Chapter 1

Log Me In, UNIX!

In This Chapter

- Turning your computer on and getting its attention
- Persuading your computer to let you use it
- Using usernames, passwords, and all that
- Logging out when you finish

If you read the exciting introduction to this book, you know that we make some Foolish Assumptions about you, the reader. Among other things, we foolishly assume that someone else has installed and set up UNIX for you so that all you have to do is turn your computer on and tell UNIX that you're there, or that a computer running UNIX is somewhere on the network that you have access to.

If you don't have UNIX already set up on a computer, the best thing you can do for yourself is find a local UNIX guru or system administrator who is willing to get you up and running. Unless you really know what you're doing, installing and setting up UNIX can be painful, frustrating, and time-consuming. We recommend that you find something more enjoyable to do, such as cleaning out the grease trap under your kitchen sink or performing urgent home surgery on yourself. (You can learn how to administer a UNIX system with some patience and perseverance, but explaining how is way beyond the scope of this book because each version of UNIX has its own procedures.)

Turning Your Computer On and Off

If you think that turning your computer on and off is easy, you may be wrong. Because UNIX runs on so many almost-but-not-quite-compatible computers — all of which work somewhat differently — you first must figure out which kind of UNIX computer you have before you can turn it on.

If a train stops at a train station, what happens at a workstation?

A *workstation* is a computer with a big screen, a mouse, and a keyboard. You may say, "I have a PC with a big screen, a mouse, and a keyboard. Is it really a workstation?" Although UNIX zealots get into long arguments over this question, for our purposes, we say that it is. Most current UNIX systems are workstations.

Turning on a workstation is easy enough: You reach around the back and turn on the switch. Cryptic things that appear on-screen tell you that UNIX is going through the long and not-at-all-interesting process of starting up. Starting up can take anywhere from ten seconds to ten minutes, depending

What you were hoping we wouldn't tell you: The difference between a PC and a workstation

First, you have to understand that this isn't a technical question — it's a theological question. Back in the olden days (about 1980), telling the difference was easy. A workstation had a large graphical screen — at least, large by the standards of those days — a megabyte of memory, a fast processor chip, a network connection, and it cost about $10,000. A PC had a lousy little screen, 64K of memory, a slow processor chip, a floppy or two, and it cost more like $4,000.

These days, your typical $800 PC has a nice screen (much nicer than what the workstation used to have), hundreds of megabytes of memory, a fast Pentium processor, a big disk, speakers, and a network connection. That's much better than what people used to call a workstation. Does that make a PC a workstation? Oh, no. Modern workstations have even better screens, buckets of memory, a turbocharged processor chip — you get the idea. What's the difference?

Maybe it's the software that people use: Most workstations are designed to run UNIX (or, in a few cases, proprietary systems similar in power to UNIX), whereas PCs run Windows or Macintosh software. Wait — some perfectly good versions of UNIX run on PC hardware, and Windows 2000/NT runs on many boxes that everyone agrees are workstations, and the latest version of Mac OS is UNIX underneath. Now what? You can get into esoteric arguments about the speed of the connection between the guts of the computer on one hand and the disks, screens, and networks on the other hand and argue that workstations have faster connections than PCs, but some examples don't fit there, either.

As far as we can tell, if a computer is *designed* to run Windows or the Mac OS, it's a PC. If it's *designed* to run UNIX, it's a workstation. If this distinction sounds feeble and arbitrary to you, you understand perfectly. Here at *UNIX For Dummies* Central, we have a couple of large PCs running UNIX (which makes them look, to our eyes, just like workstations) and a couple of other, smaller ones running Windows. Works fine for us.

on the version of UNIX, number of disks, phase of the moon, and so on. Sooner or later, UNIX demands that you log in. To find out how, skip to the section "Logging In: U(NIX) Can Call Me Al," later in this chapter.

Turning *off* a workstation is a more difficult problem. Workstations are jealous of their prerogatives and *do* punish you if you don't turn them off in exactly the right way. Their favorite punishment is to throw away all the files related to whatever you were just working on. The exact procedure varies from one model of workstation to another, so you have to ask a local guru for advice. Typically, you enter a command along these lines:

```
shutdown +3
```

This command tells the workstation to shut down (in three minutes, in this example). With some versions of UNIX, that command is too easy. The version we use most often uses this command:


```
halt
```

If you use Linux, type this command to shut down the system right away:

```
shutdown now
```

The workstation then takes awhile to put a program to bed or whatever else it does to make it feel important, because it knows that you're waiting there, tapping your feet. Eventually, the workstation tells you that it's finished. At that point, turn it off right away, before it gets any more smart ideas.

An approved method for avoiding the hassle of remembering how to turn off your workstation is never to shut off your computer (although you can turn off the *monitor*). That's what we do.

A dumb terminal

The traditional way to hook up to a UNIX system is with what's known (sneeringly) as a dumb terminal. Nobody makes dumb terminals any more, but Windows PCs have a natural ability to play dumb, so they're commonly pressed into duty as terminals. You run a terminal emulator program on a PC, and suddenly the mild-mannered PC turns into a super UNIX terminal. (Truthfully, it's more the other way around: You make a perfectly good PC that can run *Doom* and other business productivity-type applications act like a dumb terminal that can't do much of anything on its own.)

When you finish with UNIX, you leave the terminal emulator, usually by pressing Ctrl+X or some equally arcane combination of keys. (Consult your local guru: No standardization exists.) Like Cinderella at the stroke of midnight, the terminal-emulating PC turns back into a real PC. To turn it off, you wait for the PC's disks to stop running (carefully scrutinize the front panel until all the little red or green lights go out) and then reach around and turn off the big red switch. If you don't wait for the lights to go out, you're liable to lose some files.

If you have a network installed, which these days has become so cheap that nearly everyone does, your PC running Windows probably has a network connection to your UNIX system. Windows 95/98/Me/NT/2K/XP, and the Mac OS (the Macintosh operating system) have the network stuff built in.

If you do have a network connection, you can use programs called `telnet`, `ssh`, or `putty` (described in Chapter 16) to connect to your UNIX system. After one of them is running and connected to your UNIX system, within your program's window you get a faithful re-creation of a 1970s dumb terminal and you can proceed to log in.

After you connect, you use it to communicate with the computer that is running UNIX. If the terminal is wired directly to the computer, UNIX asks you to log in before you can do anything else (see the section "Hey, UNIX! I Want to Log In," later in this chapter). If not, you may have to perform some additional steps to call the computer or otherwise connect to it.

An *X terminal* is similar to an extremely stripped-down workstation that can run only one program — the one that makes X Windows work. (See Chapter 4 to find out what X Windows are — or don't. It's all the same to us.) Turning an X terminal on and off is pretty much like turning a regular dumb terminal on and off. Because the X terminal doesn't run programs, turning it off doesn't cause the horrible problems that turning off a workstation can cause. You can get X software for Windows to make a Windows PC act like an X terminal, too. If you have such a PC, ask the person who set it up how to start it and stop it.

Hey, UNIX! I Want to Log In

Whether you use a remote PC or a workstation, you have to get the attention of UNIX. You can tell when you have its attention because it demands that you identify yourself by logging in. If you use a workstation, whenever UNIX finishes loading itself, it is immediately ready for you to log in (skip ahead to the section "Logging In: U(NIX) Can Call Me Al"). You terminal users (X or otherwise), however, may not be so lucky.

Direct access

If you're lucky, your keyboard and screen are attached directly to the main computer, either because the main computer is the only one and you're sitting at it, or someone's rigged up a remote PC to log in directly. If so, it displays a friendly invitation to start working, something like this:

```
ttyS034 login:
```

Well, maybe the invitation isn't that friendly. By the way, the `ttyS034` is the name UNIX gives to your terminal. Why doesn't it use something easier to remember, like Fred or Muffy? Beats us!

This catchy phrase tells you that you have UNIX's attention and that it is all ears (metaphorically speaking) and waiting for you to log in. You can skip the next section and go directly to "Logging In: U(NIX) Can Call Me Al."

If your UNIX system displays a terminal name, make a note of it. You don't care what your terminal's name is, but, if something gets screwed up and you have to ask an expert for help, we can promise you that the first thing the guru will ask is, "What's your terminal name?" If you don't know, the guru may make a variety of nerd-type disparaging comments. But, if you can say, "A-OK, Roger. That's terminal `tty125`," your guru will assume that you are a with-it kind of user and may even try to help you. (Even if her name isn't Roger.)

Yo, UNIX! — not-so-direct access

If you're connecting over the Internet or another network, either find a local network expert to tell you how to connect, or see Chapter 16 for some suggestions.

If you're using a PC with a modem, you probably have to tell the modem to call the UNIX system. Although all terminal emulators have a way to make the call with two or three keystrokes, all these ways are different, of course. (Are you surprised?) You have to ask your local guru for info.

After your terminal is attached to the computer, turned on, and otherwise completely ready to do some work, UNIX, as often as not, doesn't admit that you're there. It says nothing and seems to ignore you. In this way, UNIX resembles a recalcitrant child — firm but kind discipline is needed here.

The most common ways to get UNIX's attention are

- Press the Return or Enter key. (We call it the Enter key in this book, if you don't mind.) Try it two or three times if it doesn't work the first time. If you're feeling grouchy, try it 20 or 30 times and use a catchy cha-cha or conga rhythm. It doesn't hurt anything and is an excellent way to relieve stress.
- Try other attention-getting keystrokes. Ctrl+C (hold down the Ctrl key, sometimes labeled Control, and press C) is a good one. So is Ctrl+Z. Repeat to taste.
- If you're attached to UNIX through a modem, you may have to do some speed matching (described in a minute): Press the Break key a few times. If you're using a terminal emulator, the Break key may be disguised as Alt+B or some other hard-to-find combination. Ask your guru.

Two modems can talk to each other in about 17,000 different ways, and they have easy-to-remember names, such as B212, V.32, and V.32bis. (*Bis* is French for "and a half." Really.) After you call the UNIX system's modem with your modem, the two modems know perfectly well which way they're communicating, although UNIX sometimes doesn't know. Every modem made since about 1983 announces the method it's using when it makes the connection. Because the corresponding piece of UNIX code dates from about 1975, though, UNIX ignores the modem's announcement and guesses, probably incorrectly, at what's being used.

If you see something like `~xxx~~r.!"` on-screen, you need to try *speed matching*. Every time you press Break (or the terminal emulator's version of Break), UNIX makes a different guess at the way its modem is working. If UNIX guesses correctly, you see the login prompt; if UNIX guesses incorrectly, you see another bunch of `~xxx~~~@(r)!"` or you see nothing. If UNIX guesses incorrectly, press Break again. If you overshoot and keep Breaking past your matched speed, keep going, and it'll come around again.

After awhile, you learn exactly how many Returns, Enters, Breaks, and whatnots your terminal needs in order to get UNIX's attention. It becomes second nature to type them, and you don't even notice what a nerd you look like while you do it. You have no way around that last part, unfortunately.

Logging In: U (NIX) Can Call Me Al

Every UNIX user has a username and password. Your system administrator assigns you a username and a password. Although you can and should change your password from time to time, you're stuck with your username.

Before you can start work, you must prove your bona fides by logging in; that is, by typing your username and password. How hard can typing two words be? Really, now. The problem is this: Because of a peculiarity of human brain wiring, you will find that you can't enter your username and password without making a typing mistake. It doesn't matter whether your username is `al` — you will type `Al`, `la`, `a;L`, and every other possible combination.

UNIX always considers upper- and lowercase letters to be different: If your username (sometimes also called your *login name*) is `egbert`, you must type it exactly that way. Don't type Egbert, EGBERT, or anything else. Yes, we know that your name is Egbert and not egbert, but your computer doesn't know that. UNIX usernames almost always are written entirely in lowercase. Pretend that you're a disciple of e. e. cummings.

When you type your username and password and make a mistake, you may be tempted to press Backspace to clear your mistake. If only life were that easy. Guess how you clear typing errors when you type your username and password? You press the # key, of course! (We're sure that it made sense in 1975.) Some — but not all — versions of UNIX have changed so that you can use Backspace or Delete; you may have to experiment. If you want UNIX to ignore everything you typed, press @, unless your version of UNIX has changed the command key to Ctrl+U (for *untype,* presumably — doubleplusungood). So, Egbert (as you typed your username), you may have typed something like this:

```
ttyS034 login: Eg##egberq#t
```

Finish entering your username by pressing Enter or Return.

After you type your username, UNIX asks you to enter your password, which you type the same way and end by pressing Enter (or Return, but we call it Enter). Because your password is secret, it doesn't appear on-screen as you type it. How can you tell whether you typed it correctly? You can't! If UNIX agrees that you typed your username and password acceptably, it displays a variety of uninteresting legal notices and a message from your system administrator (usually `delete some files, the disk is full`) and passes you on to the shell, which you find out about in Chapter 2.

If UNIX did not like either your username or your password, UNIX says `Login incorrect` and tells you to start over with your username.

In the interest of security, UNIX asks you for a password even if you type your username wrong. This arrangement confuses the bad guys — but not nearly as much as it confuses regular users. So, if UNIX rejects your password even though you're sure that you typed it correctly, maybe you typed your username incorrectly.

Password Smarts

Like every UNIX user, you should have a password. You can get along without a password only under these circumstances:

- You keep the computer in a locked, windowless room to which you have the only key, and it's not connected to any network.
- You don't mind whether unruly 14-year-olds borrow your account and randomly insert dirty knock-knock jokes in the report you're supposed to give to your boss tomorrow.

The choice of your password deserves some thought. You want something easy for you to remember but difficult for other people to guess. Here are some bad choices for passwords: single letters or digits, your name, the name of your spouse or significant other, your kid's name, your cat's name, or anything fewer than eight characters. (Bad guys can try every possible seven-letter password in less than a day.)

Good choices include such things as your college roommate's name misspelled and backward. Throw in a digit or two or some punctuation, and capitalize a few letters to add confusion, so that you end up with something like `yeLLas12`. Another good idea is to use a pair of words, like `fat;Head`.

You can change your password whenever you're logged in, by using the `passwd` program. It asks you to enter your old password to prove that you're still who you are when you logged in (computers are notoriously skeptical). Then the `passwd` program asks you to enter your new password twice, to make sure that you type it, if not correctly, at least consistently. None of the three passwords you type appears on-screen, of course. We show you how to run the `passwd` program in Chapter 2.

Some system administrators do something called *password aging;* this strategy makes you change your password every once in awhile. Some administrators put rules in the `passwd` program that try to enforce which passwords are permissible, and some even assign passwords chosen randomly. The latter idea is terrible because the only way you can remember a password you didn't choose is to write it on a sticky note and stick it on your terminal, which defeats the purpose of having passwords.

In any event, be sure that no one other than you knows your password. Change your password whenever you think that someone else may know it. Because UNIX stores passwords in a scrambled form, even the system administrator can't find out what yours is. If you forget your password, the administrator can give you a new one, but she can't tell you what your old one was.

If you really want to be paranoid about passwords, don't use one that appears in any dictionary. Some system breakers may decide to use the UNIX password-encryption program to encrypt every last word in a dictionary and then compare each of the encrypted words to your password. It's another thing to keep you awake at night.

Ciao, UNIX!

Logging out is easy — at least compared to logging in. You usually can type **logout**. Depending on which shell you're using (a wart we worry about in Chapter 2), you may have to type **exit** instead. In many cases, you can press Ctrl+D to log out.

You know that you have logged out successfully because UNIX either invites the next sucker to log in, hangs up the phone, or, if you're connected by `telnet` or `ssh`, disconnects from your program.