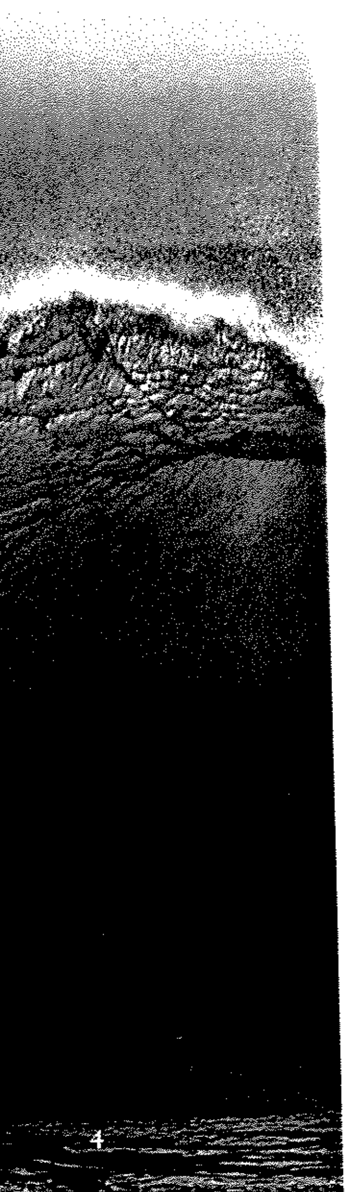
It would appear that we have [as a society] reached the limits of what it is possible to achieve with computer technology, although one should be careful with such statements as they tend to sound pretty silly in 5 years.

—JOHN VON NEUMANN, COMPUTER PIONEER 1947

TO BECOME Fluent, we need to learn the language of information technology (IT). The people who create IT are notorious for using acronyms, jargon, and everyday words in unusual ways. Acronyms like WYSIWYG (WHIZ·zee·wig), "what you see is what you get," are often meaningless even after you learn what the letters stand for. (We'll come back to the WYSIWYG story later.) Jargon like "clicking around," for navigating through an application or a series of Web pages, is meaningful only after you have actually done it. And an everyday term like **window**, which was originally chosen to suggest the idea of a portal to the computer, may no longer be an appropriate metaphor for the sophisticated computer concept. It is not surprising that coming across these terms for the first time is confusing. But is such techno-speak any weirder than, say, medical or musical terms? (For example, *bilirubin* and *hemidemisemiquaver* may sound like odd words unless you're a doctor or a musician.) Technology, like medicine and music, makes more sense when we are familiar with the vocabulary.

Our first goal is to understand why learning the right term is essential to any new endeavor. Next, we ask some simple questions about computers (like "Where's the Start button?") to review terms you may already know. But, for almost every familiar term, there is a new word or idea to learn. We also introduce some new words for the physical and computational parts of the computer. These are mostly terms you have probably heard before, but learning exactly what they mean will help you make them part of your everyday vocabulary. Along the way, we explain basic ideas like how buttons are created and how they are clicked. This starts to demystify the computer's virtual world and introduces the ideas of process and algorithm. Then we introduce the "idea" terms of IT, words like *abstraction* and *generalization*. These terms refer to deep concepts, and devoting a few minutes to understanding them will pay off throughout the rest of your Fluency study. Finally, we close with interesting stories about how people and computers have advanced, as we become more analytical in our thinking.



IT adopts many strange terms because it is filled with new ideas, concepts, and devices that never before existed. The inventors name them so they can describe and explain their ideas to others. Acronyms are common, because as engineers and scientists develop ideas, they often abbreviate them with the letters of the concept's description. The abbreviation sticks, and if the concept is important, its use extends beyond the laboratory. For example, when engineers invented the "small computer system interface," they abbreviated it SCSI. People began to pronounce it "skuzzy" rather than saying S-C-S-I. Naming-by-abbreviation creates a terminology full of acronyms. Critics call it "alphabet soup." But we can understand the terms once we understand what all the letters stand for. For example, ROM is short for "read-only memory." Even without knowing much about computers, you can guess that ROM is a special type of memory that can be read but not written to.

Using the right word at the right time is one characteristic of an educated person. Perhaps the best term to learn to use well is the French (now also English) term *le mot juste* (luh·MO·joost), which means the right word or exact phrasing.

There are two important reasons for knowing and using *le mot juste*. First, understanding the terminology is basic to learning any new subject. When we learn what new words mean, we learn the ideas and concepts that they stand for. Our brains seem to be organized so that when we name a thing or idea, we remember it. For example, in ice hockey, *icing* is the term for hitting the puck across the blue lines and the opponent's goal line. We might not even notice this amid all the passing and slap shots. By knowing this new definition for the familiar word *icing*, we start watching for it, increasing our understanding and enjoyment of the game. Precision in using the term means precision in understanding the idea. Eventually the word stops sounding weird and becomes a part of our vocabulary. At that point, we use the right word without even thinking about it.

The second reason for knowing and using the right word is to communicate with others. If we use terms properly, people understand us. We are able to ask questions and receive help—something everyone starting out in a field needs to do. When seeking help in information technology, using the right word is especially important because we often must rely on email, the telephone, or an online help facility. We have to be precise and articulate because no one is by our side to help us describe what we need. A goal of Fluency is to be able to get help from such resources. The ability to use *le mot juste* allows us to communicate, get help, and ultimately, be self-reliant.

You can learn vocabulary by reading a computer dictionary, of course, but that's *waaay* too boring. Instead, we introduce the basic terminology by asking some simple IT questions that have unexpected answers.

Where's the Start Button?

Most computers are on all the time, which is why screen savers were invented. Screen savers—animations such as a kitten prancing around the screen or a changing geometric design—are programs that sleep when the computer is in use and wake up when the computer is idle. These moving images "save the screen." Without them, a single, unchanging image can "burn" into the screen, permanently changing the phosphorous surface and creating a "ghost" of the image, which interferes with viewing. (Recent technological advances have made burn-in less of a problem and many computers simply turn off the screen when the system is idle.) You can reactivate the computer by moving or clicking the mouse, or by pressing any key.

If computers are usually on, why bother to learn where the Start button is? Because sometimes they are off, and as we will see later, we might need to **cycle power**—turn the computer off and then back on. To know where to look for the power button, we need to know how a computer is organized.

Two Basic Organizations: Monolithic or Component

Some computers are sold as **components** with a separate monitor, computer and hard drive, speakers, and other devices. (We'll discuss the individual devices later.) Many desktop PCs are organized this way. The **monolithic** package, like an iMac or a laptop, has all the devices bundled together, as shown in Figure 1.1(a). The component approach lets you mix and match parts to fit your needs, as shown in Figure 1.1(b). The all-in-one monolithic design is simpler because manufacturers decide for you which components will form a balanced, effective system. Laptops are monolithic, of course, because it is inconvenient to carry around multiple parts.

In a monolithic design, the **power switch** (⏻) is on the back of the chassis or, often with Macs, on the keyboard. For component systems, the power switch is usually on a separate box near the display containing the CD/DVD drive. Most component monitors also have their own power switches, but they control power only for the monitor, not the whole computer.

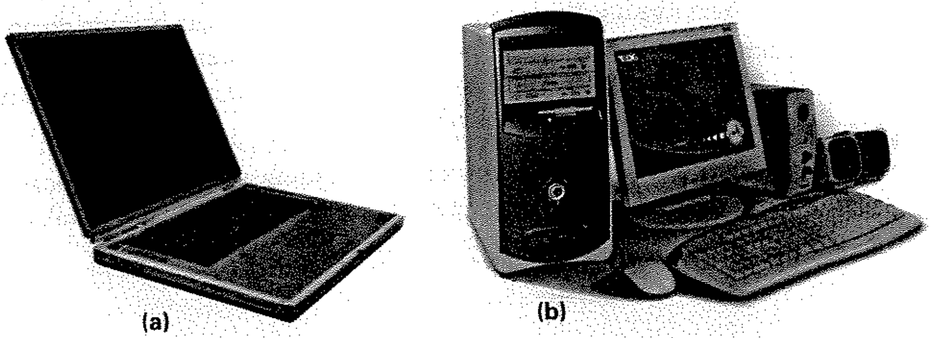
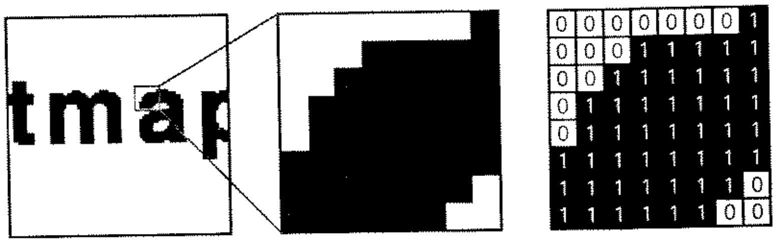


Figure 1.1 Examples of the (a) monolithic and (b) component systems.

The **monitor** is a video screen like a TV, but there are many differences. Unlike passive TVs, computers are interactive, so the monitor becomes more like a blackboard showing the information created by both the computer and the user as they communicate. Modern monitors are **bit-mapped**, meaning that they display information stored in (the bits of) the computer's memory, as illustrated in Figure 1.2. TVs generally display images live or from recorded tape, captured with a camera. The big, bulky monitors are **cathode ray tubes** (CRTs), whereas the slim, flat displays are **liquid crystal displays** (LCDs).



An enlargement of a monitor's display of the word bitmap and the corresponding bits for each pixel.

To emphasize the contrast between TVs and computers, notice that television can only show "reality"—the images *recorded* through a camera's lens. A computer creates the images it displays in its memory; the images don't have to exist in physical reality. The world that a computer creates is called **virtual reality**.

The components and the computer must be connected to each other and they must be connected to an electrical power source. For power-hungry devices like monitors, separate **power cables** are usually used, which is why there is a separate power switch. For simple devices like the keyboard or mouse, the **signal** and power wires are combined into one cable. To help us plug in the cables correctly, the computer's sockets and the cable's plugs are usually labeled with icons, as shown in Figure 1.3. So, we simply match up the icons.

Computer component plugs fit into their sockets in only one way. After you figure out which way the plug goes in, insert it into the socket gently to make sure that the pins (the stiff wires making the connections) align and do not bend. Once the plug is inserted, push it in firmly.

fit ALERT

Damage Control. When connecting computer components (or other electronic devices), plug in the power connection last. When disconnecting, remove the power connection first. Remember PILPOF—plug in last, pull out first.

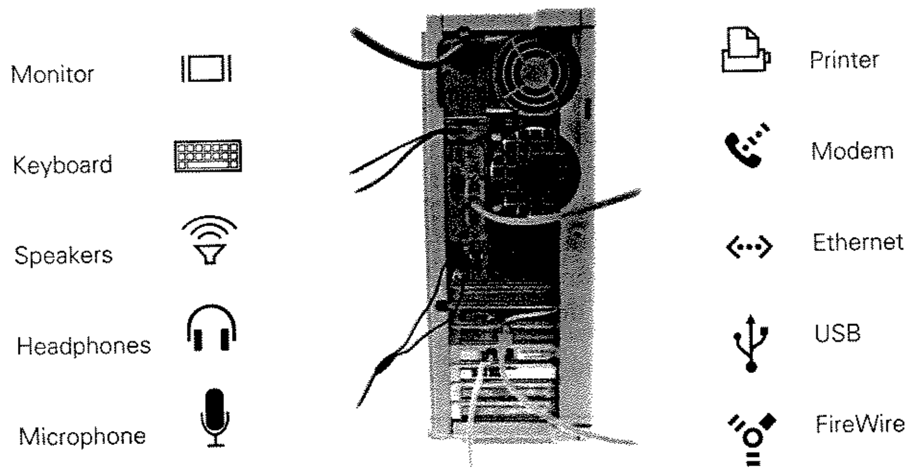


Figure 1.3 Examples of icons commonly displayed on computer cables and sockets.

Colors: RGB

Combining different amounts of three colors of light—red, green, and blue (RGB)—produces the colors you see on your computer monitor, as shown in Figure 1.4. The computer tells the monitor the right proportions of light to display with signals sent through the RGB cable. Any color can be created with some combination of intensities of these three colors. In computer applications, when we select a color from a "palette"—to change the color of text, for example—we are really telling the computer how much of these three colors of light to use.

It may seem odd that red and green combine to produce yellow, as shown in Figure 1.4. As children we learned the primary colors as yellow, red, and blue, and when mixing paints, we discovered that yellow and blue make green. The difference is that monitors mix light and in painting we mix pigment. Stage lighting

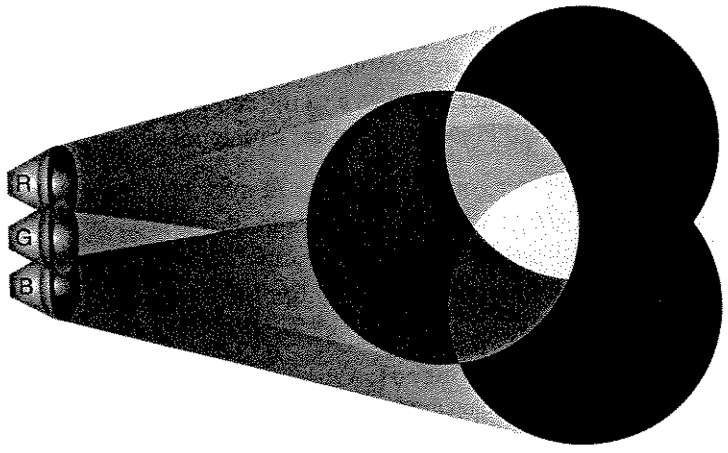
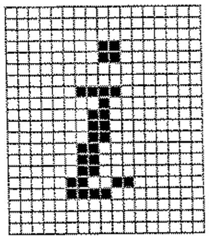


Figure 1.4 The RGB color scheme.

is another case where we mix colored light, and as we mix more colors (red, green, and blue are enough), we get closer to white. When we mix pigments, we get closer to black. Why? Because pigments absorb and reflect light. Figure 1.4 shows the color mixing of the reflected light. When white light shines on yellow pigment, it absorbs blue and reflects red and green, making yellow as you can see in Figure 1.4. Cyan pigment, a light blue, absorbs red and reflects blue and green (see Figure 1.4). Mix those two pigments together and they absorb the red and blue, reflecting only the green as we learned in kindergarten.



The monitor's screen is divided into a grid of small units called **picture elements** or **pixels**. A pixel is about the size of the dot on an *i* in 10-point type. (Check this out on your monitor.) The computer displays information on the screen by drawing each pixel in the color designated for the figure or image. For example, in text, the pixels forming the words are colored black and the surrounding pixels are colored white. The size of the grid in pixels—1024 × 768 is typical for a laptop—is important because the more pixels in each row and column, the higher the resolution of the screen image, and the smoother and crisper the result.

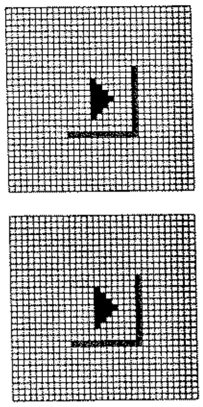
The computer must first create in its memory everything displayed on the screen, pixel by pixel. For computer-animated movies like *Toy Story*, creating images of dancing toys is extremely complex. But the appearance of reality that we see on a computer screen is much easier to generate, as we demonstrate in the following *fit*BYTE.

fit

Image Change. The size of the pixel grid displayed on the screen can be increased or decreased using the Control Panel for the Display (Windows) or Monitor (Mac).

fit

A Virtual Button. It is simple to color the screen's pixels to make a figure that looks like a believable button. On a medium-gray background, the designer colors the top and left sides of a rectangle white and the bottom and right sides black (look at the bottom image in Figure 1.5). This makes the interior of the rectangle appear to project out from the surface, because the white part is highlighted and the dark part looks like a shadow from a light source at the upper left. (If the figure doesn't look much like a button, look at it from a distance.) There is nothing special about the medium-gray/white/black combination except that it gives the lighted/shadowed effect. Other colors work, too. And, by using colors with less contrast, it is possible to give the button a different "feel"; for example, less metallic, as shown in Figure 1.5.



Two virtual buttons with different "feels."

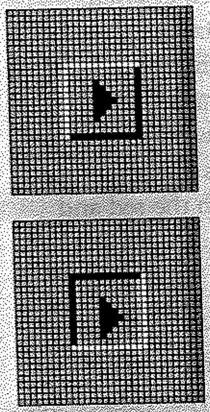


Figure 1.6 Pushing a button.

The Illusion of Button Motion

To show that the button has been pressed, the designer reverses the black and white colors and translates the icon one position down and to the right (see Figure 1.6). To **translate** a figure means to move it, unchanged, to a new position. Because our brains assume that the light source's position stays the same, the reversal of the colors changes the highlights and shadows to make the inside of the rectangle appear to be pushed in. The translation of the icon creates motion that our eyes notice, completing the illusion that the button is pressed. Notice however, that the translation of the icon down and to the right is not really a correct motion for a button "pushed into the screen," but accuracy is less important than the perception of motion.

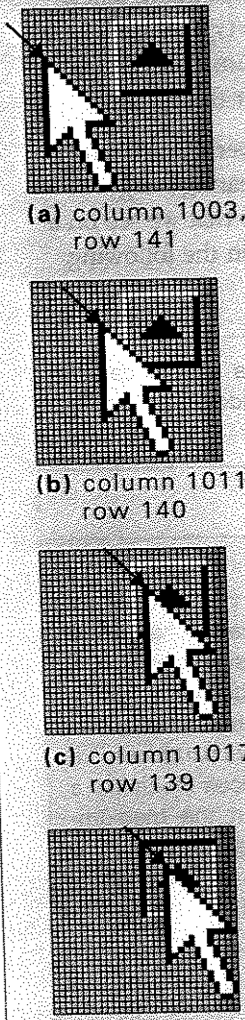
There is no real button anywhere inside the computer. The bits of the computer's memory have been set so that when they are displayed on the screen, the pixels appear to be a picture of a button. The computer can change the bits when necessary so that the next time they are displayed, the button looks as if it has been pushed in. The designer can add a "click" sound that makes the illusion even more real. But there is still no button.

Pressing a Virtual Button

If there is only a picture of a button, how can we "click" it with the mouse? That's a good question, which we can answer without becoming too technical.

We begin with the mouse pointer. The mouse pointer is a white arrow point with a black border and, like the button, must be created by the computer. When we move the mouse, the computer determines which direction it is moving and redraws the pointer translated a short distance in that direction. By repeatedly redrawing the pointer in new positions that are redisplayed rapidly, the computer produces the illusion that the pointer moves smoothly across the screen. It's the same idea as cartoon flipbooks or motion pictures: A series of still pictures, progressively different and rapidly displayed, creates the illusion of motion. The frequency of display changes is called the *refresh rate* and, like motion-picture frames, is typically 30 times per second. At that rate, the human eye sees the sequence of still frames as smooth motion.

As the mouse pointer moves across the screen, the computer keeps track of which pixel is at the point of the arrow. In Figure 1.7, the computer records the position by the column and row coordinates because the pixel grid is like graph paper. So (1003, 141) in Figure 1.7(a) means that the pixel pointed out is in the 1003rd pixel column from the left side of the screen and at the 141st pixel row from the top of the screen. With each new position, the coordinates are updated. When the mouse is clicked, as shown in Figure 1.7(d), the computer determines which button the mouse pointer is hovering over and then redraws the button to look pushed in.



(a) column 1003, row 141
(b) column 1011, row 140
(c) column 1017, row 139
(d) Click

Figure 1.7 Mouse pointer moving toward (a, b), pointing to (c), and then clicking (d) a button; the coordinates of the point of the pointer are given by their column and row positions.

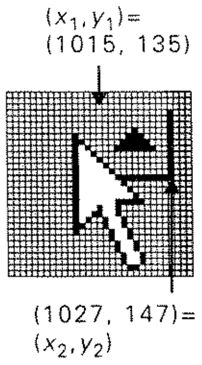


Figure 1.8

A button's location is completely determined by the positions of its upper-left and lower-right corners.

How does the computer know which button the mouse pointer is hovering over? It keeps a list of every button drawn on the screen, recording the coordinates of the button's upper-left and lower-right corners. In Figure 1.8, the button has its upper-left corner at pixel (1015, 135) and its lower-right corner at pixel (1027, 147). The two corners, call them (x_1, y_1) and (x_2, y_2), determine the position of the rectangle that defines the button: the top row of white pixels is in row y_1, the left-side column of white pixels is in column x_1, the bottom row of black pixels is in row y_2, and the right-side column of black pixels is in column x_2.

Now, if the mouse pointer's point has a row coordinate between y_1 and y_2, the pointer is somewhere between the top and bottom of the button, although it may be to the left or right of the button rather than on top of it. But if the pointer's point also has a column coordinate between x_1 and x_2, the pointer is between the left and right sides of the button; that is, it is somewhere on the button. So, for each button with coordinates (x_1, y_1) and (x_2, y_2), the computer tests whether

$x_1 <$ column coordinate of mouse pointer point $< x_2$

and

$y_1 <$ row coordinate of mouse pointer point $< y_2$

are *both* true. If so, the mouse pointer's point is over that button and the button is redrawn in the "clicked configuration." This tells the user that the command has been received and the program performs the appropriate action.

Creating a button and keeping track of the positions of the button and the mouse pointer may seem like a lot of work, but it makes using a computer easier (and more fun) for us. The metaphor of pressing a button to cause an action is so natural that software developers believe it's worth the trouble to make buttons.

This may seem like an odd question, but it's not. In casual conversation, most of us call the monitor "the computer." Technically speaking, we're usually wrong. If we are referring to a laptop or iMac, the part that actually does the computing *is* inside the same unit as the monitor, so in those cases we're right. But for component systems, the computer is not in the monitor unit, but rather in a separate box on the floor or somewhere else nearby. Calling the monitor the computer is not so much a mistake as an acknowledgment that the monitor is our interface to the computer, wherever it is.

In the component approach, the computer and most of its parts (for example, hard disk, and CD/DVD drive) are packaged together in a box called the **processor box**, though it often has a fancy marketing name (for example, minitower) that has no technical meaning. In the monolithic approach, the associated disks and drives are in the same package as the monitor; it's as if the monitor were attached to the processor box. The information in this section applies to both component and monolithic systems.

Motherboard

Inside the processor box is the **motherboard**, a printed circuit board containing most of the circuitry of a personal computer system, as shown in Figure 1.9. The name comes from the fact that smaller printed circuit boards, sometimes called **daughter boards** but more often called **cards**, are plugged into the motherboard for added functionality. A motherboard is impressive to look at with all of its fine wire patterns, colorful resistors, economy of space, and so forth. (Ask your computer dealer to show you one—it's safer than looking at the one in your computer and risking harm to it.) The motherboard is a **printed circuit** or PC board. (This use of "PC" predates "PC" meaning "personal computer" by decades.) Of the many parts on this PC board, only the microprocessor chip and the memory interest us at the moment.

Microprocessor

The **microprocessor**, found on the motherboard, is the part of a personal computer system that computes. The microprocessor is involved in every activity of

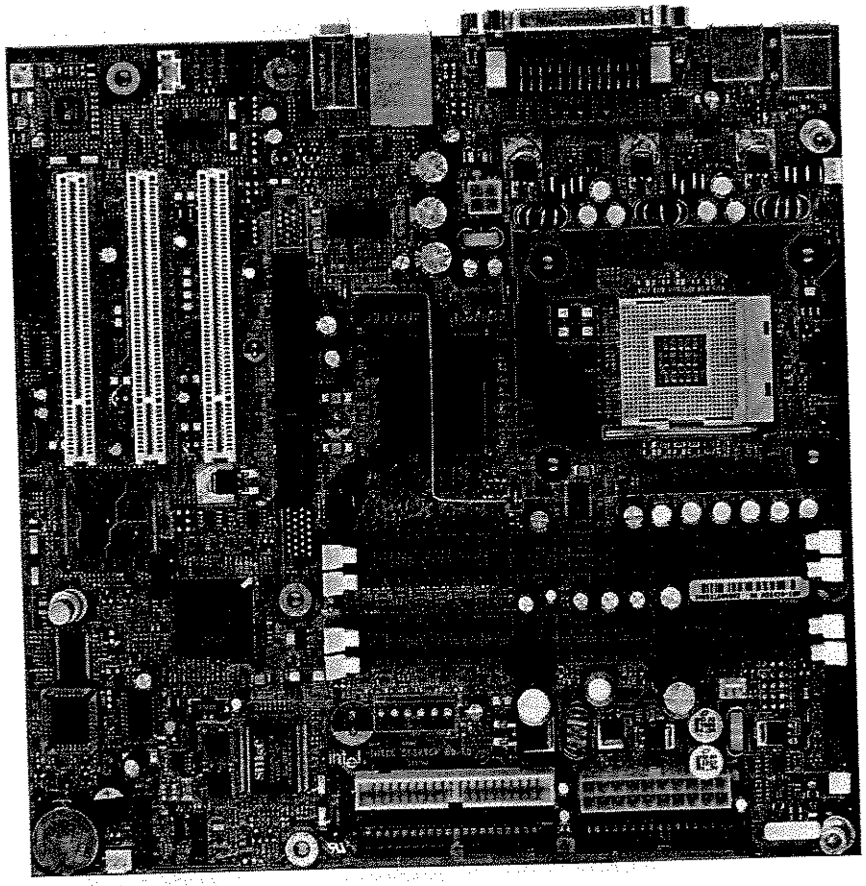


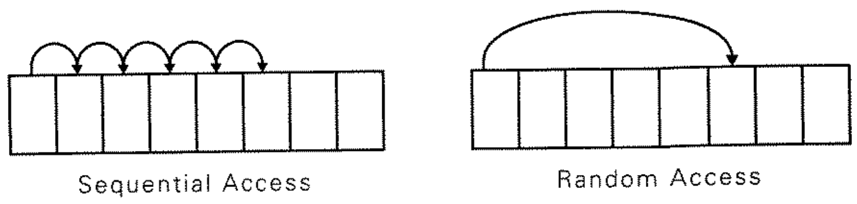
Figure 1.9 A computer motherboard.

the system, from making the mouse pointer appear to move around the screen to locating information stored on the hard disk. The microprocessor is the "smart" part of the system, so engineers often describe the other parts of a computer as "dumb." It is surprisingly easy for a computer to be "smart," as we will see. Eventually, we will even ask, "Can a computer think?"

The "micro" part of microprocessor is archaic and no longer accurate. The term "microprocessor" was adopted around 1980 when all of the circuitry for a computer first fit onto a single silicon chip. These were technically computer processors, but they were small and primitive compared to the mainframes and the "minicomputers" of the day, so they were called *micro*processors. But improvements came so quickly that in a few years microprocessors were more powerful than the largest computers of 1980. Today's microprocessors are fast, highly optimized, loaded with features, and very sophisticated. In fact, today's microprocessor chips come with *multiple* processors known as **processor cores**. Because there is nothing "micro" about today's microprocessors, we will simply use the term **processor** for the rest of this book.

The **memory** of a computer is where a program and its data are located while the program is running. For example, when you are using a word processor, the word processing program and the document being edited are stored in the computer's memory. (When they're not in memory, programs and data are stored on the hard disk; see the next section.) Computer memory is often called **RAM**, short for **random access memory**; unlike the ROM, the read-only memory mentioned earlier, RAM can be electronically read and written, making it more versatile. The basic unit of memory is a **byte**, which is described in Chapter 8. Today's personal computers have millions of bytes of RAM memory, or **megabytes** (from the Greek prefix **mega-** meaning million). See Figure 9.9 in Chapter 9 for a list of prefixes.

There are two basic ways to locate and retrieve, or *access*, information: sequential and random. Information stored sequentially is arranged in a line, so that when you want to find a specific item, you have to skip everything else stored before it, as shown in Figure 1.10. Cassette tapes, VCR tapes, and so on are examples of **sequential access**. **Random access** means that any item can be retrieved directly. Finding dictionary entries, library books, and phone numbers are examples of random access. Random access is faster than sequential access, as anyone trying to locate the last scene of a TV show on a VCR tape knows.



Sequential versus random access.

Hard Disk

The **hard disk** is not really part of the computer—technically it is a high-capacity, persistent storage peripheral device. But it is so fundamental to personal computer systems that it's helpful to think of it as a basic part. The hard disk is also referred to as the **hard drive**, and was once simply known as a **disk** before floppy disks were invented. The hard disk stores programs and data when they are not in immediate use by a computer. Disks are made from an iron compound that can be magnetized. Because the magnetism remains even when the power is off, the encoded information is still there when the power comes back on. So a disk is said to be *permanent* or *persistent* storage. A hard disk is usually located in the same box as the processor, because without access to permanent storage, the processor is crippled.

The hard disk looks like a small stack of metal washers with an arm that can sweep across and between them, as shown in Figure 1.11. The "popping" or "clicking" sound we sometimes hear is the arm moving back and forth as it accesses information on various tracks of the disk.

Saving from RAM to Hard Disk. Successful computer users save their work regularly. For example, when writing a term paper using a word processor, run the *Save* command every half hour or perhaps after completing a few pages or a section. Saving copies the information, that is, your term paper, from the RAM memory to the hard disk memory. This action is important because of the differences between the two types of memory.

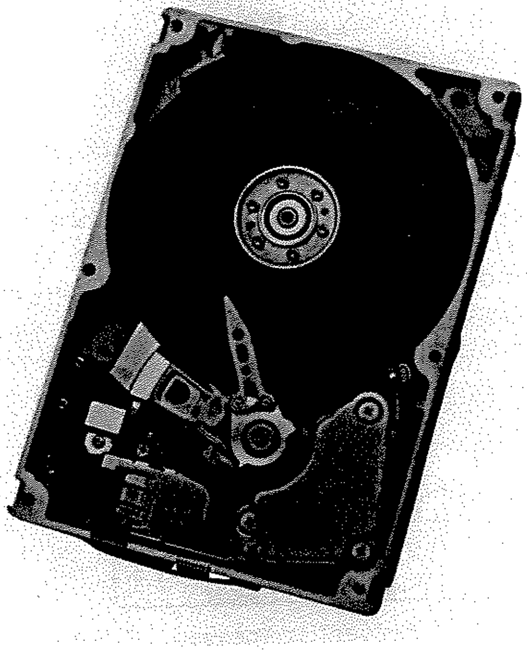


Figure 1.11 A hard disk.

Recall that hard disk memory is made from a magnetic material that "remembers," even when the power is off. Today's RAM memory, however, is made from integrated circuit (IC) technology, informally called microchips. One property of IC memory is that it is volatile, meaning that the information is lost when the power is turned off. That is, when IC memory loses power, it "forgets." This difference wouldn't matter if computers didn't fail. But they do.

When a computer crashes—that is, when it no longer works correctly or it stops running entirely—it must be restarted, a process called rebooting. If the computer's not working correctly, the *Restart* command will run. The first step in restarting is to erase the memory, causing the information stored in the RAM to be lost. If it's not working at all, then the power must be cycled—turned off, and then turned on again—also causing the information stored in the RAM to be lost, because IC memory is volatile. Either way, anything that was in the RAM, such as the latest version of your term paper, will be lost. Only the copy that is on the hard disk, which is the copy from the last time it was saved, is available when the computer "comes up." All the work since the last save is lost.

Crashes are not common, but they happen. Frequently saving your file limits the amount of work you will have to redo if your computer crashes.

The term hardware predates computers by centuries. Originally it meant metal items used in construction, like hinges and nails. The word software did not exist until computers were invented.

Software is a collective term for computer programs. The name contrasts with hardware, of course, but what does it mean for software to be "soft"? When a computer function is implemented in software, the computer performs the operation by following instructions. A program that figures out your income taxes is an example of software. If a computer function is implemented in **hardware**, the computer performs the operations directly with wires and transistors. (We say it is **hard-wired**.) The multiplication operation is an example of hardware-implemented functions.

The difference between "hard" and "soft" is like the difference between an innate ability, such as coughing, and a learned ability, such as reading. Innate abilities are "built in" biologically, and they're impossible to change, like hardware. Learned abilities are easily changed and expanded, like software. Typically, only the most primitive operations like multiplication are implemented in hardware; everything else is software. (Computers don't *learn* to perform soft operations, of course, the way we learn to read. Rather, they are simply given the instructions and told to follow them.)

*fit*BYTE

The Hard Reality. The difference between hardware and software was dramatically illustrated in 1994 when a bug was discovered in the hard-wired divide operation of Intel's Pentium processor, an approximately $200 chip. Though the wrong answers were rare and tiny, the error had to be fixed. If divide had been implemented in software, as had been the usual approach in computing's early years, Intel could have sent everyone a simple patch for $1 or $2. But hardware cannot be modified, so the Pentium chips had to be recalled at a total cost estimated at about $500 million.

Algorithms and Programs

An **algorithm** is a precise and systematic method for solving a problem. Another term for a systematic method is a **process**. Some familiar algorithms are arithmetic operations like addition, subtraction, multiplication, and division; the process for sending a greeting card; and searching for a phone number. The method used for determining when a mouse pointer hovers over a button is an algorithm. Because an algorithm's instructions are written down for some other agent (a person or a computer) to follow, precision is important.

We learn some algorithms, like arithmetic. Sometimes we figure out algorithms on our own, like finding phone numbers. In Chapter 10, we introduce **algorithmic thinking**, the act of thinking up algorithms. Writing out the steps of an algorithm is called **programming**. **Programs** are simply algorithms written in a specific programming language for a specific set of conditions.

When we ask a computer to do something for us, we ask it to **run** a program. This is literally what we ask when we click on the icon for an application like Internet Explorer. We are saying, "Run the program from the Microsoft company to browse the Internet." *Run* is a term that has been used since the invention of computers, but *execute* is a slightly better term because it emphasizes an important property of computing.

Execute

A computer **executes** a program when it *performs* instructions. The word *execute* means to follow a set of orders exactly as they are written. Computer pioneers used the word *orders* for what we now call instructions. Orders tell the computer to act in a specific way. When orders are given, the faithful agent is *not* supposed to think. "Following instructions literally" is what computers do when they run programs; it is that aspect that makes "execute" a slightly better term than "run."

In addition to run and execute, **interpret** is also a correct term for following a program's instructions, as explained in Chapter 9.

Boot

Finally, the term **booting** means to start a computer and **rebooting** means to restart it. Because booting most often happens after a catastrophic error or a crash, you might guess that the term is motivated by frustration—we want to kick the computer like a football. Actually, the term *booting* comes from *bootstrap*. Computers were originally started by an operator who entered a few instructions into the computer's empty memory using console push buttons. Those instructions told the computer to read in a few more instructions—a very simple operating system—from punch cards. This operating system could then read in the instructions of the real operating system from magnetic tape, similar to a VCR tape. Finally, the computer was able to start doing useful work. This incremental process was called **bootstrapping**, from the phrase "pulling yourself up by your bootstraps," because the computer basically started itself. Today the instructions to start a computer are stored on a microchip called the boot ROM.

The Words for Ideas

Although understanding the physical parts of IT—monitors, motherboards, and memory—seems very important, we are not too concerned with them here. Instead, we will focus on *concept* words, such as those discussed in this section.

Abstract

One of the most important "idea" words used in this book is the verb *abstract*. It has several meanings. In British mysteries, *to abstract* means *to remove*, as in *to steal*: "The thief abstracted the pearl necklace while the jeweler looked at the diamond ring." In information technology, *to abstract* also means to remove, but the thing being removed is not physical. The thing being removed is an idea or a process, and it is extracted from some form of information.

To **abstract** is to remove the basic concept, idea, or process from a situation. The removed concept is usually expressed in another, more succinct and usually more general form, called an **abstraction**.

We are familiar with abstraction in this sense. Parables and fables, which teach lessons in the form of stories, require us to abstract the essential point of the story. When we are told about a fox that can't reach a bunch of grapes and so calls them sour, we abstract the idea from the story: When people fail to reach a goal, they often decide they didn't want to reach that goal in the first place.

Notice two key points here. First, many but not all of the details of the story are irrelevant to the concept. When abstracting, we must decide which details of the story are relevant and which are irrelevant. The "grapes" and the "fox" are unimportant, but "failure" is important. Being able to tell the difference between important and unimportant details is essential to understanding the point of a story, and to abstraction in general. Second, the idea—the abstraction—has meaning beyond the story. The point of repeating the parable, of course, is to convey an idea that applies to many situations.

"Generalize"

A process similar to abstraction is to recognize the common idea in two or more situations. Recognizing how different situations have something basic in common is why we create parables, rules, and so on.

To **generalize** is to express an idea, concept, or process that applies to many situations. The statement summing up that idea is called a **generalization**.

For example, most of us notice that twisting a faucet handle left turns water on and twisting it right turns it off. Not always—some water taps have only a single "joy stick" handle, and others have horizontal bars that pull forward. But since it's true most of the time, we generalize that "on" is to the left and "off" is to the right. Perhaps we also notice that twisting lids, caps, screws, and nuts to the left usually loosens them, and turning them to the right usually tightens them. Again, we generalize that left means loosen and right means tighten. We probably also generalize that both situations are examples of the same thing! A generalization of generalizations.

Noticing patterns and generalizing about them is a very valuable habit. Although generalizations do not always apply, recognizing them gives us a way to begin in a new, but similar, situation.

"Operationally Attuned"

Another term related to extracting concepts and processes refers to being aware of how a gadget works. To be **operationally attuned** is to apply what we know about how a device or system works to simplify its use.

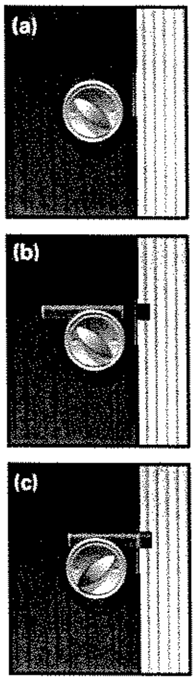
For example, we previously generalized that with few exceptions, all caps, lids, screws, and nuts tighten by turning right and loosen by turning left. We might know this intuitively, but knowing it *explicitly* makes us operationally attuned. Knowing this fact as a rule means that when a lid or nut is stuck, we can twist it very hard, certain that we are forcing it to loosen rather than tighten. Being operationally attuned makes us more effective.

The term operationally attuned is introduced here to emphasize that thinking about how IT works makes it simpler to use. We don't expect to be experts on all of the technology—few are. But by asking ourselves, "How does this work?" and using what we learn by thinking about the answer, we will be more successful at applying IT. Our Fluency study focuses on learning enough to answer many of the "How does this work?" questions well enough to succeed.

"Mnemonic"

Mnemonic is a rather unusual term that we use in IT and in other fields as well. The silent *m* implies that it's a word with an unusual past.

A mnemonic (ni·MĂ·nik) is an aid for remembering something. The reminder can take many forms, such as pronounceable words or phrases. We remember the five *fit*BYTE



Tuning In. In our daily lives, we use hundreds of devices, systems, and processes. For some, like a car ignition, we quickly learn which way to turn the key because it only turns in one direction. We don't think about how it works. Using it becomes a habit. Other gadgets, however, have more leeway, and for them it helps to be attuned to their operation. One example is a deadbolt lock, which moves a metal bar from the door to the doorframe to lock the door. Thinking about how the lock works tells us whether the door is locked or not. Referring to Figure 1.12(a), notice which way the knob is turned. By visualizing the internal works of the lock, we can imagine that the top of the knob is attached to the bar. When the knob is pointing left, the bar must be pulled back—that is, unlocked. When the knob is pointing right, the bar is extended, so the door is locked. We may not know how the lock really works, but explaining its operation in our own terms means that we can see at a distance whether the door is locked or unlocked. It might save us from getting up from the sofa to check if the door is locked.

Figure 1.12 **Deadbolt lock. (a) The external view. (b) Internal components, unlocked. (c) Internal components, locked. Thinking about how the deadbolt works allows us to see at a glance whether the door is locked or not.**

Great Lakes with the acronym HOMES—Huron, Ontario, Michigan, Erie, and Superior. Earlier in this chapter, we mentioned PILPOF—plug in last, pull out first And when Pluto was downgraded from planetary status, we had to change our mnemonic for the planets: My very educated mother just served us nachos.

There are many IT details that we need to know only occasionally, like when to connect power. They're not worth memorizing, but they're inconvenient to look up. So, if we can think of a mnemonic that helps us remember the details when we need them, using technology is simpler.

Analytical Thinking

Using the right terms makes learning IT simpler. Becoming more analytical is an equally valuable habit to acquire. When we say that the world record in the mile run has improved or that computer performance has improved, we are making very weak statements. They simply assert that things have changed over time for the better. But has the change been infinitesimally small or gigantic? How does the change compare to other changes? Since we can easily find information on the Internet, we can compare an earlier measure of performance with a recent one. Thinking analytically is essential to becoming Fluent in IT, but it is also useful in our other studies, our careers, and our lives.

Mile Runs

When Moroccan runner Hicham El Guerrouj broke the world record on July 7, 1999, the news reports trumpeted that he "smashed," "eclipsed," and "shattered" the world record set six years earlier by Noureddine Morceli of Algeria (see Figure 1.13). El Guerrouj had run a mile in an astonishing 3 minutes, 43.13 seconds, an impressive 1.26 seconds faster than Morceli. The descriptions were not hyperbole. People around the world truly marveled at El Guerrouj's accomplishment, even though 1.26 seconds seems like an insignificant difference.

To put El Guerrouj's run into perspective, notice that 45 years had passed since Englishman Roger Bannister attracted world attention as the first man in recorded history to run a mile in less than 4 minutes (see Figure 1.13). His time was 3:59.4. In 45 years, the world's best runners improved the time for the mile by an astonishing 16.27 seconds. (Notice that El Guerrouj's 1.26 seconds was a significant part of that.) As a rate, 16.27 seconds represents an improvement from 15.038 miles per hour to 16.134 miles per hour, or just over 7 percent. Given that Bannister's world-class time was the starting point, an improvement in human performance of that size is truly something to admire.

Comparing to 20-Year-Olds

How do these world champions compare to average people? Most healthy people in their early 20s—the age group of the world record setters—can run a mile in 7.5 minutes. This number was chosen because it covers the ability of a majority of the people in the age range, and is approximately twice the time El Guerrouj needed. To say El Guerrouj is twice as fast as an average person is to say he is faster by a factor of 2. (The factor relating two numbers is found by dividing one by the other; for example 7.5 / 3.75 = 2.)

This factor-of-2 difference is a rough rule for the performance gap between an average person and a world champion for most physical strength activities such as

Figure 1.13 The runners Hicham El Guerrouj (left) and Roger Bannister (right).

running, swimming, jumping, and pole vaulting. The factor-of-2 rule tells us that no matter how hard most people try at physical activities, their performance can improve by roughly twice. Of course, most of us can only dream of achieving even part of that factor-of-2 potential. Nevertheless, the factor-of-2 rule is an important benchmark.

When we compared world champions, we said there was a 7 percent improvement and that El Guerrouj's speed was about a factor-of-2 times faster than the speed of an average person. There is a difference between expressing improvement as a *percentage* and expressing improvement as a *factor*. We find a **factor of improvement** by dividing the new rate by the old rate. So, to find El Guerrouj's improvement over Bannister's, we divide their rates (16.134 / 15.038) to get 1.07. Percentages are a closely related computation found by dividing the *amount of change* by the old rate (16.134 – 15.038) / 15.038 = 0.07 and multiplying the result by 100. The added complexity of percentage is potentially confusing, so we use the simpler factor-of-improvement method. El Guerrouj was a factor-of-1.07 times faster than Bannister and about a factor-of-2 times faster than an average person.

As another example of analytical thinking, let's compare computer speeds. The UNIVAC I, the first commercial computer, unveiled in 1951 (and current when Bannister set his record), operated at a rate of nearly 100,000 addition operations (adds) per second. By comparison, a typical PC today—say, the portable IBM ThinkPad—can perform a billion additions per second or so. This factor-of-10,000 improvement over UNIVAC I (1,000,000,000/100,000) is truly remarkable. But, consider this—the ThinkPad is no record setter. It's the sort of computer a college student can afford to buy. Engineering workstations can easily do several billion adds per second, boosting the factor even higher. And an Intel computer called ASCI Red, built for Sandia National Laboratory, held the world record for computer speed in 1999, when El Guerrouj set his record. ASCI Red ran at an astonishing 2.1 trillion floating-point adds per second. (Floating-point adds are decimal arithmetic operations that are more complex than the additions used to measure the speed of the UNIVAC I.) Compared to the UNIVAC I, ASCI Red is a factor of 21 million times faster!

fit

Faster Still. ASCI Red was the fastest of its day, but its day has passed. Several computers have eclipsed its performance, and better designs continue to emerge. For the latest speed tests, see `www.netlib.org/benchmark/top500.html`.

Perhaps nothing else in human experience has improved so dramatically. In roughly the same time period that human performance improved by a factor of

1.07 as measured by the mile run, computer performance improved by a factor of 21,000,000. Can we comprehend such a huge factor of improvement, or even the raw speed of ASCI Red?

try it **Computing the Factor of Improvement.** Flyer 1, the aircraft Orville and Wilbur Wright flew at Kitty Hawk, North Carolina, traveled so slowly (10 mph) that the brother who wasn't piloting could run alongside as it flew just off the ground. The SR-71 Blackbird, probably the world's fastest plane, flies at 2,200 mph, three times the speed of sound.

What is the factor of improvement between Blackbird and Flyer 1?

Speed of Blackbird = 2,200 mph
Speed of Flyer 1 = 10 mph
Factor of improvement = 2200 / 10 = 220

The Blackbird is a factor-of-220 faster than Flyer 1.

*fit*BYTE

Think About It. Most of us can appreciate the 7 percent improvement of El Guerrouj's run over Bannister's, and probably the factor-of-2 improvement in average versus world champion performance. Those we can imagine. But factors of improvement in the thousands or millions are beyond our comprehension. Notice that if El Guerrouj had improved on Bannister by a factor of 21,000,000, he'd have run the mile in 11.4 microseconds. That's 11.4 millionths of a second. What does that mean?

- Human visual perception is so slow that El Guerrouj could run 3,000 miles at that rate before anyone would even notice he had moved.
- The sound would still be "inside" the starting gun 11.4 microseconds after the trigger was pulled.
- Light travels only twice as fast.

Both the raw power of today's computers and their improvement over the last half-century are almost beyond our comprehension.

Benefits of Analytical Thinking

To summarize, we have made our understanding of recent speed improvements crisper by applying simple analysis: Rather than accepting the statement that the mile run and computers have improved, we learned the facts given as two measurements of performance. But once we had the data, we did not leave it as two separate observations: 100,000 additions in 1951 versus 2.1 trillion additions in 1999. Instead, we analyzed their relationship by figuring the factor of improvement: 2,100,000,000,000/100,000. ASCI Red is faster by a factor of 21 million.

This analysis let us compare the improvement to other advancements, and to put them all into perspective. The mile run, improved by an apparently small factor of 1.07 times in 45 years, is still very impressive when we recall that champions are only about a factor-of-2 better than average people. Computer performance has improved by unimaginable amounts. Although our original statement, "the mile run and computers have improved," is correct, our analysis helps us to be much more expressive and precise. Analytical thinking helps us understand more clearly the world of information technology and the physical world in which we live.

Our only remaining task is to define the first acronym mentioned in the chapter: **WYSIWYG**. Remember that it stands for "what you see is what you get." To understand the term, recall that the computer creates the virtual world we see on our monitors. The representation the computer uses to keep track of the things on the screen is very different from the picture it shows us. For example, the text you are reading was stored in the computer as one very long line of letters, numbers, punctuation, and special characters, but it is displayed to me as a nicely formatted page like the one you are reading. The computer processes this representation—the long sequence of letters—to create the nicely formatted page. Original text editing software couldn't do that, so users had to work with the long sequence. If you wanted to make a change to the document, you had to imagine what it would look like when printed.

Eventually, text editing systems were programmed to show the user the page as it would appear when printed. Changing the text became much easier. This property was described as "what you see (when editing) is what you get (when it's printed)" or WYSIWYG. Text editors with the WYSIWYG property became known as **word processors**.

In this chapter we focused on learning IT terms in context. We learned to:

- Know and use the right word because as we learn words, we learn ideas; knowing the right words helps us to communicate.
- Ask questions to review basic and familiar terms, such as monitor, screen saver, RAM, and software.
- Understand some new terms, such as sequential access, volatile, and motherboard.
- Consider a brief list of "idea" words, such as abstract and generalize.
- Save our work regularly.
- Think analytically by looking at improvements in the mile run and computer speed.

We're not done, however. All of the chapters introduce new terms when new ideas are discussed. Learning and remembering these key terms will help you learn and remember the ideas. Key terms appear in the glossary at the end of the book.

Check the glossary when the meaning of a term slips your mind. In addition to the glossary in this book, there are several good online glossaries. It's a good idea to find one with your Web browser and bookmark it—that is, save its URL. We will discuss URLs in Chapter 3. Meanwhile, check the glossary to learn what the acronym URL means.

Review Questions

Multiple Choice

1. Computer monitors are different from TVs because
   a. monitors are bit-mapped whereas TVs are not
   b. TVs are interactive whereas monitors are not
   c. monitors are CRTs whereas TVs are not
   d. more than one of the above

2. Screen savers
   a. are useful because they prevent burn-in
   b. save energy
   c. can be turned off with a key press or a mouse click
   d. all of the above

3. The display for a laptop is most likely a
   a. TV
   b. RGB display
   c. LCD
   d. CRT display

4. Mice and keyboards do not have power cords because
   a. they are not electrical
   b. the power and the signal wires are in one cable
   c. they run on batteries
   d. none of the above

5. The last cable you plug in should be the
   a. monitor cable
   b. keyboard cable
   c. printer cable
   d. power cable

6. RGB stands for
   a. red, green, black
   b. red, gray, blue
   c. rust, black, brown
   d. red, green, blue

7. A combination of red light and green light produces
   a. blue
   b. purple
   c. yellow
   d. brown

8. A typical monitor
   a. has over a million pixels
   b. has exactly a million pixels
   c. displays pixels in only one color
   d. none of the above

9. How many pixels make up each button in Figure 1.8 (page 11)?
   a. 19 × 19
   b. 304
   c. 1024 × 768
   d. none of the above

10. How is the process for clicking a check box similar to clicking a button?
    a. The tip of the arrow must be inside the *x*, *y* coordinates that make up the check box.
    b. The user must click the mouse button.
    c. The configuration of the check box must change from unchecked to checked or vice versa.
    d. all of the above

1. The ______ is involved in every activity of the computer system.
2. Knowing ______ is important to understanding technology and being understood when talking about it.
3. A ______ saves a CRT monitor from "burn in" when it is not in use.
4. The last cable you should plug in should be the ______.
5. There are ______ pixels on a typical laptop?
6. The number of times a second that images on the screen are redrawn is called the ______.
7. The ______ is the active point of the mouse pointer.
8. A specified result sought through the use of a precise and systematic method is a(n) ______.
9. ______ is the proper term used when a computer performs the instructions in a program.
10. The process of starting a computer is called ______.
11. Gleaning the central idea or concept from a situation is called ______.
12. A device that helps you remember a fact or concept is a ______.
13. A WYSIWYG text editor is called a ______.

14. On the computer, programs and information are stored on the ______.
15. The formulation of an idea, concept, or process that can be applied in many situations is called a ______.

Exercises

1. Here are some of the acronyms found in this chapter. Next to each, write its name and its meaning.

   SCSI ______
   IT ______
   CRT ______
   LCD ______
   RGB ______
   CD ______
   RAM ______
   ROM ______
   PC ______
   IC ______
2. Create a list of mnemonics that you know and their meanings.
3. Find the details of the computer system you are using. If you do not have the manual, use an ad from a newspaper, flyer, or a Web site. Write down the specifications for your system, paying close attention to the terms and specs that are unfamiliar. Make a note to learn more about these in further chapters.
4. It took Magellan's expedition three years to circumnavigate the globe. The space shuttle makes an orbit in about 90 minutes. Calculate the factor of improvement.
5. Ellery Clark of the United States won the long jump in the 1896 Olympiad in Athens, Greece with a jump of 20 feet 9 3/4 inches. The current world record is held by Mike Powell of the United States, who jumped a distance of 8.95 meters. What is the factor of improvement? Be sure to convert between feet and meters.
6. Ray Harroun won the first Indianapolis 500 in 1911 with a speed of 74.59 miles per hour. The 2004 race was won by Buddy Rice with a speed of 138.518 mph. What is the factor of improvement?
7. In 2005, Yelena Isinbayeva set the world pole vault record at 5.01 meters. William Hoyt set the record in 1896 with a vault of 3.30 meters. What is the factor of improvement that she made over the men's record of 1896? What is the percentage increase?