Sabbir Ahmed

DATE: March 7, 2018

CMSC 421: HW 01

---

1 Compare and contrast microkernel and monolithic kernel-based operating systems. Name one kernel that follows each of these models.

Ans A microkernel implements user and kernel services in different address spaces, whereas a monolithic kernel system uses the same address space. This separation makes microkernels easily extendable with new services being added to its user space. The monolithic kernel requires the entire kernel to be rebuilt after adding new services. Since only the kernel services are located in its address spaces, the microkernels are comparatively smaller than a monolithic kernel. Also, if a service fails in a microkernel, the operating system remains unaffected. Monolithic kernels, however, execute faster as they use system calls to communicate between application and hardware. Microkernels rely on the relatively slower message passing to communicate. Performance of microkernels are also affected by increased system-function overhead. Monolithic kernels also require lesser code in their design than microkernels.

2 What is the purpose of a system call? Give at least 3 concrete examples of potential system calls in an OS and explain why each would be a system call.

Ans System calls provide an interface to the services made available by an operating system. Some examples of system calls include reading user inputs from a keyboard and a mouse. These are system calls because they interface the hardware directly using low-level routines. Another example of a system call is keeping track of and changing the work directory, which is important for the system to know the user's relative and absolute path. Opening and creating files are also system calls, since they interface with the memory of the system.

3 Describe what a context switch is. How does context switching differ between processes and between threads within a single process?

Ans When an interrupt occurs, the system needs to save the current context of the process running on the CPU so that it can restore that context when its processing is done. Context switching is the task of switching the CPU to another process. It requires performing a state save of the current process and a state restore of a different process. Context switching is much more expensive and slower than switching between threads in a multithreaded system. Because threads share the resources of the process to which they belong, it is more economical to create and context-switch threads.

4 Name two methods of IPC (interprocess communication) and discuss the pros and cons of each.

Ans The two fundamental models of interprocess communication are shared memory and message passing. Message passing is useful for exchanging smaller amounts of data, because no conflicts need be avoided. It is also easier to implement in a distributed system than shared memory. Shared memory can be faster than message passing, since message-passing systems are typically implemented using system calls. It however suffers from cache coherency issues, which arise because shared data migrate among the several caches.

5 What is the difference between a program being executed by a single thread and one that is being executed by multiple threads? Compare and contrast task parallelism and data parallelism as it relates to multithreaded programs.

Ans A program executed on a single thread will force all its tasks to run sequentially. If an application consists of multiple tasks, the program will have to wait for each of them to terminate successfully before starting the next steps of the procedure. Running a program on multiple threads allows concurrent execution of its tasks. Data parallelism focuses on distributing subsets of the same data across multiple computing cores and performing the same operation on each core. Task parallelism involves distributing threads across multiple computing cores.