The basic job of an OS is to abstract hardware into friendly interfaces for programmers.The difference between kernel and user programs are: Kernel modules have separate address space Kernel modules have higher execution privilege Kernel modules do not execute sequentially. Kernel modules can be interrupted. Kernel modules must be preemptable. Kernel modules can share data The two major views of what constitutes the OS are a resource manager or extended machine. The four architectural stages are: vacuum tubes (single user), transistors (batch systems), ICs (multiprogramming), PCs (multithreaded)

\* What is spooling?

Computer Hardware Review: The major point is to understand these things

in the context of the OS. For example, the OS has to deal with issues

of managing the memory hierarchy, and to an extent, it takes advantage

of that hierarchy by using one level as a cache for the next level down.

Computer Hardware Review

\* Understand why the OS must "know" certain details of the underlying

architecture. What various details might these be?

\* What are user mode and kernel mode (again)?

\* What's the difference between virtual and physical addresses?

\* I/O:

+ What is programmed (aka synchronous) I/O?

+ What is interrupt-driven (aka asynchronous) I/O?

+ What is DMA?

\* What is caching used in general for in the memory storage hierarchy?

The Operating System Zoo

\* Identify/distinguish the nine types of operating systems (related

to architecture, not internal OS organization).

Operating System Concepts

\* What's a process? What comprises one?

\* What is deadlock, informally?

\* Other than the sort of "mundane normal" ideas, why are files used in an OS?

\* What is a pipe?

System Calls

\* What are they used for?

\* Unix system calls:

+ fork

+ wait, waitpid

+ variants of exec

+ kill

+ chdir, getcwd

+ a few file management calls: open, close, read/write, lseek

+ a few directory management calls: mkdir, rmdir, link, unlink, mount, umount

+ (Some of the above are related to program 1; others are from the notes or both.)

+ I won't specifically ask you how to program these or what the exact format is. Mainly, you should expect me to ask what they are used for as well as certain associated properties.

+ Why does Windows decouple the name of system call and the API function?

\* How the shell gets things done.

Operating System Structure

\* Identify/distinguish the six identified types of operating systems (related

to internal OS organization, not architecture).

\* What is a virtual machine?

\* What is the difference between an exokernel and a virtual machine?

\* What are the advantages of the client/server (microkernel) approach?

Chapter 2

Processes

\* What is a process?

\* What are the principal events causing process creation?

\* What are the principal events causing process termination?

\* What are the various states a process may be in, and how might it

get into that state?

\* What is a process table entry (sometimes called process control block) for? What information does it maintain?

Multiprogramming: what is it?

Threads

\* What is a thread?

\* What are the benefits of threads?

\* Compare/contrast user-space threads and kernel-space threads.

\* What's the big disadvantage of user threads on a single-threaded kernel?

\* Briefly describe, compare/contrast the various methods of building

servers (in this context).

\* Describe pop-up thread

\* Issues in converting single-threaded to multithreaded:

+ global variables

+ reentrant code

+ signal handling

+ stack management

Making single-threaded code multithreaded:

\* what are the pitfalls?

IPC:

Interprocess Communication

\* What are the three major issues concerning IPC?

\* What's a race condition?

\* What's a critical region (critical section)?

\* Busy waiting:

+ What is a busy wait? A spin lock?

+ What's wrong with disabling interrupts?

+ What's the issue with using software-only lock variables?

+ What's the problem with strict alternation?

+ Peterson's solution: what's up with that?

+ TSL: understand how it works, its properties, etc. Likewise the similar XCHG.

+ Describe what priority inversion is, what causes it.

\* Describe/understand the producer-consumer problem

\* Semaphores and mutexes:

+ what an up and down does in a semaphore

+ difference between binary and counting semaphore

+ How semaphores might be implemented, depending on architecture.

+ relevance of mutex actions to threads

\* Monitors:

+ What are the various advantages and disadvantages? Where are

they implemented, and how?

+ What's a condition variable?

\* How might message-passing might be applied to producer-consumer?

\* What is a barrier?

Classical IPC Problems

\* Describe/understand the one classical IPC problems listed.

\* What does it model?