in memory. This is an important piece of information for the scheduler

- briefly explain the actions that occur in the computer and/or OS during a context switch

a process will be saved in a PCB of a kernel while loading a saved part of a new process so that it can run.The parents can also pass initialization data.

**Ch 4**

- explain how a thread is different from a process.

Processes are programs loaded into memory and executing, threads are an executable segment of a process. A process is an executing program with multiple threads of control.

- explain why a programmer might use a thread instead of a process.

Threads allow for performing several tasks at the same time, and are much more efficient with certain tasks such as context switching.

- identify and explain at least three primary benefits of using multiple threads in a program

1. much more **responsive** since they can take user input as a separate operation

2. threads do not require techniques to be arranged by the programmer to make **resource sharing** available.

3. Economy: There is much less cost to create and context switch threads, since they share the resources of their processes.

- explain what task parallelism is, and provide an example that uses task parallelism.

Task parallelism is a computing method that distributes tasks across multiple computing cores. Each task performs a unique operation

- explain what data parallelism is, and provide an example that uses data parallelism.

Data parallelism distributes subsets of the same data across multiple computing cores, all performing the same operation.

- explain how a many-to-one threading model compares to a one-to-one threading model.

Many-to-one threading maps multiple user-level threads onto one kernel thread while one-to-one
maps each user-level thread onto a separate kernel thread. Many-to-one is lightweight but less scalable, while one-to-one offers better performance but is more resource-intensive.

- explain the difference between asynchronous and synchronous threading.

With asynchronous, the parent creates a child thread and resumes its execution, so that the parent and child execute concurrently and independently. With synchronous threading, the parent thread creates one or more children and must wait for all children to terminate before it resumes execution.

- explain the difference in resource usage between a thread and a process; provide an example.

Processes can share resources only through techniques such as shared memory and message passing. Such techniques must be explicitly arranged by the programmer. However, threads share the memory and the resources of the process to which they belong by default. The benefit of sharing code and data is that it allows an application to have several different threads of activity within the same address space.

**Ch 5**

- briefly explain the difference between preemptive and non-preemptive scheduling

non preemptive: When a process switches from the running state to the waiting state (for example, as the result of an I/O request or an invocation of wait() for the termination of a child process) and When a process terminates

- preemptive: When a process switches from the running state to the ready state (for example, when an interrupt occurs) and When a process switches from the waiting state to the ready state (for example, at completion of I/O)

- identify what kinds of actions the dispatcher needs to conduct, and why they need to be conducted

- The dispatcher gives control of the cps to the process selected by the scheduler.
- should be as fast as possible, since it is invoked during every context switch.

- explain how FCFS scheduling works, identify the benefit of using FCFS, and identify a problem with using FCFS

How: The process that requests the CPU first is allocated the CPU first

Benefit: Simple to understand, write, and manage

Problem: However, it often has long average waiting times. Troublesome for interactive systems where it is important that each process gets a share of the CPU at regular intervals

- explain how SJF scheduling works, identify the benefit of using SJF, and identify a problem with using SJF

- associates with each process the length of the process's next CPU burst. When the CPU is available, it is assigned to the process that has the smallest next CPU burst

- explain how RR scheduling works, identify the benefit of using RR, and identify a problem with using RR

How: - Similar to FCFS but preemption is added to enable the system to switch between processes. The queue is circular but also like the FIFO queue. The CPU scheduler picks the first process from the ready queue, sets a time to interrupt after X time quantum and dispatches the process.

Benefit: This will allow all processes to at least be run once so that a few won't get stuck at the end with the possibility of never running.

Problem: verage waiting time is often long

- explain different ways resources can be shared between a parent and a child process

In operating systems, resources can be shared between processes using shared memory, IPC mechanisms such as pipes and sockets, signals for synchronization, and semaphores for mutual exclusion. These mechanisms allow efficient and secure sharing of resources.

- briefly explain the differences and commonalities between RR, FCFS, SJF, and SRTF

The main common thing between all of the algorithms is that there are cases where each one could fall back into a FCFS protocol. If RR has all processes that finish before the quantum time then it'll be FCFS, same with if the SJF protocol has processes of all of the time to process. SJF and SRTF are almost the exact same except SFJ is nonpreemptive but SRTF is. RR is preemptive. FCFS is non preemptive

- identify and briefly explain the difference between coarse and fine grained multithreading, and provide an example

Coarse-grained multithreading switches between threads at large intervals, while fine-grained multithreading switches at small intervals. Examples include time-sharing in an OS for coarse-grained and simultaneous multithreading (SMT) for fine-grained.

**Ch 9**

- describe pages, frames, and page tables, and identify the relationship between each of them

PAGES - logical blocks of memory of the same size
FRAMES - fixed-sized blocks of physical memory
PAGE TABLES - contains the base address of each frame of physical memory, indexed by the logical page number
Pages are memory blocks, frames are physical units of memory, and page tables map them. They manage physical memory use by processes.

- explain the difference between internal and external fragmentation; provide examples as needed

Internal fragmentation refers to wasted space within a memory block or page due to allocating more memory than is needed, while external fragmentation refers to wasted space in memory due to the inability to allocate contiguous blocks of memory, often caused by variable-sized memory requests.

- explain commonalities and differences between first fit, next fit, worst fit, and best fit

First fit - Allocate the first hole that is big enough. Searching can start either at the beginning of the set of holes or at the location where the previous first-fit search ended. We can stop searching as soon as we find a free hole that is large enough.

Best fit - Allocate the smallest hole that is big enough. We must search the entire list, unless the list is ordered by size. This strategy produces the smallest leftover hole.

Worst fit - Allocate the largest hole. Again, we must search the entire list, unless it is sorted by size. This strategy produces the largest leftover hole, which may be more useful than the smaller leftover hole from a best-fit approach.

SOMEONE ASK ABOUT NEXT FIT

- explain what page hits and misses means, and what the consequences of each might be

hit - When we want to load the page on the memory, and the page is already available on memory

miss - when the page is not already available on memory, may result in OS or app crash

**Ch 10**

- explain pages and frames, and the differences and commonalities between them

Pages and frames are fixed-size blocks of memory used for virtual and physical memory management, respectively. They enable efficient memory allocation and division into smaller chunks. Pages are managed by the OS, while frames are managed by hardware.

- describe virtual memory, pre-paging, demand paging, page management and replacement

- Virtual memory: Use more memory than physically available.
- Pre-paging: Anticipate and load pages before they are needed.
- Demand paging: Load pages only when needed.
- Page management: Allocate and manage virtual memory pages.
- Page replacement: Swap out pages when there is no more available space.

- analyze and explain the various paging algorithms such as FIFO, LRU, LFU, OPT, Clock, and MFU

- FIFO (First In, First Out) - pages are replaced in the order they were brought in.
- LRU (Least Recently Used) - pages that have not been accessed recently are replaced first.
- LFU (Least Frequently Used) - pages with the fewest references are replaced first.
- OPT (Optimal) - the page that will not be used for the longest time is replaced.
- Clock - pages are kept in a circular buffer and a pointer moves around to determine which page to replace.
- MFU (Most Frequently Used) - pages with the most references are replaced last.