

# Take Control

## *of* Permissions in Leopard

*by* Brian Tanaka

### Table of Contents (1.1)

Read Me First .....	2
Introduction .....	4
Permissions Quick Start .....	5
Problems and Solutions .....	6
About Permissions .....	7
The Anatomy of Permissions .....	10
Choose a Method of Setting Permissions .....	15
Set Permissions Using the Info Window .....	17
Set Permissions Using Third-Party Tools .....	19
Use Access Control Lists .....	24
Understand Default Permissions .....	29
Work with User Names, UIDs, and GIDs .....	45
Understand Ignore Ownership .....	55
Repair Permissions with Disk Utility .....	58
Learn Advanced Unix Techniques .....	61
Learn More .....	73
Appendix A: Fixes for Common Problems .....	74
Appendix B: Converting To Octal .....	80
Appendix C: Use the man Command .....	83
About This Book .....	84



## READ ME FIRST

Welcome to *Take Control of Permissions in Leopard*, version 1.1, published in September 2008 by TidBITS Publishing Inc. Although this book has a different title, it is effectively the second edition of *Take Control of Permissions in Mac OS X*. This book helps you control the often-perplexing world of permissions in Mac OS X 10.5 Leopard. It explains how permissions work, how to resolve common problems, and how to best control access to your files in a variety of situations. This book was written by Brian Tanaka and edited by Tonya Engst (with help from Sandro Menzel).

Copyright © 2008, Brian Tanaka. All rights reserved.

The price of this ebook is \$10. If you want to share it with a friend, please do so as you would a physical book. [Click here](#) to give your friend a discount coupon. [Discounted classroom copies](#) are also available.

### Updates

We may offer free minor updates to this book. To read new information or access any new versions of this ebook's PDF, click the Check for Updates link on the [cover](#). On the resulting Web page, you can also sign up to be notified about updates to the PDF via email. If you own only the print version of the book, contact us at [tc-comments@tidbits.com](mailto:tc-comments@tidbits.com) to obtain a PDF update.

### Basics

In reading this book, you may get stuck if you are unfamiliar with the way I describe working with the Mac. Please note the following:

- **Path syntax:** I occasionally use a *path* to show the location of a file or folder. For example, Mac OS X stores most utilities, such as Terminal, in the Utilities folder. The path to Terminal is: **`/Applications/Utilities/Terminal`**.

The slash at the beginning of a path tells you to start from the root level of the disk. Some paths begin with ~ (tilde), which is a shortcut for any user's home directory. For example, if a person with the user name **joe** wants to install fonts that only he can access, he would install them in the **`~/Library/Fonts`** folder, which is another way of writing **`/Users/joe/Library/Fonts`**.

- **Menus:** When I describe choosing a command from a menu in the menu bar, I use an abbreviated description. For example, the abbreviated description for the menu command that opens an Info window on a file or folder from the Finder is “File > Get Info.”

## What's New

This book is a complete, Leopard-specific update to *Take Control of Permissions in Mac OS X*. Whereas that book covered all versions of Mac OS X up to and including Mac OS X 10.4 Tiger, this book focuses tightly on Mac OS X 10.5 Leopard.

In Leopard, Apple has made a number of changes that relate to permissions. Most significantly, Apple has eliminated NetInfo. Though most users never used NetInfo Manager or the command line `ni` tools, if you are a power user who used NetInfo to work with permissions, you'll need to learn some new techniques. For instance, see [Change an account's UID](#) (p. 48).

Other changes in Leopard include:

- Leopard has changed the options in the Accounts pane in System Preferences. As in previous versions of Mac OS X, you can create Administrator and Standard accounts, but the process for creating a Managed account by applying parental controls is now more streamlined and has more options. Entirely new in the Accounts pane in Leopard is the capability to set up Sharing Only accounts to be used by people on the network accessing files on your Mac. Another new option is a Guest account, intended for people who want to sit at your Mac's keyboard and use its software.

I talk more about these accounts and their associated permissions in [About Permissions](#) (p. 7).

- You can now manipulate Unix groups from System Preferences. See [Manage Groups](#) (p. 51).
- ACLs are on by default in Leopard, and they are used more extensively. I cover this in [Use Access Control Lists](#) (p. 24).
- The NSUmask technique no longer works in Leopard. See [Set Permissions for New Items](#) (p. 32). (Info relating to this fact is new in version 1.1 of this ebook.)

## INTRODUCTION

Even if you don't know a thing about permissions, if you're using Mac OS X 10.5 Leopard, you're using them right now. Every file and folder on your computer carries permissions from the moment it's created until the moment it's deleted. Because permissions are literally everywhere on your computer and because they control who can access what, it's tremendously advantageous to understand them. You'll have better control over your Mac, and you'll be able to share items and access shared items with greater ease.

Problems arising from improperly set permissions are common and can be frustrating: Sharing files among users on one computer can be problematic if you don't understand permissions, and sharing items on a network raises yet another set of potential problems.

In this book I teach you how to prevent and fix permissions problems with ease and much more. You'll learn how to interpret and manipulate permissions with the Info window in the Finder, Disk Utility, third-party tools, and Unix commands. You'll learn about accounts and groups, and how permissions control them; how default permissions work; how to repair permissions; and how to ignore permissions on an attached volume.

Equipped with this expertise, you'll be able to handle permissions problems when sharing files locally or across networks, booting from multiple volumes, exchanging files with other users, running FTP and Web servers, and much more.

## PERMISSIONS QUICK START

The first sections of this book teach the basics of permissions and how to set them. The remaining sections explore more advanced techniques and concepts that help you solve problems.

### Learn about permissions:

- Find out what permissions are, and why you need them. See [About Permissions](#) (p. 7).
- Permissions are composed of simple interrelated parts. Discover how they work together. See [The Anatomy of Permissions](#) (p. 10).
- If you already know a bit about permissions and want an overview of what's new in Leopard, read [What's New](#) (p. 3).

### Set permissions:

- There's more than one way to set permissions. See [Choose a Method of Setting Permissions](#) (p. 15).
- Learn to [Set Permissions Using the Info Window](#) (p. 17) and to [Set Permissions Using Third-Party Tools](#) (p. 19). And, if you need more fine-grained tools for controlling permissions, read [Use Access Control Lists](#) (p. 24).
- To solve a permissions-related problem, see [Problems and Solutions](#) (next page), for a quick index to helpful info.
- If you enjoy working in Unix or need the fine-grained control that Unix can provide, [Learn Advanced Unix Techniques](#) (p. 61).

### Delve deeper into permissions:

- Discover how your Mac assigns default permissions in [Understand Default Permissions](#) (p. 29), and increase your permissions IQ by reading [Work with User Names, UIDs, and GIDs](#) (p. 45).
- Learn to use two important Mac OS X features in [Understand Ignore Ownership](#) (p. 55), and [Repair Permissions with Disk Utility](#) (p. 58).
- Unix commands empower you to do things you can't do from the graphical user interface, which you'll see when you [Learn Advanced Unix Techniques](#) (p. 61).

## PROBLEMS AND SOLUTIONS

I discuss a variety of common problems in [Appendix A: Fixes For Common Problems](#) (p. 74), but you will find help with solving other problems throughout the book. Use the links below to navigate to info that will help you with specific problems:

- I'm having trouble with [The Shared Folder](#) (p. 35).
- The Info window doesn't show permissions settings I know exist. See [Set Permissions Using Third-Party Tools](#) (p. 19) and [Learn Advanced Unix Techniques](#) (p. 61).
- I don't own my own files! See [Work with User Names, UIDs, and GIDs](#) (p. 45).
- I am concerned about the privacy of files and folders that I created and saved in my user account, and I want to make sure that others on the computer cannot access them in any way. Read [The Case of the Promiscuous Folder](#) (p. 34).
- When do I use Ignore Ownership on This Volume? See [Understand Ignore Ownership](#) (p. 55).
- Everyone tells me to use Repair Permissions but I don't understand what it does. Learn the real story in [Repair Permissions with Disk Utility](#) (p. 58).
- I can see why understanding octal is useful when setting permissions, but I can't seem to get my head around it. See [Appendix B: Converting To Octal](#) (p. 80).
- When I copy or create items, I can't predict what the permissions will be. It's driving me batty! Find help in [Understand Default Permissions](#) (p. 29).

## ABOUT PERMISSIONS

Like all Unix-based operating systems, Mac OS X is designed to make it easy for multiple people, termed *users*, to share the same computer. Each user has a user account (or more than one in some situations).

Having your own personal user account is useful and convenient for a number of reasons. It enables you, for instance, to customize your account settings and preferences without affecting other users. It also enables you to store and organize your personal files and folders in your *home folder*—a special folder reserved for your exclusive use.

**NOTE** I discuss user accounts in this book enough to help you understand permissions. If you want more info, I highly recommend reading [Take Control of Users & Accounts in Leopard](#).

In addition to an account for each user, Mac OS X computers have other types of accounts organized in a system based on Unix account management.

Traditionally, in Unix there is an all-powerful account called root. Root has absolute authority and can, among other things, override permissions on items and change item ownership without restriction. All other accounts fall into two categories: *user accounts*, which are accounts used by actual humans, and *system accounts*, which are not associated with specific users and exist to perform tasks requiring special authority but not the absolute authority of root.

User accounts, on Unix systems, can be granted the power to run individual programs as root, most commonly via the *sudo* facility.

In Mac OS X, this highly flexible system of root account, system, and user accounts, and the granting of arbitrary administrative power, is nicely presented in a simplified form. Specifically, it offers several types of accounts:

- **Root:** The root account is authorized to do anything.
- **Administrator:** Administrator accounts can perform certain administrative tasks that require authorization beyond that of standard accounts, such as: changing global system preferences; creating, managing, and deleting user accounts; and changing

permissions on items. This is the default account type for the first user account created on a newly installed Mac OS X computer.

- **Standard:** Standard accounts have limited authority on the computer, mostly restricted to activities that affect their accounts alone. This is the default account type for ordinary users.
- **Managed:** This account has been around in previous versions of Mac OS X, though each version has brought a different twist to how the account is configured and what limits could be applied to it. In Leopard, these accounts are called “Managed with Parental Controls,” and administrators can restrict Internet content deemed inappropriate, set when and for how long the account can be used, and control access to applications, email, and iChat. If you are concerned about what your child is doing with the computer, this is a great choice.
- **Sharing Only:** Sharing Only accounts provide remote file-sharing access, but a Sharing Only user can’t change the settings on the computer or log in via the normal account login window. Instead, a Sharing Only user works at a different computer and connects to the Sharing Only account only to access or drop off files.
- **Group:** Groups aren’t actually accounts in the sense that the others are. Rather, a group is a *collection* of accounts. Allowing folks to create groups from the same interface where they create regular accounts is Apple’s attempt to provide a convenient and easy way to make Unix groups. I discuss Unix groups and why they’re useful in [Manage Groups](#).
- **Guest account:** If the guest account is enabled, visitors who do not have a permanent account on the Mac can log in as Guest without providing a password. When the user logs in, a temporary user account (with a home folder) is created, and when the user logs out, the corresponding home folder is deleted.

Every item on your computer belongs to, or is *owned by*, an account. (For the sake of brevity, I use *item* as a general term meaning “file, folder, or disk” except where I need to be more specific.) For instance, when you create a new file, that new file is owned by your user account. If another user, logged into her own account, creates a file, that file will be owned by *her* user account. In addition to being

owned by a particular user account, every item on your computer carries with it a set of *permissions* that control which user accounts can access it and what kind of access they have.

That, in a nutshell, is the purpose of permissions: they control who can do what to which files, folders, and disks.

Permissions, combined with accounts and ownership, are exceedingly useful for a number of reasons, including:

- **Security:** Permissions are a critical component in the security model of all Unix-based operating systems, including Mac OS X. For instance, if a standard user account is compromised in some way by a malicious attacker, the attacker should not be able to alter critical system files because the permissions on those files disallow tampering by non-administrator accounts.
- **Privacy:** Permissions on your private files and folders can be set so that user accounts other than your own have limited access or no access at all.
- **Controlled sharing:** Because permissions are powerful and flexible, you can exercise a significant degree of control over which users can access items you choose to share and what those users can do with the shared items.
- **System integrity:** Permissions prevent non-administrator accounts from damaging the system by altering important system items, and they prevent users from tampering with other users' items.

## THE ANATOMY OF PERMISSIONS

As I mentioned in the previous section, every item on your computer is owned by an account and carries a set of permissions. These permissions control the access that each of three classes—owner, group, and other—has to an item.

Here's a quick explanation of what I mean by owner, group, and other:

- **Owner:** The *owner* is the user account that owns an item, such as a file, folder, or disk. Every item is owned by an account. (Traditionally in Unix, this is known as the user class, and Unix commands abbreviate it with a u.)
- **Group:** In addition to being owned by a user account, every item is also owned by a *group*. A group is a set of user accounts conceptually clumped together so permissions can apply to its members collectively. Mac OS X provides a number of default groups, and you can create additional groups.
- **Other:** Everyone else! *Other* refers to all user accounts on the system other than the owner and members of the group. You will see this type referred to as “others” (in the Finder’s Info window) and “world” (by other tools).

Permissions for an item say whether owner, group, and other have these three permissions, which **Table 1**, next page, further explains:

- **Read:** View the contents of the item.
- **Write:** Change the item.
- **Execute:** Execute the item.

**NOTE** **Table 1** covers permissions from the Unix point of view. The Finder describes them using somewhat different terminology. See [Set Permissions Using the Info Window](#).

## CHOOSE A METHOD OF SETTING PERMISSIONS

You can set permissions in Mac OS X in more than one way. Namely, you can use:

- The Info window in the Mac OS X Finder
- Third-party tools
- Unix commands

Each method has relative advantages and disadvantages, as you can see in **Table 2**.

<b>Table 2: Methods of Setting Permissions</b>		
<b>Method</b>	<b>Advantages</b>	<b>Disadvantages</b>
Info window	Easier and more intuitive than Unix commands.	Limited capabilities. Not as powerful or flexible as Unix commands.
Third-party tools	More features than the Info window.	Not present by default. May require additional expense.
Unix commands	More powerful and flexible than the Info window. May be faster if you are familiar with Unix.	Less intuitive than the Info window.

Deciding which method to use in any given situation is up to you, and depends on the demands of the situation at hand.

Personally, I prefer to set permissions from the Terminal using Unix commands. I find that it's the fastest, easiest, and most flexible way to handle simple and complex permissions. For simple situations, the amount of time and effort I expend is roughly the same or less than I would expend setting permissions via a graphical user interface such as the Info window or a third-party tool. And, for more complicated tasks, I save a huge amount of time and effort not only because the command line Unix tools themselves are tremendously effective, but also because I can combine those tools with the power of the Unix shell and other Unix tools.

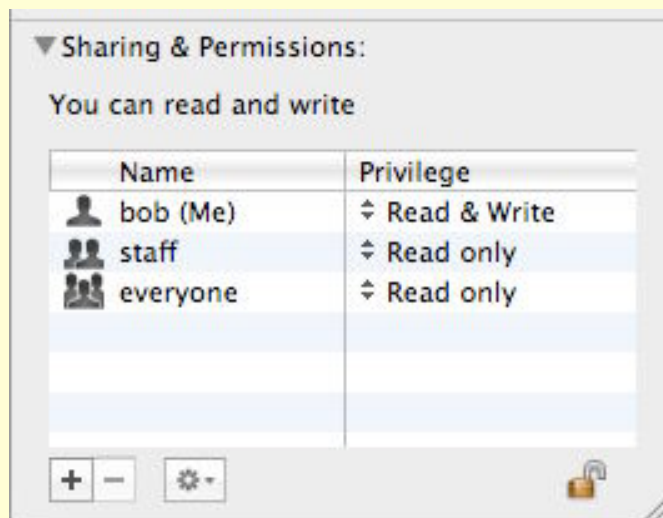
## SET PERMISSIONS USING THE INFO WINDOW

For simple tasks relating to setting permissions, I recommend that you use the Finder's Info window. The Info window does not provide the degree of control that Unix commands do. You can't, for instance, manipulate individual Unix permissions. However, setting permissions via the Info window is quite easy and is perfectly adequate for most day-to-day situations.

Set permissions with the Info window by following these steps:

1. Working in the Finder, select an item.
2. Choose File > Get Info (Command-I) to open the Info window.
3. In the Info window, you may need to click the Sharing & Permissions triangle to reveal more detail, as shown in **Figure 1**.

**FIGURE 1**



Ownership and permissions for a typical file as viewed in the Info window.

4. Click the lock icon (🔒) to change permissions. You'll be prompted to authenticate, which you should do.
5. Use the Privilege pop-up menus to set permissions for your account, group, and everyone. (If you are wondering how these options map to their Unix equivalents, see **Table 3**, next page.)
6. Close the Info window.

## SET PERMISSIONS USING THIRD-PARTY TOOLS

The Info window is adequate for most day-to-day, basic permissions editing. However, it has a limited feature set and can't handle many permissions-related tasks. Fortunately, several third-party tools perform these missing tasks. I cover three popular ones here: FileXaminer (below), [Super Get Info](#), and [XRay](#).

All three tools do what the Info window does and more, and provide finer control over permissions. The interfaces and features—at least as far as permissions are concerned—are very similar, and they all provide good help documentation and contextual menus. Because they are so similar, I find myself hard pressed to strongly recommend one over the others, but **Table 4** contains some factors to consider.

Table 4: Suggested Third-Party, Graphical Permission Tools		
Software	Cost	Notes/URL
FileXaminer (see below)	\$10	Most features overall. Fewer configurable preferences. Shorter demo period (7 days versus 14 days). <a href="http://www.gideonsoftworks.com/filexaminer.html">http://www.gideonsoftworks.com/filexaminer.html</a>
<a href="#">Super Get Info</a>	\$20	Uncluttered, intuitive interface. Most expensive. Does not support set UID bit, set GID bit, or sticky bit. <a href="http://www.barebones.com/products/super/index.shtml">http://www.barebones.com/products/super/index.shtml</a>
<a href="#">XRay/XRay II</a>	\$10	Nice interface. Highly rated by users. Plug-in architecture (allows development of new features by other developers) According to the developer, XRay 1.1 isn't fully Leopard-compatible, but I've included it because a beta of XRay II, which is slated to support Leopard, may be available in 2008. <a href="http://www.brockerhoff.net/xray/">http://www.brockerhoff.net/xray/</a>

### FileXaminer

FileXaminer, a \$10 utility from Gideon Softworks, has several advantages over the Info window (<http://www.gideonsoftworks.com/filexaminer.html>). A partial list of its permissions-related features includes:

- **Batch edit:** You can change permissions (and other settings) on multiple items simultaneously.

## USE ACCESS CONTROL LISTS

In Mac OS X 10.4 Tiger, Mac OS X began to offer access control lists (ACLs). *ACLs* supplement Unix permissions and provide more detailed control over permissions. They also provide better interoperability with Samba and Windows.

ACLs provide advantages over simply using traditional Unix permissions, but the average Mac user running a non-server version of Mac OS X may never need them. Unless you are supporting multiple users with complex sharing requirements, you may want to skim or skip this section.

### Advantages of ACLs

Though Unix permissions are adequate for most situations, ACLs provide some clear advantages, including:

- ACLs offer more kinds of permissions than read, write, and execute. For example, ACLs provide a delete permission.
- ACLs are tremendously valuable if you have multiple individuals or groups working on collaborative projects, because they work around the “single user, single group” problem, which is perhaps the biggest limitation of traditional Unix permissions.

For example, if you have a file that’s owned by the user **bob** and the group **developers**, you cannot grant permissions for a user who is not **bob** or in the group **developers**. However, with ACLs, you can specify that user **frida** can, for instance, write to the file even though she is not in the **developers** group. Likewise, you can specify permissions for another group such as, for example, one called **designers**.

- ACLs coexist nicely with traditional Unix permissions. ACLs are evaluated, and, if none apply, the Unix permissions on the item apply.

## UNDERSTAND DEFAULT PERMISSIONS

Because all items have permissions, Mac OS X must decide which permissions to apply when an item is created, copied, moved, extracted from an archive, and so on. Understanding these default behaviors is invaluable. You'll know what to expect, and you'll identify problems quickly and fix them by changing permissions to suit your needs.

### Permissions on New Items

Various factors influence the default, initial permissions on new items. At the most fundamental level, a global *bit mask* turns off certain permission bits by default. A bit mask is a pattern of zeros and ones that influences the pattern of another series of zeros and ones, as I show in the examples ahead.

The global bit mask in Mac OS X is `000 010 010`.

**NOTE** In Unix documentation, you will see this global bit mask expressed as `022 octal`. Octal is the base-8 numbering system. Counting in octal is simple: 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 20, 21, 22, and so on. Octal skips 8 and 9, as well as 18 and 19.

The base permissions—the original permissions used in conjunction with the global bit mask to calculate default permissions—are:

- For folders: `111 111 111 (rwxrwxrwx)`
- For files: `110 110 110 (rw-rw-rw-)`

When the base permission bits “drop through” the mask, any **1** that encounters another **1** in the mask is cleared, that is, changed to **0**.

## WORK WITH USER NAMES, UIDS, AND GIDS

Since Mac OS X displays the user name of the account that owns a folder or file, it's easy to assume that that account with that *user name* owns the item. In reality, it's not quite so simple. The owner is actually determined by a number called the *UID*—the *user identification number*, not by the user name. In addition the group is, in fact, determined by the *GID*—the *group identification number*.

### How They Work

A famous quotation from *Romeo and Juliet* reads: “What’s in a name? That which we call a rose by any other name would smell as sweet.” Deep down in Unix and Mac OS X, names—or, rather, user names—are just as incidental.

Every account has a user name (Mac OS X calls it the *short name*), a numeric UID, and a numeric GID. User names, such as **rose**, are strictly for the convenience of we humans who use the accounts. After all, it's easier, if your name is Rose, to remember **rose** than some number, such as 501.

When it comes to file ownership, however, the operating system doesn't care if your user name is **rose**. It cares only about the UID. So, when it assigns or determines ownership, it uses the UID, not the user name, and it uses a numeric GID to identify an item's group ownership.

To reveal your account's UID and primary GID, type **id** in a Terminal window (and then press Return). The output of the **id** command looks like this:

```
uid=502(btataka) gid=20(staff) groups=20(staff),  
80(admin)
```

As you can see, my account's UID is 502. My account's primary group is **staff**, which is GID 20. My account also belongs to group 80: **admin**. (An account can belong to multiple groups. I discuss groups in [Manage Groups](#).)

To view the same information for another user, use the **id** command with the user name. For instance, if I want to know the UID for **bob**, I type: **id bob**.

## UNDERSTAND IGNORE OWNERSHIP

You can tell Mac OS X to ignore ownership of all files and folders on a non-boot volume. Since permissions depend on ownership, when you ignore ownership, permissions are much less effective. This can be helpful in a variety of situations, such as sharing files with a group of people or dealing with ownership problems resulting from having multiple boot volumes.

Although ignoring permissions makes them less effective, it does not make them totally irrelevant, because, despite the name of the option, ignoring ownership doesn't actually ignore ownership. Instead, each item on the volume takes on a special UID and GID that tell the operating system to act as if any account that accesses the item is the item's owner. (Ownership of items created before ignore permissions was enabled reverts to the original owners when the ignore ownership option is disabled.)

For example, if you ignore ownership on a volume, the operating system will report that whichever account you're using owns all items on that volume. If a *different* account looks at items on the volume, the operating system will report that *that* account is the owner.

It's like setting a box with your name written on it on a table and leaving the room. When the next person walks in, he sees *his* name written on the box. If another person walks in and stands beside the first person, she will see *her* name. To whom does the box belong? Anyone who looks at it! Magic!

However, when an account attempts to work with the item, the permissions relevant to the item's owner apply, as usual. If, for instance, the permissions allow the owner read permission only, then that item is read-only.

Still, for the vast majority of typical items on a typical volume, this effectively means that all accounts have free access to all items since most items have liberal permissions for the owner. The net effect then, is that ownership is ignored, and permissions aren't a hindrance. Some people refer to this phenomena as *floating permissions*, although that's an inaccurate, unofficial term: the ownership floats, not the permissions.

## REPAIR PERMISSIONS WITH DISK UTILITY

Mac OS X's Disk Utility can verify and repair permissions on some files and folders. To be accurate, it doesn't actually *repair* permissions. Rather, it simply *resets* permissions according to guidelines I discuss later in this section.

Further, to say Disk Utility repairs permissions implies that permissions can "go bad" or become corrupted over time and therefore need repairing because they're broken. This is not true. Permissions stay the way they are set until someone or something comes along and sets them another way.

So, if permissions stay the way they are, what's setting them long after they're initially set and causing all the fuss? In other words, why do we need the Repair Permissions feature at all? A number of things can be at fault:

- **Installers:** Some installers change permissions on existing files and folders as a necessary part of the installation process, but fail to return them to their proper settings. To guard against slovenly installers, try running Verify or Repair Permissions after installing software, especially third-party software.
- **User error:** Mistakes can lead to problems requiring use of Disk Utility. A simple mistake with the `chmod` command, for instance, can play havoc with permissions, requiring correction by Disk Utility.

Much of the software you install on your Mac is installed from packages. When a package is installed, the installer creates a Bill of Materials (.bom) file in that package's receipts file. Installation receipts are stored in `/Library/Receipts/`. The Bill of Materials file lists the files installed by the package and the initial permissions for those files.

**NOTE** To list all files installed by a package, use the `lsbom` command.

## LEARN ADVANCED UNIX TECHNIQUES

I saved this more advanced portion for the end of the book because many readers want to dive in and use tools that are familiar to them; therefore, I first discussed tools offering a graphical user interface. This section is for those who want to take control of the command line and harness its power. Because Unix is a broad and deep subject, this section is meant more as a practical overview than an exhaustive exploration.

These advanced techniques are not required for taking control of permissions, and we've covered quite a bit of ground already, so don't worry if you've hit information overload and want to skip this section.

### NOTE WHY USE UNIX?

Unix has a reputation for being hard to learn. Though this is often true, it's also paradoxically easy to use. Once you clear the initial learning hurdle, the power and flexibility at your fingertips makes working with your computer easier.

**TIP** Tiger and Leopard users have a choice of two free (almost identically-named) Dashboard widgets that translate Unix permissions between symbolic and absolute mode:

- Unix Permissions Calculator: <http://vocaro.com/trevor/software/widgets/>
- Unix Permission Calculator: <http://liepins.org/dashboard/>

The permissions-related commands are:

- **ls:** List files; covered in [List Folder Contents with ls](#).
- **chmod:** Change file mode; see [Change Permissions with chmod](#).
- **chown:** Change owner; flip ahead to [Change Owners with chown](#).
- **chgrp:** Change group owner; see [Change Groups with chgrp](#).
- **chflags:** Change flag; covered briefly in [Change File Flags with chflags](#).

## LEARN MORE

I hope this book has whetted your appetite for more knowledge about permissions in particular and UNIX in general. The resources listed below are excellent places to continue your journey of discovery.

### Ebooks:

- *Take Control of Users & Accounts in Leopard*  
by Kirk McElhearn, published by TidBITS Publishing Inc.
- *Take Control of Sharing Files in Leopard*  
by Glenn Fleishman, published by TidBITS Publishing Inc.

### Printed books:

- *UNIX for Mac OS X: Visual QuickPro Guide*  
by Matisse Enzer, published by Peachpit Press
- *UNIX in a Nutshell*  
by Arnold Robbins, published by O'Reilly Media, Inc.
- *The Mac OS X Command Line: Unix Under the Hood*  
by Kirk McElhearn, published by Sybex

### Web sites:

- **ONLamp.com:** An Introduction to Unix Permissions  
[http://www.onlamp.com/pub/a/bsd/2000/09/06/FreeBSD\\_Basics.html](http://www.onlamp.com/pub/a/bsd/2000/09/06/FreeBSD_Basics.html)
- **Wikipedia:** File system permissions  
[http://en.wikipedia.org/wiki/File\\_system\\_permissions](http://en.wikipedia.org/wiki/File_system_permissions)
- **Apple:** Troubleshooting permissions issues in Mac OS X  
<http://docs.info.apple.com/article.html?artnum=106712>
- **Apple:** Security Overview: Permissions  
[http://developer.apple.com/documentation/Security/Conceptual/Security\\_Overview/Concepts/chapter\\_3\\_section\\_9.html](http://developer.apple.com/documentation/Security/Conceptual/Security_Overview/Concepts/chapter_3_section_9.html)

### Usenet Newsgroups:

- **Comp.unix:** (via Google)  
<http://groups.google.com/group/comp.unix/topics>
- **Comp.unix.admin:** (via Google)  
<http://groups.google.com/group/comp.unix.admin/topics>

## APPENDIX A: FIXES FOR COMMON PROBLEMS

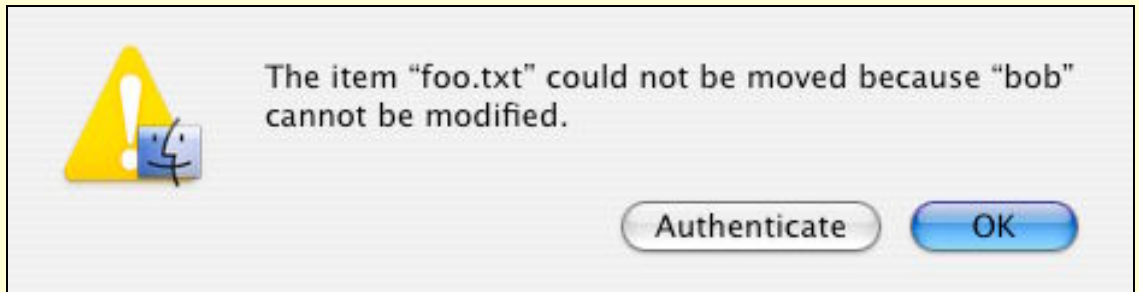
This section explores and solves a variety of permission problems.

### Can't Give a File to Another User

Perhaps the simplest and most common permissions problem arises when you need to give a file to another user on the same computer. Fortunately, this isn't a problem in the sense that something is wrong. Rather, the existing permissions, which are right, are not working the way you expect them to work.

Let's say you want to give a file called `foo.txt` to user account **bob**. If you try to simply drag the file from your home folder to **bob's** home folder, you'll receive an error because your user account does not have permission to write to that folder (**Figure 7**).

**FIGURE 7**



Trying to copy a file to another user's home folder gives this error.

You cannot write to **bob's** home folder because, by default, each user's home folder is writeable only by that user. In this way, Mac OS X protects files and folders from other users. In this case, only **bob** can write to his home folder, and you are denied. If you view the Info window for **bob's** home folder, you will see in the Sharing & Permissions area, that "You can read only." Likewise, if **bob** were to try to drop a file into my home folder, he would receive the "could not be moved" error dialog.

There are many ways to get the file to **bob**. Two of the best are to use **bob's** Drop Box or to override the permissions by authenticating as an administrator. Let's explore both of these methods.

## APPENDIX B: CONVERTING TO OCTAL

Converting from something like `rw-r-x-r-x` to 755 octal can be a bit confusing at first. Once you understand how it works, you'll find it's quite easy. As an example, let's convert `rw-r--r--` to octal. Split the permissions into three groups of three:

`rw- r-- r--`

Substitute 4 for r, 2 for w, 1 for x, and 0 for dash to get:

420 400 400

Now add the numbers in each of the three groups: 6 4 4

And smush the digits together: 644

Et voilà! `rw-r--r--` is 644 octal.

Now I know some of you are thinking, "Great. But what about the set UID, set GID, and sticky bits?" Good question. The set UID, set GID and sticky bits have the same "4, 2, and 1" values. So: set UID equals 4, set GID equals 2, and sticky bit equals 1.

For example, the 1 at the beginning of 1755 indicates that this folder has its sticky bits set.

As wacky as it sounds, the octal numbers (e.g. 644, 1755) that we've constructed via this quick shortcut are both individual digits smushed next to each other (as we've done so far) *and*, taken as whole numbers, the correct octal values for the permissions they describe. In other words: 644, the three digit octal number, is equal to 110100100, the nine digit binary number. That means, for instance, that write permission for group is literally 20 octal, and, for another instance, set UID is literally 4000 octal!

Understanding that principle isn't necessary for day-to-day work; Unix nerds all over lead perfectly happy, productive lives without knowing about it. But, if you're curious about the math behind all this, read on.

## APPENDIX C: USE THE MAN COMMAND

Some Unix gurus like to say “RTFM!” (are-tee-eff-em!) when asked how to do something in Unix. If offered in a helpful way, RTFM stands for “read the fine manual.” If offered in a grumpier way, RTFM stands for “read the %\*#@ \$ manual.”

Whichever way it’s offered, RTFM is good advice. The Unix manual has a wealth of information, and knowing how to find what you need in it is extremely valuable.

To see the manual page (or, simply: man page) for a command, at a shell prompt enter `man` followed by the name of the command. For example: `man ls`. But what if you don’t know the name of the command? You can search by keyword with the `apropos` command. Enter: `apropos keyword`, where *keyword* is the term you’re looking for.

For example, enter: `apropos cron`. The results are:

```
cron(8) - daemon to execute scheduled commands (Vixie Cron)
crontab(1) - maintain crontab files for individual users (V3)
crontab(5) - tables for driving cron
```

Note the numbers in parentheses. Those are the section, or chapter, numbers. To see the first manual page (by section) for a command or file, simply enter: `man command`. However, to see a page other than the first, enter: `man section command` or `man -a command` (to see all pages). For example: `man 5 crontab`.

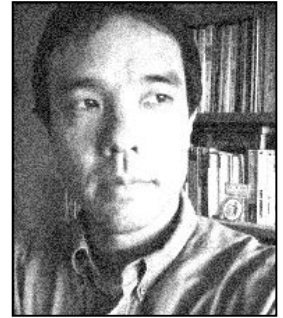
For more information about `man`, enter (wait for it): `man man`. For more **info** about `apropos`, enter: `man apropos`.

## ABOUT THIS BOOK

Thank you for purchasing this Take Control book. We hope you find it both useful and enjoyable to read. We welcome your comments at [tc-comments@tidbits.com](mailto:tc-comments@tidbits.com). Keep reading in this section to learn more about the author, the Take Control series, and the publisher.

### About the Author

Since 1993, Brian Tanaka has worked for a variety of companies including The Well, SGI, Intuit, Nintendo of America, and RealNetworks. He has contributed articles to *LinuxWorld*, *Sysadmin Magazine* and *Linux Journal*.



### Author's Acknowledgements

I want to thank, first and foremost, Tonya Engst for her patience, encouragement, and invaluable editing. I also want to thank Adam Engst, Chris Pepper, Glenn Fleishman, Kirk McElhearn, Dan Frakes, Joe Kissell, Andy Affleck, Matt Neuburg, David Gabbe, and Greg Gilmore for their invaluable tips, comments, and advice.

### About the Publisher

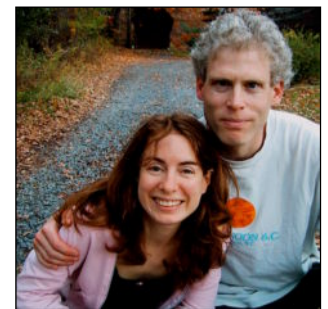
Publishers Adam and Tonya Engst have been creating Mac-related content since they started the online newsletter *TidBITS*, in 1990. In *TidBITS*, you can find the latest Macintosh news, plus read reviews, opinions, and more (<http://www.tidbits.com/>).



Adam and Tonya are known in the Mac world as writers, editors, and speakers. They are also parents to Tristan, who thinks ebooks about clipper ships and castles would be cool.

### Production Credits

Link-making AppleScript: Matt Neuburg  
List macros: Sharon Zardetto  
Take Control logo: Jeff Tolbert  
Tech Editor: Sandro Menzel (some sections)  
Editor in Chief: Tonya Engst  
Publisher: Adam Engst



## *Take Control of Permissions in Leopard*

ISBN: 1-933671-36-X

Version 1.1

Copyright © 2008, Brian Tanaka. All rights reserved.

TidBITS Publishing Inc.

50 Hickory Road

Ithaca, NY 14850 USA

<http://www.takecontrolbooks.com/>

TAKE CONTROL books help readers regain a measure of control in an oftentimes out-of-control universe. Take Control books also streamline the publication process so that information about quickly changing technical topics can be published while it's still relevant and accurate.

The electronic version of this book does not use copy protection because copy protection makes life harder for everyone. So we ask a favor of our readers. If you want to share your copy of this ebook with a friend, please do so as you would a physical book, meaning that if your friend uses it regularly, he or she should buy a copy. Your support makes it possible for future Take Control ebooks to hit the Internet long before you'd find the same info in a printed book. Plus, if you buy the ebook, you're entitled to any free updates that become available.

Although the author and TidBITS Publishing Inc. have made a reasonable effort to ensure the accuracy of the information herein, they assume no responsibility for errors or omissions. The information in this book is distributed "As Is," without warranty of any kind. Neither TidBITS Publishing Inc. nor the author shall be liable to any person or entity for any special, indirect, incidental, or consequential damages, including without limitation lost revenues or lost profits, that may result (or that are alleged to result) from the use of these materials. In other words, use this information at your own risk.

Many of the designations used to distinguish products and services are claimed as trademarks or service marks. Any trademarks, service marks, product names, or named features that appear in this title are assumed to be the property of their respective owners. All product names and services are used in an editorial fashion only, with no intention of infringement of the trademark. No such use, or the use of any trade name, is meant to convey endorsement or other affiliation with this title.

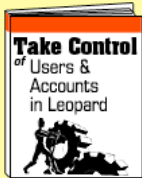
This title is an independent publication, and it has not been authorized, sponsored, or otherwise approved by Apple Inc. Because of the nature of this title, it uses terms that are trademarks or registered trademarks of Apple Inc.; to view a complete list of the trademarks and of the registered trademarks of Apple Inc., visit <http://www.apple.com/legal/trademark/planetmlist.html>.

## FEATURED TITLES

Now that you've seen this book, you know that Take Control books have a great layout and real-world info that puts you in control. Click any book image below or [visit our Web catalog](#) to add to your book collection!

### Take Control of Users & Accounts in Leopard

by Kirk McElhearn



Get the most out of Leopard with real-world strategies for creating and managing user accounts! Covers parental controls.

**\$10**

### Take Control of Sharing Files in Leopard

by Glenn Fleishman



Discover the many improvements to file sharing in Leopard, and learn about giving remote users access in order to share files.

**\$10**

### Take Control of Troubleshooting Your Mac

by Joe Kissell



Whether your Mac won't turn on, kernel panics, or is too slow, you can troubleshoot the problem with Joe's expert advice.

**\$10**

### Take Control of Your iPhone

by Ted Landau

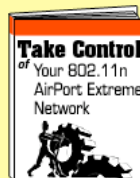


Get the most out of your iPhone! Avoid trouble, solve problems, find hidden tricks, and use your iPhone like a pro.

**\$10**

### Take Control of Your 802.11n AirPort Extreme Network

by Glenn Fleishman



Make your 802.11n AirPort network fly and learn real-world set up and troubleshooting techniques.

**\$10**

### More Titles!

Delve into even more topics, including:

- Running your Mac—upgrading, switching, customizing, fonts, syncing, and sharing files.
- Buying gear—Macs, cameras, and digital TVs.
- More topics—.Mac, email, spam, podcasting, AirPort, Wi-Fi security, GarageBand, iWeb, domain names, and even Thanksgiving dinner!

[Click here to buy the full 86-page "Take Control of Permissions in Leopard" for only \\$10!](#)