KERNEL

Peter J. Denning, Naval Postgraduate School, Monterey, California

January 2008
Rev 5/26/08

Abstract: The kernel is the subset of operating system functions that must be executed in privileged mode.

Keywords: kernel, supervisor, nucleus, microkernel, interrupt

The kernel is the subset of an operating system's functions that must be executed in privileged or supervisor mode, so that they might properly protect users and allocate resources among them. The privileged mode gives those programs unrestricted memory access and the right to executive sensitive instructions. Kernel functions typically include:

1. Basic Input/Output System (BIOS). Routines giving a process access to the essential I/O devices connected to the system -- the display, keyboard, mouse, hard disks, and network. BIOS starts those devices and responds to their interrupts when they are done.

2. Interrupt Management. Routines for transferring the processor to a high priority task shortly after a signal indicates the need for the task. The names of interrupt handling routines are in an "interrupt vector". The actual routines are elsewhere -- for example, the page fault handler is part of the memory manager (see below), and the network packet received handler is part of the interprocess communication manager (see below).

3. Process Management. Routines for switching processors among processes; using the clock interrupt for round-robin scheduling; sending semaphore signals among processes; creating and removing processes; and controlling entry to supervisor state.

4. Memory Management. Routines for mapping virtual to real addresses; transferring pages between main memory and secondary memory; controlling multiprogramming load; and handling page fault and protection violations.

5. Interprocess Communication. Routines for opening and closing connections between processes; transferring data over open connections; handing network protocols and routing; and processing name service.

6. Security. Routines for enforcing the access and information-flow control policies of the system, changing protection domains, and encapsulating programs.

The kernels of early operating systems were constrained by memory limitations to be small. MIT's Compatible Time Sharing System (1960) had a total of 256 kilobytes of main memory, of which half was allocated to the kernel. Early Unix systems (1972) also had small kernels. By the early 1980s, system such as Berkeley Unix grew their kernels to take advantage of larger main memory. However, large kernels were more error prone, less secure, and slow.

Operating systems designers introduced the microkernel as a way to minimize the operating system code that had to be main-memory resident and operate in supervisor mode. Only the bare essentials were included; all other routines were executed outside the kernel in the user's memory space. For example: a device driver can be executed in user mode; only the instructions that invoke it and receive its interrupts are privileged. Also, file access control lists are stored in the file system; the microkernel security manager simply verifies that a requested access is permitted by the access list. This strategy has led to very efficient, fast, and compact microkernels for many operating systems.