Enter virtualization. Instead of giving the customer an account on the server, the provider gives him a fresh *virtual machine* to use as he pleases. The customer then can run any operating system and any applications (see Figure 7-1). Virtualization software ensures that these are isolated from the rest of the machine (which may be leased out to more customers). The *hypervisor*, on which the virtual machines run, contains two main parts: a *reference monitor*, which makes sure that no virtual machine can access another virtual machine's resources (especially its data), and a *scheduler*, which ensures that each virtual machine gets a fair share of the CPU.

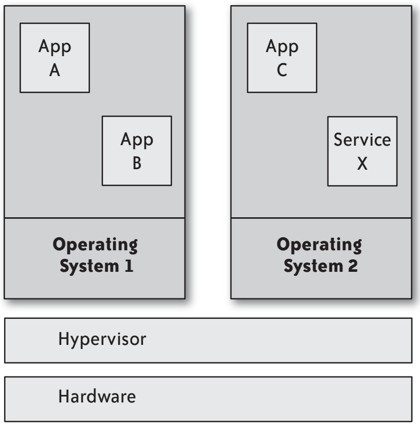

FIGURE 7-1. Virtual machine architecture

COULDN'T YOU JUST USE AN OPERATING SYSTEM?

Time-sharing operating systems have existed since the early 1960s, and enable several mutually distrusting users to run processes at the same time. Would it not be sufficient to give each user an account on, say, a Unix-based machine?

This would certainly let users share the computational resources. However, it is unsatisfactory because the user has much less flexibility and performance isolation.

In terms of flexibility, the user could only run software that is compatible with the machine's operating system; there is no way to run a different operating system, or to change the operating system. Indeed, it would be impossible for the user (without administrative support) to install software that requires root permissions.