**SYSTEM MODEL**

A system consists of a finite number of resources to be distributed among a number of competing processes. The resources are partitioned into several types, each consisting of some number of identical instances. Memory space, CPU cycles, files, and I/O devices (such as printers and DVD drives) are examples of resource types. If a system has two CPUs, then the resource type CPU has two instances. Similarly, the resource type printer may have five instances.

A process must request a resource before using it and must release the resource after using it. A process may request as many resources as it requires to carry out its designated task. Obviously, the number of resources requested may not exceed the total number of resources available in the system. In other words, a process cannot request three printers if the system has only two.

Under the normal mode of operation, a process may utilize a resource in only the following sequence:

**1. Request:** The process requests the resource. If the request cannot be granted immediately (for example, if the resource is being used by another process), then the requesting process must wait until it can acquire the resource.

**2. Use:** The process can operate on the resource (for example, if the resource is a printer, the process can print on the printer).

**3. Release:** The process releases the resource.

The request and release of resources may be system calls, examples are the request() and release() device, open() and close() file, and allocate() and free() memory system calls. Similarly, the request and release of semaphores can be accomplished through the wait() and signal() operations on semaphores or through acquire() and release() of a mutex lock. For each use of a kernel-managed resource by a process or thread, the operating system checks to make sure that the process has requested and has been allocated the resource. A system table records whether each resource is free or allocated. For each resource that is allocated, the table also records the process to which it is allocated. If a process requests a resource that is currently allocated to another process, it can be added to a queue of processes waiting for this resource.

A set of processes is in a deadlocked state when every process in the set is waiting for an event that can be caused only by another process in the set. The events with which we are mainly concerned here are resource acquisition and release.

To illustrate a deadlocked state, consider a system with three CD RW drives. Suppose each of three processes holds one of these CD RW drives. If each process now requests another drive, the three processes will be in a deadlocked state. Each is waiting for the event “CD RW is released,” which can be caused only by one of the other waiting processes. This example illustrates a deadlock involving the same resource type.

Deadlocks may also involve different resource types. For example, consider a system with one printer and one DVD drive. Suppose that process Pi is holding the DVD and process Pj is holding the printer. If Pi requests the printer and Pj requests the DVD drive, a deadlock occurs.