**Delayed writes: Disadvantage:** Will result in a data loss if the system crashes at the wrong time. **Dual-mode CPUs:** **Advantage**: Prevent user processes from directly accessing disk drives and other peripherals. **Timer Interrupts:** **Advantage**: Prevents user processes from monopolizing a CPU core. **DMA Controllers:** **Advantage**: Speed up data transfers between the disk drive and the main memory. **Lightweight Processes:** **Advantage**: Are much cheaper to create than conventional processes. **Microkernels: Disadvantage**: Introduce additional context switch delays in the processing of requests. **Modular Kernels: Disadvantage:** Let users add new features to the kernel. **Time sharing: Advantage:** Allows multiple interactive users to share the same computer. **Multi-threaded servers: Advantage:** Can process multiple client requests in parallel.

**Delayed Writes:** Reduce the number of physical disk accesses, let the disk scheduler optimize physical disk access, and do not guarantee the durability of recent updates. **Compared to the time a processer takes to perform a cycle, disk accesses are:** 10 to 20 million times slower. **What is the purpose of timer interrupts?** They notify the kernel that a user process has exceeded its time slice. **How does a processor switch from user mode to privileged mode?** Through the interrupt mechanism. **Why is memory protection always implemented in the hardware?** Because it needs to be done very fast. **Under which conditions will a process go from the RUNNING state to the READY state?** It loses the CPU to another process (interrupt**). When should the OS swap processes out of the main memory?** When there is not enough space in main memory. **What happens when a process receives a SIGKIL signal?** It terminates. **What do pthreads do?** They use a portable library, they provide very fast context switches within threads of a same task, and they use pthread\_create(…) to create new threads. They do **NOT** let multi-threaded programs use multiple cores at the same time.

**Lightweight processes:** Share the address space of their parent. **Timer interrupts:** Limit the amount of time a process can occupy a core. Microkernel: Delegates most of its duties to user-level servers. **Ready Queue:** Contains all the processes waiting for the CPU. **Privileged mode:** Allows the CPU to execute input/output instructions. **fork():** Creates a new process. **execvp():** Loads in memory the program to be executed by a given process. **kill():** Sends a signal to another process. **signal():** Specifies what a process should do when it receives a signal. **DMA:** Gives a disk controller direct access to the main memory.

**How to preventing address spaces of other processes?** By adding memory Protection. **In a Dual-mode CPU, how can the CPU switch from user mode to privileged mode?** when it processes an interrupt ( as the interrupt will leave the program counter in the safe location inside the kernal) **Difference between hard and soft deadline?** Missing soft deadline is inconvenience while missing hard deadline causes catastrophe. **What if Unix had remained in assembly language?** It would not have ported to architecture and would NOT have had the same impact. **Why would fork() be a costly system calls?** b/c it requires making a complete copy of the address space of the forking process. **In which state is a process performing a disk I/O?** In the block or waiting state **what event brings a blocked process into ready queue?**  Completion of a pending system request **what is the default action that a Unix process takes when receiving a signal? how can it be prevented? Is it always possible?** process terminates. with catch signal/signal() system call. no. not always possible eg. sigkil().

**Why is memory protection always implemented in hardware?** Because any software solution would be too slow, and it must be done for every memory reference. **What is the major disadvantage of the master slave organization in OS for multiprocessor architectures?** It requires all kernel functions to be performed on a single processor, which may become a bottle neck **What is the major disadvantage of single-threaded file servers?** They can only process one request at a time **Why should all multiprocessor operating systems support kernel-level threads?** It is the best way to allow parallel user programs to run on more than one core **Which processes should we not suspend unless we absolutely need to do so?** Ready processes

**What is the major disadvantage of user-level threads? a.)** You cannot use blocking system calls. .**b.)** The whole set of threads sharing an address space will be blocked each time one  of the threads does a blocking system call

**Main disadvantage of delayed writes?** Will result in a data loss if the system crashes at the wrong time.

**Main advantage of dual-mode CPUs ?** Prevent user processes from directly accessing disk drives and other peripherals.

**Main advantage of timer interrupts?** Prevent user processes from monopolizing a CPU core.

**Main advantage of DMA controllers?** Speed up data transfers between the disk drive and the main memory. On the other hand, they do not provide user processes with direct access to disk drives and other peripherals. Would be very bad!

**Main advantage of lightweight processes?** Are much cheaper to create than conventional processes.

**Main disadvantage of microkernels?** Introduce additional context switch delays in the processing of requests.

**Main advantage of modular kernels?** Let users add new features to the kernel.

**Main disadvantage of modular kernels?** Increase the risk of system crashes.

**Main advantage of time sharing?** Allows multiple interactive users to share the same computer.

**Main advantage of multi-threaded servers?** Can process multiple client requests in parallel.

**How can we prevent processes from accessing the address spaces of other processes? How can we prevent user processes from tampering with the kernel?** By adding memory protection.

**In a dual-mode CPU, how can the CPU switch from user mode to privileged mode?** When it processes an interrupt (as the interrupt will leave the program counter in a safe location INSIDE the kernel.)

**What is the main difference between real-time applications with hard and soft deadlines?** Missing a hard deadline can have catastrophic consequences while missing a soft deadline is a mere inconvenience

**What would have happened if Unix had remained written in assembly language?** It would not have been ported to other architectures and would NOT have had the same impact.

**Why is fork( ) one of the costliest system calls?** Because it requires making a copy of the address space of the forking process.

**What is the default action that a Unix process takes when it receives a signal?** The process terminates.

**What can it do to prevent that from happening?** The process can catch the signal

**Is this always possible?** NO, the SIGKIL signal cannot be caught.NO, signal number nine cannot be caught.