1.1 Computing Systems

Computing system Computer hardware, software, and data, which interact to solve problems

Computer hardware The physical elements of a computing system

Computer software The programs that provide the instructions that a computer executes

In this book we explore various aspects of computing systems. Note that we use the term "computing system," not just "computer." A computer is a device. A **computing system**, by contrast, is a dynamic entity, used to solve problems and interact with its environment. A computing system is composed of hardware and software, and the data that they manage. **Computer hardware** is the collection of physical elements that make up the machine and its related pieces: boxes, circuit boards, chips, wires, disk drives, keyboards, monitors, printers, and so on. **Computer software** is the collection of programs that provide the instructions that a computer carries out. And at the very heart of a computer system is the information that it manages. Without data, the hardware and software are essentially useless.

The general goals of this book are threefold:

- To give you a solid, broad understanding of how a computing system works
- To develop an appreciation for and understanding of the evolution of modern computing systems
- To give you enough information about computing so that you can decide whether you wish to pursue the subject further

The rest of this section explains how computer systems can be divided into abstract layers and how each layer plays a role. The next section puts the development of computing hardware and software into historical context. This chapter concludes with a discussion about computing as both a tool and a discipline of study.

Layers of a Computing System

A computing system is like an onion, made up of many layers. Each layer plays a specific role in the overall design of the system. These layers are depicted in Figure 1.1 and form the general organization of this book. This is the "big picture" that we will continually return to as we explore different aspects of computing systems.

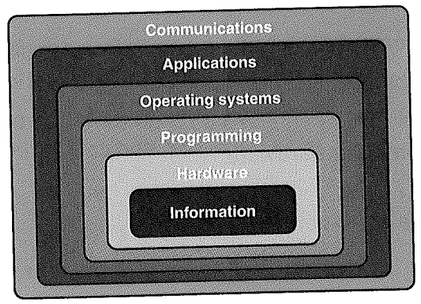


FIGURE 1.1 The layers of a computing system

You rarely, if ever, take a bite out of an onion as you would an apple. Rather, you separate the onion into concentric rings. Likewise, in this book we explore aspects of computing one layer at a time. We peel each layer separately and explore it by itself. Each layer, in itself, is not that complicated. In fact, a computer actually does only very simple tasks—it just does them so blindingly fast that many simple tasks can be combined to accomplish larger, more complicated tasks. When the various computer layers are all brought together, each playing its own role, amazing things result from the combination of these basic ideas.

Let's discuss each of these layers briefly, and identify where in this book these ideas are explored in more detail. We essentially work our way from the inside out, which is sometimes referred to as a bottom-up approach.

The innermost layer, information, reflects the way we represent information on a computer. In many ways this is a purely conceptual level. Information on a computer is managed using binary digits, 1 and 0. So to understand computer processing, we must first understand the binary number system and its relationship to other number systems (such as the decimal system, the one humans use on a daily basis). Then we can turn our attention to how we take the myriad types of information we manage—numbers, text, images, audio, and video—and represent them in a binary format. Chapters 2 and 3 explore these issues.

The next layer, hardware, consists of the physical hardware of a computer system. Computer hardware includes devices such as gates and circuits, which control the flow of electricity in fundamental ways. This core electronic circuitry gives rise to specialized hardware components such as the computer's central processing unit (CPU) and memory. Chapters 4 and 5 of the book discuss these topics in detail.

The programming layer deals with software, the instructions used to accomplish computations and manage data. Programs can take many forms, be performed at many levels, and be implemented in many languages. Yet, despite the enormous variety of programming issues, the goal remains the same: to solve problems. Chapters 6 through 9 explore many issues related to programming and the management of data.

Every computer has an operating system (OS) to help manage the computer's resources. Operating systems, such as Windows XP, Linux, or Mac OS, help us interact with the computer system and manage the way hardware devices, programs, and data interact. Knowing what an operating system does is key to understanding the computer in general. These issues are discussed in Chapters 10 and 11.

The previous (inner) layers focus on making a computer system work. The applications layer, by contrast, focuses on using the computer to solve specific real-world problems. We run application programs to take advantage of the computer's abilities in other areas, such as helping us design a building or play a game. The spectrum of area-specific computer software tools is far-reaching and involves specific subdisciplines of computing, such as information systems, artificial intelligence, and simulation. Application systems are discussed in Chapters 12, 13, and 14.

Computers no longer exist in isolation on someone's desktop. We use computer technology to communicate, and that communication is a fundamental layer at which computing systems operate. Computers are connected into networks so that they can share information and resources. The Internet, for example, evolved into a global network, so that there is now almost no place on Earth that you cannot communicate with via computing technology. The World Wide Web makes that communication relatively easy; it has revolutionized computer use and made it accessible to the general public. Chapters 15 and 16 discuss these important issues of computing communication.

Most of this book focuses on what a computer can do and how it does it. We conclude with a discussion of what a computer *cannot* do, or at least cannot do well. Computers have inherent limitations on their ability to represent information, and they are only as good as their programming makes them. Furthermore, it turns out that some problems cannot be solved at all. Chapter 17 examines these limitations of computers.

Sometimes it is easy to get so caught up in the details that we lose perspective on the big picture. Try to keep that in mind as you progress through the information in this book. Each chapter's opening page reminds you of where we are in the various layers of a computing system. The details all contribute a specific part to a larger whole. Take each step in turn and you will be amazed at how well it all falls into place.

Abstraction

The levels of a computing system that we just examined are examples of abstraction. An **abstraction** is a mental model, a way to think about something, which removes or hides complex details. An abstraction leaves only the information necessary to accomplish our goal. When we are dealing with a computer on one layer, we don't need to be thinking about the details of the other layers. For example, when we are writing a program, we don't have to concern ourselves with how the hardware carries out the instructions. Likewise, when we are running an application program, we don't have to be concerned with how that program was written.

Abstraction A mental model that removes complex details

Numerous experiments have shown that a human being can actively manage about seven (plus or minus two, depending on the person) pieces of information in short-term memory at one time. This is called Miller's law, based on the psychologist who first investigated it.[1] Other pieces of information are available to us when we need it, but when we focus on a new piece, something else falls back into secondary status.

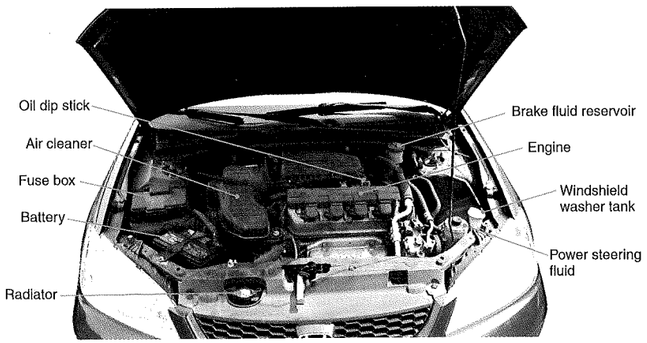
This concept is similar to the number of balls a juggler can keep in the air at one time. Human beings can mentally juggle about seven balls at once, and when we pick up a new one, we have to drop another. Seven may seem like a small number, but the key is that each ball can represent an abstraction, or a chunk of information. That is, each ball we are juggling can represent a complex topic as long as we can think about it as one idea.

We rely on abstractions every day of our lives. For example, we don't need to know how a car works to drive one to the store. That is, we don't really need to know how the engine works in detail. You need to know only some basics about how to interact with the car: how the pedals and knobs and steering wheel work. And we don't even have to be thinking about all of those things at the same time. See Figure 1.2.

Even if we do know how an engine works, we don't have to think about it while driving. Imagine if, while driving, we had to constantly think about how the spark plugs ignite the fuel that drives the pistons that turn the crankshaft. We'd never get anywhere! A car is much too complicated for us to deal with all at once. All the technical details would be too many balls to juggle at the same time. But once we've abstracted the car down to the way we interact with it, we can deal with it as one entity. The irrelevant details are ignored, at least for the moment.

Abstract art, as the name implies, is another example of abstraction. An abstract painting represents something, but doesn't get bogged down in the details of reality. Consider the painting shown in Figure 1.3, entitled *Nude Descending a Staircase*. You can see only the basic hint of the woman or the staircase, because the artist is not interested in the details of exactly how the woman or the staircase looks. Those details are irrelevant to the effect the artist is trying to create. In fact, the realistic details would get in the way of the issues that the artist thinks are important.

Abstraction is the key to computing. The layers of a computing system embody the idea of abstraction. And abstractions keep appearing within individual layers in various ways as well. In fact, abstraction can be seen throughout the entire evolution of computing systems, which we explore in the next section.



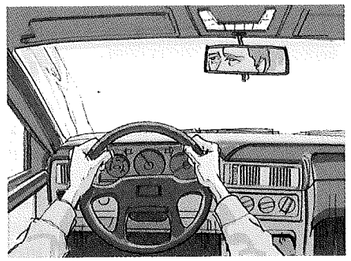


FIGURE 1.2 A car engine and the abstraction that allows us to use it