LINFO

# Kernel Mode Definition

*Kernel mode*, also referred to as *system mode*, is one of the two distinct modes of operation of the CPU (central processing unit) in Linux. The other is *user mode*, a *non-privileged* mode for *user programs*, that is, for everything other than the *kernel*.

When the CPU is in kernel mode, it is assumed to be executing *trusted* software, and thus it can execute any instructions and reference any memory addresses (i.e., locations in memory). The kernel (which is the core of the operating system and has complete control over everything that occurs in the system) is *trusted* software, but all other programs are considered *untrusted* software. Thus, all user mode software must request use of the kernel by means of a *system call* in order to perform privileged instructions, such as *process* creation or *input/output* operations.

A system call is a request to the kernel in a Unix-like operating system by an *active process* for a service performed by the kernel. A process is an executing instance of a program. An active process is a process that is currently advancing in the CPU (while other processes are waiting in memory for their turns to use the CPU). Input/output (I/O) is any program, operation or device that transfers data to or from the CPU and to or from a peripheral device (such as disk drives, keyboards, mice and printers).

The Linux kernel was *non-preemptive* through Version 2.4. That is, while a process is in kernel mode, it cannot be arbitrarily suspended and replaced by another process (i.e., preempted) for the duration of its *time slice* (i.e., allocated interval of time in the CPU), in contrast to user mode, except when it voluntarily relinquishes control of the CPU. Processes in kernel mode can, however, be interrupted by an *interrupt* or an *exception*.

An *interrupt* is a signal to the operating system that an *event* has occurred, and it results in a change in the sequence of instructions that is executed by the CPU. In the case of a *hardware interrupt*, the signal originates from a hardware device such as a keyboard (i.e., when a user presses a key), mouse or *system clock* (a circuit that generates pulses at precise intervals that are used to coordinate the computer's activities). A *software interrupt* is an interrupt that originates in software, and it is usually triggered by a program in user mode. The standard procedure to change from user mode to kernel mode is to call the software interrupt 0x80. An exception is an unusual condition, for example an invalid instruction in a program.

All processes begin execution in user mode, and they switch to kernel mode only when obtaining a service provided by the kernel. This change in mode is termed a *mode switch*, not to be confused with a *context switch* (although it sometimes is), which is the

switching of the CPU from one process to another.

When a user process runs a portion of the kernel code via a system call, the process temporarily becomes a *kernel process* and is in kernel mode. While in kernel mode, the process will have *root* (i.e., administrative) privileges and access to key system resources. The entire kernel, which is not a process but a controller of processes, executes only in kernel mode. When the kernel has satisfied the request by a process, it returns the process to user mode.

Some CPUs, including the nearly ubiquitous x86-compatible (i.e., Intel-compatible) processors, are designed to accommodate more than two execution modes. However, all standard kernels in Unix-like operating systems utilize only kernel mode and user mode.

Some operating systems, such as MS-DOS (the predecessor to the Microsoft Windows operating systems) do not have a distinct kernel mode; rather, they allow user programs to interact directly with the hardware components. However, Unix-like operating systems use the dual mode mechanism to hide all of the low level details regarding the physical organization of the system from application programs launched by the user as a means of assuring system stability and security.

The Linux kernel Version 2.6 (which was introduced in late 2003) is *preemptive*. That is, a process running in kernel mode can be suspended in order to run a different process. This can be an important benefit for *real time* applications (i.e., systems which must respond to external events nearly simultaneously).

Unix-like kernels are also *reentrant*, which means that several processes can be in kernel mode simultaneously. However, on a single-processor system, only one process, regardless of its mode, will be progressing in the CPU at any point in time, and the others will be temporarily blocked until their turns.

Created November 30, 2004. Updated June 6, 2006.