decisions. The user's group IDs are also included in every associated process and thread.

In the course of normal system use, the user ID and group ID for a user are sufficient. However, a user sometimes needs to **escalate privileges** to gain extra permissions for an activity. The user may need access to a device that is restricted, for example. Operating systems provide various methods to allow privilege escalation. On UNIX, for instance, the *setuid* attribute on a program causes that program to run with the user ID of the owner of the file, rather than the current user's ID. The process runs with this **effective UID** until it turns off the extra privileges or terminates.

1.7 Virtualization

Virtualization is a technology that allows us to abstract the hardware of a single computer (the CPU, memory, disk drives, network interface cards, and so forth) into several different execution environments, thereby creating the illusion that each separate environment is running on its own private computer. These environments can be viewed as different individual operating systems (for example, Windows and UNIX) that may be running at the same time and may interact with each other. A user of a **virtual machine** can switch among the various operating systems in the same way a user can switch among the various processes running concurrently in a single operating system.

Virtualization allows operating systems to run as applications within other operating systems. At first blush, there seems to be little reason for such functionality. But the virtualization industry is vast and growing, which is a testament to its utility and importance.

Broadly speaking, virtualization software is one member of a class that also includes emulation. **Emulation**, which involves simulating computer hardware in software, is typically used when the source CPU type is different from the target CPU type. For example, when Apple switched from the IBM Power CPU to the Intel x86 CPU for its desktop and laptop computers, it included an emulation facility called "Rosetta," which allowed applications compiled for the IBM CPU to run on the Intel CPU. That same concept can be extended to allow an entire operating system written for one platform to run on another. Emulation comes at a heavy price, however. Every machine-level instruction that runs natively on the source system must be translated to the equivalent function on the target system, frequently resulting in several target instructions. If the source and target CPUs have similar performance levels, the emulated code may run much more slowly than the native code.

With virtualization, in contrast, an operating system that is natively compiled for a particular CPU architecture runs within another operating system also native to that CPU. Virtualization first came about on IBM mainframes as a method for multiple users to run tasks concurrently. Running multiple virtual machines allowed (and still allows) many users to run tasks on a system designed for a single user. Later, in response to problems with running multiple Microsoft Windows applications on the Intel x86 CPU, VMware created a new virtualization technology in the form of an application that ran on Windows. That application ran one or more **guest** copies of Windows or other native x86 operating systems, each running its own applications. (See Figure 1.16.)

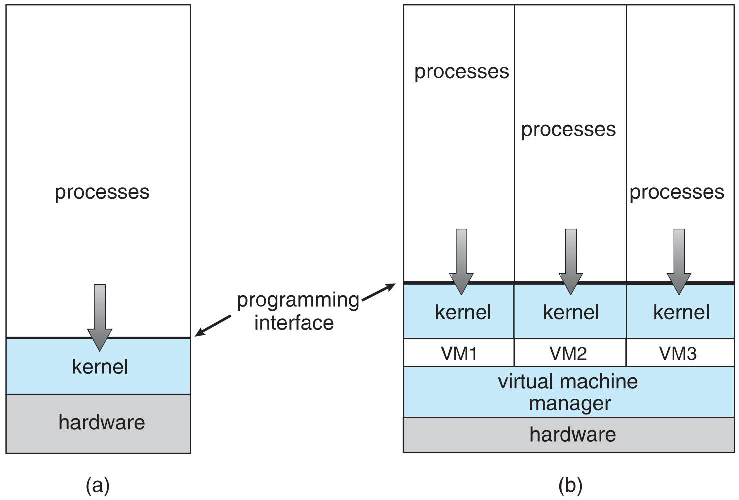

Figure 1.16 A computer running (a) a single operating system and (b) three virtual machines.

Windows was the **host** operating system, and the VMware application was the **virtual machine manager** (**VMM**). The VMM runs the guest operating systems, manages their resource use, and protects each guest from the others.

Even though modern operating systems are fully capable of running multiple applications reliably, the use of virtualization continues to grow. On laptops and desktops, a VMM allows the user to install multiple operating systems for exploration or to run applications written for operating systems other than the native host. For example, an Apple laptop running macOS on the x86 CPU can run a Windows 10 guest to allow execution of Windows applications. Companies writing software for multiple operating systems can use virtualization to run all of those operating systems on a single physical server for development, testing, and debugging. Within data centers, virtualization has become a common method of executing and managing computing environments. VMMs like VMware ESXand Citrix XenServer no longer run on host operating systems but rather *are* the host operating systems, providing services and resource management to virtual machine processes.

With this text, we provide a Linux virtual machine that allows you to run Linux—as well as the development tools we provide—on your personal system regardless of your host operating system. Full details of the features and implementation of virtualization can be found in Chapter 18.

1.8 Distributed Systems

A distributed system is a collection of physically separate, possibly heterogeneous computer systems that are networked to provide users with access to the various resources that the system maintains. Access to a shared resource increases computation speed, functionality, data availability, and reliability. Some operating systems generalize network access as a form of file access, with the details of networking contained in the network interface's device driver.

Others make users specifically invoke network functions. Generally, systems contain a mix of the two modes—for example FTP and NFS. The protocols that create a distributed system can greatly affect that system's utility and popularity.

A **network**, in the simplest terms, is a communication path between two or more systems. Distributed systems depend on networking for their functionality. Networks vary by the protocols used, the distances between nodes, and the transport media. **TCP/IP** is the most common network protocol, and it provides the fundamental architecture of the Internet. Most operating systems support TCP/IP, including all general-purpose ones. Some systems support proprietary protocols to suit their needs. For an operating system, it is necessary only that a network protocol have an interface device—a network adapter, for example —with a device driver to manage it, as well as software to handle data. These concepts are discussed throughout this book.

Networks are characterized based on the distances between their nodes. A **local-area network** (**LAN**) connects computers within a room, a building, or a campus. A **wide-area network** (**WAN**) usually links buildings, cities, or countries. A global company may have a WAN to connect its offices worldwide, for example. These networks may run one protocol or several protocols. The continuing advent of new technologies brings about new forms of networks. For example, a **metropolitan-area network** (**MAN**) could link buildings within a city. BlueTooth and 802.11 devices use wireless technology to communicate over a distance of several feet, in essence creating a **personal-area network** (**PAN**) between a phone and a headset or a smartphone and a desktop computer.

The media to carry networks are equally varied. They include copper wires, fiber strands, and wireless transmissions between satellites, microwave dishes, and radios. When computing devices are connected to cellular phones, they create a network. Even very short-range infrared communication can be used for networking. At a rudimentary level, whenever computers communicate, they use or create a network. These networks also vary in their performance and reliability.

Some operating systems have taken the concept of networks and distributed systems further than the notion of providing network connectivity. A **network operating system** is an operating system that provides features such as file sharing across the network, along with a communication scheme that allows different processes on different computers to exchange messages. A computer running a network operating system acts autonomously from all other computers on the network, although it is aware of the network and is able to communicate with other networked computers. A distributed operating system provides a less autonomous environment. The different computers communicate closely enough to provide the illusion that only a single operating system controls the network. We cover computer networks and distributed systems in Chapter 19.

1.9 Kernel Data Structures

We turn next to a topic central to operating-system implementation: the way data are structured in the system. In this section, we briefly describe several fundamental data structures used extensively in operating systems. Readers