Inexpensive Backup Solutions for Open Systems

Covers Windows, Linux, Unix, and OS X

Backup & Recovery

O'REILLY®

W. Curtis Preston

O'REILLY®

Backup & Recovery

Backup & Recovery offers a comprehensive approach to the challenges you face in backing up your data. It provides hands-on, practical, straightforward instructions and advice for backing up and recovering vital systems—without resorting to commercial backup software. It covers everything from basic Linux, Windows, and Mac OS workstations to complicated DB2, Oracle, and Sybase databases, and a lot of things in between. *Backup & Recovery* helps you develop a philosophy of backup and describes how to schedule backups for optimum effectiveness and coverage.

Highlights include:

- Basic backup tools, including dump, tar, cpio, ditto, dd, ntbackup, and rsync
- Coverage of open-source backup tools including Amanda, BackupPC, and Bacula
- How to achieve near-continuous data protection using open-source tools
- A step-by-step approach to bare-metal recovery of Windows, Linux, Mac OS, VMware, HP-UX, AIX, and Solaris systems (it's not as hard as you think!)
- How to back up and recover Oracle, DB2, Sybase, MySQL, PostgreSQL, SQL Server, and Exchange, whether you're a sysadmin or a DBA
- Criteria for evaluating commercial solutions and for determining whether open-source solutions can meet your needs
- Criteria for evaluating all types of backup hardware, including tape, optical, and disk-based targets such as virtual tape libraries (VTLs)

It's not just about the tools: *Backup & Recovery* is full of practical lessons, tricks, and advice to help you solve backup and recovery problems of any size. Whether your budget barely stretches to cover the cost of the backup media or allows you to buy a silo bigger than your house, this book will help you formulate a practical backup and recovery strategy that meets your needs.

W. Curtis Preston has specialized in designing data-protection systems since 1993 and has designed such systems for many environments, both large and small. His lively prose and wry, real-world approach has made him a popular author and speaker.

"W. Curtis Preston is the most knowledgeable person I know about backup and recovery. Not only does he know his stuff, but he has a unique knack of making such a complex subject interesting and understandable. Read his book: I guarantee, you'll learn something—make that, a lot of things."

—Rich Friedman, Features Editor, *Storage* Magazine

oreilly.com

US $54.99 **CAN $72.99**

ISBN: 978-0-596-10246-3

Backup and Recovery

W. Curtis Preston

O'REILLY®
Beijing • Boston • Farnham • Sebastopol • Tokyo

Backup and Recovery
by W. Curtis Preston

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Editor: Mike Loukides and Debra Cameron
Production Editor: Lydia Onofrei
Copyeditor: Mary Anne Mayo
Proofreader: Lydia Onofrei

Indexer: Reg Aubry
Cover Designer: Karen Montgomery
Interior Designer: David Futato
Illustrators: Robert Romano and Jessamyn Read

Printing History:

December 2006: First Edition.

10-digit ISBN: 0-596-10246-1
13-digit ISBN: 978-0-596-10246-3
[LSI] [04/17]

This book is dedicated to those brave men and women who give themselves in service to our country.

"Greater love has no one than this, that one lay down his life for his friends."

—John 15:13

Table of Contents

Part II. Open-Source Backup Utilities

Part V. Database Backup

Preface

I hope you learn half as much reading this book as I did writing it. This was quite an interesting project, where we took the original book and expanded its scope so much that we had to change its title. I wrote *Unix Backup and Recovery* seven years ago, and a lot has changed since then—both in the industry and in my life. The biggest change in the industry has been the proliferation of Windows, Mac OS, Exchange, and SQL Server in the data center. (I never saw the Apple Xserve coming.)

The biggest change for me has been having my eyes opened to backup and recovery applications beyond those considered "traditional." It's true that I spend most of my professional life consulting with large companies that spend enough on backup software and hardware to fund a small army. I enjoy doing that. It's very rewarding to show a company how to save millions of dollars a year and make their backups and restores faster and more reliable in the process. (By the way, if you need help with your backup system, drop an email to *curtis@backupcentral.com*—that's what I do for a living.)

I also spend a good deal of the time traveling the world speaking to users about how to do this themselves. And when I do, I always get questions like these:

- I got a quote for backup software from XYX, and they want $XXXX for backup software! Where am I supposed to get that kind of money!?
- I couldn't afford backup software from XYZ, so we bought ABC instead, and it stinks. Can you recommend something better?
- None of the commercial utilities can back up my MySQL or PostgreSQL database. How do I do that?
- How do I do bare-metal recovery on ABC operating system?
- Aren't there open-source utilities that do this kind of thing?

So while I'm actually preparing to write my next book on how to select, install, and manage commercial backup software systems, I felt that this book needed to come first. This book is aimed at the people who feel that the commercial software products aren't meeting all their needs.

Perhaps you're a small shop that can't spend $10,000 just to get decent backup software. Perhaps you're already using a commercial backup software package, but you don't want to spend thousands of dollars on their agent to back up your DB2 databases, or you can't find anybody to back up your MySQL or PostgreSQL databases. This book is about giving you options—*free* options.

Almost everything I talk about in this book is either included with your operating system or application, or is available as an open-source project. (The commercial products I do mention cost only $99.) You may be amazed at what you can do for free or almost free.

I Wish I'd Had This Book

I wanted to write a book that would ensure that no one would ever have to start from scratch again, and I believe that my contributors and I have done just that. It contains every backup tool that I wish I had when I first entered the backup business and every lesson and trick that I've learned along the way. It covers how to back up and recover everything from a basic Linux, Windows, or Mac OS workstation to a complicated DB2, Oracle, or Sybase database—and a lot of things in between. Whether your budget barely stretches to cover the cost of the backup media or allows you to buy a silo bigger than your house, this book has something for you. Whether your task is to figure out how to back up, with no commercial utilities, an environment such as the one I first encountered or to choose from among more than 50 commercial backup utilities, this book will tell you how to do it. With that in mind, let me mention a few things about this book that are unique.

Only the Recovery Matters

As my friend Joe Fitzpatrick used to tell me, "No one cares if you can back up—only if you can recover." Yet how many backup chapters have you read that dedicate less than 10 percent to recovery? You won't find that in this book. I have tried very hard to ensure that recovery is given equal treatment.

Products Change

Some people may be surprised that there are no product names mentioned in the commercial backup section. I did this for several reasons, the main one being that products change constantly. It would be impossible to keep this book up to date

with more than 50 backup products that are available for Unix alone. In fact, the book would be out of date by the time it hit the shelves. Instead, this book explains the *concepts* of commercial backup and recovery software, allowing you to apply those concepts to the claims that the vendors are currently making. Up-to-date information about specific products is available on *http://www.backupcentral.com*.

Backing Up Databases Is Not That Hard

If you're a database administrator (DBA), you may not be familiar with the commands necessary to back up your database. If you're a system administrator (SA), you may not be familiar with the architecture of the database platform your DBA is using. Both concepts are explained in detail in this book. I explain the backup utilities in plain language so that any DBA can understand them, and I explain database architecture in such a way that an SA, even one who has never before seen a database, can understand it.

Bare-Metal Recovery Is Not That Hard

One of these days you will lose the operating system disk for an important system, and you will need to recover it. This is called a *bare-metal recovery*. The standard recovery method described in many backup products' documentation is to install a minimal operating system and restore on top of it. This is the worst possible method to do a bare-metal recovery of a system; among other problems, you end up overwriting some of the system files while the system is running from the very disk to which you are trying to restore. The best ways to do bare-metal recoveries for AIX, Solaris, HP-UX, Windows, Linux, and Mac OS are covered in detail in this book.

How This Book Is Organized

This book is divided into six parts, which are described in the following sections.

Part I

Part I of this book contains just enough information to whet your backup and recovery appetite.

Chapter 1, *The Philosophy of Backup*
: Describes my philosophy about backup, such as why you should back up, and a little bit about how to do it, too.

Chapter 2, *Backing It All Up*
: Goes into detail about the essential elements of a good backup and recovery system.

Part II

This section covers the basic backup utilities that are available to back up your system, and several open-source backup systems to help you manage those backups.

Chapter 3, *Basic Backup and Recovery Utilities*
: Covers the basic backup and recovery utilities you're likely to find in Unix, Windows, or Mac OS, including `dump`, `tar`, `cpio`, `dd`, `ditto`, `ntbackup`, and `rsync`.

Chapter 4, *Amanda*
: Covers the ever-popular Advanced Maryland Disk Archiver, or Amanda.

Chapter 5, *BackupPC*
: Explains the disk-only backup system called BackupPC, which can actually back up far more than just your PC.

Chapter 6, *Bacula*
: Covers Bacula. It roams the data center at night and sucks the vital essence from your computers.

Chapter 7, *Open-Source Near-CDP*
: Covers three near continuous data protection (near-CDP) products, including `rsync` with snapshots, `rsnapshot`, and `rdiff-backup`.

Part III

If you have outgrown the capabilities of free utilities or would just like to take advantage of new backup and recovery technologies, you'll need to look at a commercial product. You should also know about the latest hardware that is on the market to assess your full range of backup and recovery options.

Chapter 8, *Commercial Backup Utilities*
: Is your guide to the hundreds of features available in the over 50 commercial backup products available on the market today, allowing you to make an educated purchase decision.

Chapter 9, *Backup Hardware*
: Explains the many different types of backup hardware available today, and provides criteria to help you decide which type of backup drive is right for you.

Part IV

A bare-metal recovery is the fastest way to bring a dead system back to life, even if its operating system drive is completely destroyed.

Chapter 10, *Solaris Bare-Metal Recovery*
: Explains Sun's flash archive product, which is the Solaris equivalent of AIX's `mksysb`.

Chapter 11, *Linux and Windows*
: Explains a number of procedures and tools that can be used to perform bare-metal recovery of both Linux and Windows systems. It includes a discussion of Ghost for Linux (G4L) an open-source ghosting product.

Chapter 12, *HP-UX Bare-Metal Recovery*
: Covers the `make_net_recovery` and `make_tape_recovery` tools, which now come with HP-UX to perform bare-metal recoveries.

Chapter 13, *AIX Bare-Metal Recovery*
: Discusses AIX's `mksysb`, probably one of the oldest and best-known bare-metal recovery tools.

Chapter 14, *Mac OS X Bare-Metal Recovery*
: Covers how to perform your own bare-metal recovery of a Mac OS X machine.

Part V

This section explains in plain language an area that presents some of the greatest backup and recovery challenges that a system administrator or database administrator will face—backing up and recovering databases.

Chapter 15, *Backing Up Databases*
: Explains database architecture while relating each architectural element to the appropriate term in DB2, Exchange, Informix, MySQL, Oracle, PostgreSQL, SQL Server, and Sybase. This chapter will be your friend if you're an SA who's afraid of databases or a DBA learning a new database.

Chapter 16, *Oracle Backup and Recovery*
: Explains how to perform Oracle hot backups using `rman` or user-managed backup.

Chapter 17, *Sybase Backup and Recovery*
: Shows how to use the backup server to back up Sybase ASE.

Chapter 18, *IBM DB2 Backup and Recovery*
: Explains how to back up and recover DB2 databases.

Chapter 19, *SQL Server*
: Explains how to back up and recover SQL Server databases.

Chapter 20, *Exchange*
: Explains how to back up and recover Exchange databases using the built-in `ntbackup` plug-in for Exchange.

Chapter 21, *PostgreSQL*
: Explains how to back up and recover PostgreSQL databases.

Chapter 22, *MySQL*
: Provides an overview of the various backup and recovery options available for MySQL.

Part VI

The information contained in this part of the book is by no means unimportant; it simply wouldn't fit anywhere else!

Chapter 23, *VMware and Miscellanea*
: Includes VMware backups, the oft-debated "live filesystem dumps" question, and even some backup poetry.

Chapter 24, *It's All About Data Protection*
: Provides some food for thought, discussing the fact that backups are not the answer to all problems; you should also be thinking about other areas of data protection, such as archiving, disaster recovery, and storage security.

What's New in This Book

See preceding section. Seriously, this book has about 75 percent new material when compared with *Unix Backup & Recovery*. Some chapters in the first book were completely rewritten for this book. Here are the highlights of those changes:

A new philosophy
: This book reflects my new backup philosophy, which is that it's all about disk—especially for smaller shops.

New backup commands
: We've added `ntbackup`, `ditto`, and `rsync` to the basic utilities chapter.

Amanda
: The Amanda chapter is completely updated to reflect the developments of the past seven years.

Commercial utilities
: The commercial utilities chapter has been updated to reflect the advances in backup and recovery in the past seven years.

HP-UX
: The `make_net_recovery` and `make_tape_recovery` tools have changed, and so has the chapter covering them.

Backup hardware
: Boy, has hardware changed in seven years! Disk targets, virtual tape libraries, and data de-duplication systems. I cover it all.

There are eleven *completely new chapters*, significantly expanding the scope of this book. Here are the topics covered in these new chapters:

DB2
: How to back up DB2 using its built-in capabilities

Exchange
: How to back up Exchange using `ntbackup`

SQL Server
: How to back up SQL Server using its built-in capabilities

MySQL
: How to back up and recover MySQL databases based on the MyISAM, InnoDB, and NDB storage engines

PostgreSQL
: How to back up and recover this popular open-source database using either `pg_dump` or `pg_dumpall`

BackupPC
: How to use BackupPC, a completely disk-based backup and recovery system with a web frontend

Bacula
: How to use Bacula, an open-source backup product that roams the datacenter at night and sucks the vital essence from your computers

Near-CDP
: How to use snapshots and replication to make backups

Solaris
: How to do bare-metal recovery using flash archive

Linux and Windows bare-metal recovery
: How to use a Linux LiveCD or Ghost for Linux to perform bare-metal recovery of Windows and Linux operating systems

Mac OS X
: How to use the built-in, bare-metal recovery in OS (it isn't too hard)

What's Missing?

For various reasons, some chapters from *Unix Backup & Recovery* did not make it into this book. All of the following chapters are now available online at *http://www.backupcentral.com*. The one challenge, of course, is that these chapters have not been updated. Therefore, we've put them in our wiki so that anyone who wants to help us update them can do so.

- Tru64 Bare-Metal Recovery
- IRIX Bare-Metal Recovery
- Informix Backup and Recovery
- Clearcase Backup and Recovery
- High Availability

Speaking of BackupCentral.com

We've completely redesigned *http://www.backupcentral.com* using a content management system, forums, and MediaWiki. My number one goal for the new Backup Central is to make it much easier to provide you dynamic content and to build a strong community around backup and recovery issues. The new Backup Central has some really great features:

- phpBB forums for various backup-related topics, including one for discussing the book. Come join the discussions.
- A mailing list for each forum, allowing you to follow the discussions via the forum or email. Any posts in the forum are sent to the mailing list, and emails sent to the mailing list result in posts or replies in the forum.
- A multidirectional connection between backup-related Usenet newsgroups, mailing lists, and phpBB forums. One of the things I was reminded of while writing this book is that Usenet is alive and well, and I want to bring this great resource to the Backup Central community and to create another portal into this underutilized resource. Each relevant Usenet newsgroup has an associated mailing list and forum, and all messages to Usenet, the mailing list, or the forum go to the appropriate forum, mailing list, and newsgroup.
- A wiki based on MediaWiki, the same software that drives Wikipedia. One of the things you will find there is a wike entry for every chapter in this book. We're going to use these entries to update and further the ideas you find in this book. We're doing this for two reasons:
  - The first reason is that one of the problems with writing a technical book is that the second you go to press, something changes. While we're in the process of getting this book printed, MySQL will come out with three more storage engines, QTParted will probably support NTFS, and Bacula's Windows Server will become generally available. We'll use the wike to keep things like this up to date.
  - The second reason is because my contributors and I don't know all the answers, folks. We did our best to come out with a solid book for you, but we haven't seen everything you've seen. We'd love it if you help us further the ideas mentioned in this book, help us to explain the scenarios under which a given procedure won't work, or how a given procedure should be enhanced. (For example, right now a friend of mine is trying to help me understand how to get `rsync` to better handle millions of files. His testing won't be done in time for press. Put it in the Wiki, Jason.) Join the new Backup Central and save the world—or at least its data.

I'll see you at *http://www.backupcentral.com*!

Conventions Used in This Book

The following typographical conventions are used in this book:

`Constant width`
: Indicates command-line computer output, computer-generated messages, and code examples. It is also used when referring to commands and parameters in text.

`Constant-width italic`
: Indicates variables in text.

`Constant width bold`
: Indicates user input in command-line examples.

`Constant width italic bold`
: Indicates variables in command-line examples.

Italic
: Introduces new terms and indicates URLs, files, directories, hostnames, and file extensions.

How to Contact Us

We have tested and verified all the information in this book to the best of our ability, but you may find that features have changed (or even that we have made mistakes!). Please let us know about any errors you find, as well as your suggestions for future editions, by writing to:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international/local)
707-829-0104 (fax)

We have a web page for the book that lists examples, errata, or any additional information. You can access this page at:

http://www.oreilly.com/catalog/9780596102463/

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about books, conferences, Resource Centers, and the O'Reilly Network, see the O'Reilly web site at:

http://www.oreilly.com

Safari® Enabled

When you see a Safari® Enabled icon on the cover of your favorite technology book, that means the book is available online through the O'Reilly Network Safari Bookshelf.

Safari offers a solution that's better than e-books. It's a virtual library that lets you easily search thousands of top tech books, cut and paste code samples, download chapters, and find quick answers when you need the most accurate, current information. Try it for free at *http://oreilly.com/safari*.

This Book Was a Team Effort

It's true; my license plate does say MRBAKUP. But that doesn't mean I know everything about backup and recovery. In fact, I've never even used some of the operating systems or database platforms covered in this book! It would be a disservice to you, the reader, for me to write chapters on those products—but I wanted the chapters in the book. So I hired a team of experts to write the chapters for you. Approximately 250 pages of this book were written by others, and contributors are recognized at the beginning of the chapter(s) they wrote.

Contributors

It's not an easy thing to write a chapter in someone else's book. Not only do you have to write, but you have to write based on someone else's design. There are also tight deadlines, and the process is nothing but hurry up and wait. I couldn't have done it without them, so please allow me to formally thank all of my contributors.

Amanda
: Contributed by Dmitri Joukovski and Stefan G. Weichinger. Thanks for seeing this one through.

BackupPC
: Contributed by Don "Duck" Harper. Quack!

Bacula
: Contributed by Adam Thornton. Thanks for bringing Bacula to the book.

Near-CDP
: Contributed by Michael Rubel, Ben Escoto, and David Cantrell. This chapter morphed a few times, and I appreciate your patience as it gelled in my head.

AIX bare-metal recovery
: Contributed by Mark Perino. I think you are the fastest writer on the team.

HP-UX bare-metal recovery
: Contributed by Eric Stahl and Ron Goodwyn. Great collaborative effort, guys.

Linux and Windows bare-metal recovery
: Contributed by Reed Robins. Maybe we can do it this way, or that way, or that way! Did I change the scope of the chapter enough? Thanks.

Mac OS X bare-metal recovery
: Contributed by Leon Towns-von Stauber. Thanks, Leon. I sure am glad Mario told me to give you a call. Your chapter was perfect.

Solaris bare-metal recovery
: Contributed by Aaron Gersztoff. Be careful what you ask for, right, Aaron?

DB2 backup and recovery
: Contributed by Jeff Richardson, Kulvir S. Bhogal, and Kondal Yennaram. You guys all came through in a pinch, and I'm most grateful.

Exchange backup and recovery
: Contributed by Scott Harris. More pictures! Fewer pictures! Make it like this, no make it like that! Isn't it fun writing for me?

SQL Server backup and recovery
: Contributed by Scott Harris. Look at that, Scooter! You're the only one who was crazy enough to write two chapters for me. Thanks.

Sybase backup and recovery
: Contributed by Edward Barlow, who updated a chapter originally written by Bryn Smith. Another contributor I couldn't have done without. Thanks.

Dump internals
: Contributed by David Young. When are you going to move out here with your Mom?

Without these folks, this book would contain substantially less information than what you find here.

Technical Editors

Another group of people I must thank is our technical reviewers, and we had a *lot* of them! The problem with writing a book with a scope this big is that you need specialists and tech reviewers as well. Because of that, most technical editors reviewed only one or two chapters. I couldn't have done it without them. I'm sure I've miss a few, but here's my best attempt at listing them all (alphabetically by first name):

Adrin Kow	Andy Shellam	Anthony Johnson
Axel Schwenke	Ben Garrett	Brian Eliassen
Brian Peasland	Charles Whealton	Chris Thomas
Christoph Haas	Craig Barratt	D.A. Morgan
Dana Diederich	Daniel Callahan	Dave Mehler
David Boyd	Edward Conba	Eric Gilmore
Eric Stahl	Finn Henningsen	Frank Sweetser
Greg Lehey	Ian Gorrie	Ian Herd

James Bougor
Jeff Frost
Jeffrey P. Humes
John Madden
Lenz Grimmer
Mark Dawson
Matthew Huff
Mohammed Mehdi
Patrick Matthews
Rob Worman
Scott Boss
Simon Riggs
Tammy Bednar
Vitalis Jerome
Jayesh Thakrar
Jeff Harbert
John Haight
Kern Sibbald
Marcel Lans
Mark Perino
Megan Restuccia
Neal A. Lucier
Paul Muggeridge
Rodrigo Real
Scott Harris
Steve Hanson
Todd Toles
Wil Coulbourn
Jeff Badger
Jeff Richardson
John Hurley
Kumar Sundaram
Mark D. Powell
Massimiliano Daneri
Mike Harrold
Norbert Munkel
Ralph R. Hirtler
Satyaprakash Pandey
Shane Seymour
Stewart Smith
Víctor A. Rodríguez
William Cole

Horror Stories

Also giving this book some flavor are those who contributed horror stories. Even if I couldn't use your story in the book, I want to thank you for sending one in.

Brian O'Neill
David Bregman
Hywel Matthews
Jason Frankovitz
Jim Donnellan
Karl Langdon
Michael Rice
Richard Ackerman
William Birch
Brian Sakovitch
David J. Young
Jack Coats
Jason Shupe
John Merryman
Kevin Suttle
Michael Tobin
Scott Boss
William S. Duncanson
Chris Pritchard
Harry Tirrell
James Hunt
Jim Damoulakis
Jorgen Lie
Mark Perino
Natalie Meek
Theo Van Dinter

Special Mention

There were a few people who were extremely helpful in one way or another throughout this project. I'd like to send a special thank you to them.

Anthony Johnson

It's not every CEO who would volunteer to tech review a chapter that is essentially a free competitor with his own product. I hope you and Storix do very well. You've got quite the Linux and AIX bare-metal recovery tool there.

Brian Peasland

You beat the living crap out of the first draft of the Oracle chapter, and rightfully so. The new chapter is *significantly* better thanks to your thorough and honest review. (You made me rewrite half of it, dude!)

Deb Cameron
: Isn't it fun editing a book with 18 authors from 3 continents, several time zones, a few different native languages, using about 60 technical editors? Let's do it again sometime!

Joshua D. Drake
: Thanks, Joshua for spending the time you did with me on the phone to help me better understand PostgreSQL. Next time, you should be more forthright with your own opinions. A guy really doesn't know where he stands with you.

Lenz Grimmer
: You were my liaison in the MySQL community, and I definitely needed the help. Thanks to you and the whole MySQL team.

Lynn Stone
: Thank you for helping get this project off the ground. I couldn't have done it without you. Only I know your secret identity.

Tammy Bednar
: For such a busy woman, you gave me exactly what I needed for the Oracle part of this project. You've obviously done a lot of work on the Oracle backup products, and it shows. I hope you'll see my newfound respect for your products in the Oracle chapter.

Zmanda
: Thanks for the support on the Amanda chapter, and for bringing commercial support to a very popular open-source backup tool.

Mario Obejas
: Thank you so much for referring me to Leon.

I Don't Know It All

If there's one thing I learned while writing this book, it's that I do not know everything there is to know about backups. If you have a better way to do anything described in this book, have learned any special tricks, or have written any neat utilities that you think would help other people do backups and recoveries, let me know. Email me at *curtis@backupcentral.com*. Your tricks or utilities may be included in the next edition of the book and listed immediately on *http://www.backupcentral.com*.

How Can I Say Thanks?

How can I begin to thank the hundreds of people who helped me?

To God: May any praise for this book go to You alone.

To my wife, Celynn: I say "thank you" for the many nights you spent alone while I pounded away at my keyboard somewhere around the globe. You're a special woman who never gave up on me or my dream. I love you.

To my daughter, Nina: You were only seven when the first book came out. Now you're a beautiful young lady who is growing up so fast. I'm going to have to get a gun and sit on the porch.

To my daughter, Marissa: You were only two when the first book came out. Now you're a beautiful nine-year old—my, how time has flown. Let's go to the park and ride our bikes together.

To my parents: What can I say? You always believed in me. You always used to tell me, "I don't care if you're a ditch digger. Just be the best darn ditch digger in the world." Well, being a backup guy is as close as you can get to being a ditch digger in the computer business, and I "wrote the book" on that.

To Bob Walker for helping me get my first job in backups, and Ron Rodriguez for being all too eager to give it to me.

To Susan Davidson, who didn't fire me when I couldn't recover that purchasing database in 1992: that second chance was all I needed to become the expert in backup that I am today. If you had fired me (and I'm sure a few people wanted you to), who knows where I'd be today. (If you're curious about the story, look for the sidebar "The One That Got Away" in Chapter 1.)

To Collective Technologies for helping me round out my skills enough to see that I wanted to specialize in backup and recovery, and for supporting me when I wrote the first book.

To Jason Stege, Robin Young, Jeff Williams, Reed Robins, and Elia Harris: Thanks for believing in me when I started my own company. I hope I did right by you.

To Mark Shirman and all my friends at GlassHouse: Thanks for giving me a place where I finally feel like I'm using my talents.

To my wife's family: Thank you for raising such a wonderful lady. Thank you for treating me as one of your own and supporting us on our quest. *Pahingi ng sinagang?*

To all the teachers who kept trying to get me to live up to my potential: You finally got through.

To O'Reilly: Thank you for the opportunity to bring this much-needed book to market.

To Deb Cameron and Michael Loukides, my editors: We'll have to actually meet one of these days! I don't know how you do this, reading the same book over and over, without letting your eyes just glaze over. You're great editors, and I could really tell that you put your all into this project. Thank you, thank you, and thank you. (Now don't edit *that* sentence, OK?)

To the reader: Thank you for purchasing this book. I hope you learn as much reading it as I did writing it.

PART I
Introduction

Part I consists of the following two chapters:

Chapter 1, *The Philosophy of Backup*
: Describes why you should back up, and a little bit about how to do it.

Chapter 2, *Backing It All Up*
: Goes into detail about the essential elements of a good backup and recovery system.

CHAPTER 1

The Philosophy of Backup

I back up; therefore, I will be.

When I look at the title of this chapter, I think about the old Steve Martin stand-up routine in which he said that in philosophy class, "you learned just enough to screw you up for the rest of your life." (Steve studied the important questions, like "Is it OK to yell 'movie' in a crowded fire house?" I promise not to do that.) However, "The Philosophy of Backup" did seem like an appropriate name for this chapter, since we're going to talk about the *why* of backup. (We'll also talk a little about the *how*, of course.)

Champagne Backup on a Beer Budget

A good backup and recovery system is essential for a company of any size. Unfortunately, IT doesn't always get the budget it needs, and the backup system almost never gets the money that it needs. Well, if you agree that you need a very good backup system, but you don't have enough money to pull that off, know that this book was written with you in mind. You need champagne backup on a beer budget. Welcome to the club.

Just because you have a small budget doesn't mean you have to do without backup. Most of the backup systems in this book can be implemented in small environments for a few hundred dollars—including hardware.

Don't worry, enterprise customers—there's plenty in here for you as well. The more you use the techniques taught in this book, the more money you can save for other IT projects. By the time you're done implementing all the ideas in this book, hopefully my next book will be done, which will be right up your alley. It will cover nothing but commercial data protection solutions, including multiplatform commercial backup and recovery systems, continuous data protection, near continuous data protection, data de-duplication backup systems, replication, and the like.

Now that you've read this far, you may find yourself asking questions like these:

- Why should I read this book?
- Can I really back up with open-source backup software?
- Why should I be using disk?
- Why should I back up at all?
- How do I find a balanced way to back up (wax on/wax off)?

Let's get started answering these questions.

Why Should I Read This Book?

If you've been doing system administration for some time, you may be asking yourself this question. There are many answers. Perhaps self-preservation is your primary motivator. You'd like to make sure you don't lose your job the next time a disk drive dies. Perhaps you've already got a decent backup system and you'd just like to make it better. Maybe you are looking for some new ideas about how to deal with upcoming backup and recovery needs. What follows are some of the reasons *I* think you should read this book.

Schadenfreude

Schadenfreude is a German word that means to take joy in the misfortunes of others. It's why we watch those weird videos on the Internet where some idiot tries to do something stupid and ends up hurting himself. Each of the sidebars in this book is a true horror story that really happened to someone I know. These are not urban legends or horror stories passed on from admin to admin. These are firsthand encounters with disaster. There's a schadenfreude element to reading these stories, of course. But each story also makes a point, and it was not just made up to make that point. The things that I warn about in this book really happen. This can be a very tough job if you are not prepared, so read closely. You might want to start by reading the sidebar "The One That Got Away" later in this chapter. It's the story of the defining moment in my career.

You Never Want to Say These Words

"We lost only a few days' worth of data." In the sidebar "The One That Got Away," I said that we lost only a few days worth of data. I swore the day I said these words that I would never say them again. From that day forward, I was convinced of the importance of backups. I never again assumed anything, and I began to study everything I could about backup technology. This book represents my attempt to compile what I have learned about inexpensive backups into a single volume, and it is written so that no one who reads it should ever need to utter the preceding statement. In

The One That Got Away

"You mean to tell me that we have absolutely no backups of *paris* whatsoever?" I will never forget those words. I had been in charge of backups for only two months, and I just knew my career was over. We had moved an Oracle application from one server to another about six weeks earlier, and there was one crucial part of the move that I missed. I knew very little about database backups in those days, and I didn't realize that I needed to shut down an Oracle database before backing it up. This was accomplished on the old server by a `cron` job that I never knew existed. I discovered all of this *after* a disk on the new server went south.

"Just give us the last full backup," they said. I started looking through my logs. That's when I started seeing the errors. "No problem," I thought, "I'll just use an older backup." The older logs didn't look any better. Frantic, I looked at log after log until I came to one that looked as if it were OK. It was just over six weeks old. When I went to grab that volume, I realized that we had a six-week rotation cycle, and we had overwritten that volume two days before.

That was it! At that moment, I knew that I'd be looking for another job. This was our purchasing database, and this data loss would amount to approximately two months of lost purchase orders for a multibillion-dollar company.

So I told my boss the news. That's when I heard, "You mean to tell me that we have absolutely no backups of *paris* whatsoever?" Isn't it amazing that I haven't forgotten its name? I don't remember any other system names from that place, but I remember this one. I felt so small that I could have fit inside a 4 mm tape box. Fortunately, a system administrator worked what, at the time, I could only describe as magic. The dead disk was resurrected, and the data was recovered straight from the disk itself. We lost only a few days' worth of data. Our department had to send a memo to the entire company saying that any purchase orders entered in the last two days had to be reentered. I should have framed a copy of that memo to remind me what can happen if you don't take this job seriously enough. I didn't need to though; its image is permanently etched in my brain.

Some of this book's reviewers said things like, "That's pretty bold! You're writing a book on backups, and you start it out with a story about how you messed up. Some authority you are!" Why did I include it? Through all the years, and all the outages, this one sticks in my mind. Perhaps that's because it's the only one that almost "got me." Had it not been for the miraculous efforts of a wonderful administrator named Joe Fitzpatrick, my career might have been over before it started. I include this anecdote because:

- It's the one that changed the direction of my career.
- There are several valuable lessons that I learned from it, which I discuss in this book.
- It could have been avoided if I had had a book like this one.
- You must admit that it's pretty darn scary.

my opinion, *no amount of data loss is acceptable*. I would also wager that you would be hard-pressed to find an end user who would feel much different. Whether it's a spreadsheet that one person created or a customer database representing hours or days of sales invoices and the efforts of hundreds of people—ask the person who needs the data how much data loss they think is acceptable. Every statement, every opinion, every story, and every chapter in this book is based on the premise that any data loss is unacceptable. Let me state that again for emphasis.

With the technology that is now available, there is no reason for any data to be lost—that is, *if* backups are given the proper attention and priority that they need.

You're Curious About Open-Source Backup Products

Just a few years ago, you could perform your backups with a few scripts and `dump`, `tar`, or `cpio`, or `ntbackup`. The demand for midrange computers grew astronomically, and the need for bigger databases, larger drives or filesystems, long filenames, and long pathnames grew proportionally. These large databases and filesystems started shipping, which then created a large market for commercial backup utilities, and one or two such products emerged; scores of others eventually followed.

Some of these early products were just GUIs and volume management built on top of existing native backup utilities to provide enhanced levels of functionality. Other companies felt that these native utilities had many limitations that could not be fixed without abandoning them altogether. Those companies chose to develop custom, even proprietary, backup methods. They attempted to overcome the limitations that products based on `dump` and `tar` could not overcome.

In recent years, the demand for centralized backup and recovery has also given rise to a number of open-source backup and recovery tools, six of which are covered in this book. The open-source backup market followed a pattern similar to the commercial products mentioned. The original open-source backup product, Amanda, is a wrapper around the native utility of your choice. BackupPC leaves data in its original format, and Bacula uses a custom format designed to overcome the limitations of GNU `tar`.

There are now a number of choices in the open-source backup market. It's quite possible that one or more of the open-source products covered in this book can meet your backup and recovery needs. This book is currently the only resource that covers all of these tools in a single place.

You Want to Learn About Disk-Based Backup

If you haven't heard of disk-based backup or disk-to-disk-to-tape (D2D2T) backup, then it's time to turn off the digital video recorder (DVR) and pick up a trade magazine or two. (Of course, your DVR is nothing more than disk-based backup of your

TV. And if you're occasionally making VHS tapes of your DVR shows, it's even a D2D2T system.) The use of disk in backup and recovery systems has exploded in the last few years, and it's really solving a lot of problems.

Chapter 9 covers backup hardware and goes into much more detail about why disks have become a very attractive backup target. Here is a quick summary of some of those reasons:

Cost
: The biggest reason that disk has become such an attractive backup target is that the cost of disk has been dramatically reduced in the last few years. The cost of a reasonably priced disk array is now approximately the same price as a similarly sized tape library filled with media. When you consider some of the things you can do with disk, such as eliminating full backups and redundant files, disk becomes even less expensive.

Reliability
: Unlike tapes, disks are closed systems that aren't susceptible to outside contaminants. In addition, the actual media of a hard drive is, well, *hard* when compared to a piece of tape media. The result is that an individual disk drive is inherently more reliable than a tape drive. Disk drives become even more reliable when you put them in a RAID array.

Flexibility
: Generally speaking, tape drives can only go two speeds: *stop* and *very fast*. Yes, some tape drives support variable speeds. However, they can usually only slow down to about 40 percent of the rated speed of the drive. Disk drives, on the other hand, work at whatever speed you need them to go. If you need to go a few hundred megabytes per second, put a few drives in a RAID group, and blast away. Then if you need that same RAID group to write at 10 KB/s, go ahead. Unlike tape drives, disk drives have no problem writing slowly, then quickly, then slowly, then.... You get the picture. This makes disk a perfect match for unpredictable backup streams. Once all that random data has been written in a serial fashion on your disk device, the disks can easily stream backup data to tape—if that's what you want to do. Some people are foregoing that step altogether and replacing it with replication. Try doing that with a tape drive.

Disk-based backups are also an extremely economical way to bring completely automated backups to small and medium businesses (SMBs). While a large tape library can be very inexpensive (on a dollars-per-gigabyte basis) and very expandable, the same is not always true of smaller libraries aimed at the SMB market. The big challenge is expandability. The less expensive a tape library is, the less expandable it usually is. (There are always exceptions, of course.) By comparison, some of the completely automated open-source backup products mentioned in this book can be used with a single disk drive costing less than $100. If you need to expand beyond that, just buy another disk and add it to your volume manager. You can also buy

RAID controllers that allow you to start with one disk and add more as your needs grow. You can use this method to expand from hundreds of gigabytes to many terabytes of capacity.

Why Back Up?

I've heard it all. I've been accused of caring only about backups. It's been said that I think the whole world revolves around a cartridge reel. I've said that someday the world's going to crash, and I'm going to have the backup. The question is: how serious are *you* about protecting your data? To help you come to a decision on this matter, let's talk about what happens if you don't have good backups.

That's Not What I Meant!

I was administering a QA group for a major software company. When the software did its setup, it created a directory in *$HOME/foo* so the software could install the user portion. A QA person was doing the install under root. The software created the directory *$HOME/foo*; literally, *$HOME* was the directory name. He submitted the bug fix, and decided to get rid of the useless directory. You've probably guessed it:

```
# rm -rf $HOME
```

(This was on a standard Unix system where *$HOME* for root was still */*.)

Once I finally stopped laughing, I went and got the install media to rebuild the machine (no jumpstart image for that one, unfortunately, nor any backups for the various QA servers). Fortunately, most of the critical data was on the NFS server.

—*William Birch*

What Will Lost Data Cost You?

To answer this question, you need to consider what kind of data you are backing up. This is a perfect time to include people who may not consider themselves computer people. Get input from other departments to answer this question. When all those 1s and 0s come together, just what kind of information are we talking about? Do you use manual accounting methods or are your company's financial records stored in some accounting software somewhere? When a customer calls in and orders something, do you jot that down on a carbon-copied order form or do you enter it in some sort of order processing program? What about things like budgets, memoranda, inventories, and any other "paperwork" that you throw around from day to day? Do you keep copies of every important memo that you send, or do you depend on the computer for that?

If you're like most people, you have grown quite dependent on these things we call computers. You forget how much of your work has been saved in the form of little magnetized bits spread out across a bunch of spinning platters. Maybe you work in an environment in which you've never lost a disk, so you've never had to do a restore. Maybe you've never fat-fingered a key and deleted an important file. If that's the case, remember what my dad used to say: "motorcycle riders come in two types—those who have fallen and those who will fall." The same is true of disk drives. If you've never had a failed disk drive, trust me, your turn is coming!

So what would you lose if you lost data? To quantify this, we need to examine the types of information that may reside in your environment and what would happen if you lost each type of information. Most of what you could lose is very tangible—and quantifiable in monetary terms—and it might surprise you.

Lost customers
: This is quite possibly the most tangible and most devastating of all losses. If your entire customer database is on a computer somewhere, how will you know who they are if that computer dies? You might actually "lose" your customers and never find them again. You could also lose customers who depend on data that is on one or more of your computers; if the customer finds out that you have lost his data, he will undoubtedly be less than impressed with you. The degree to which this data loss affects him may not even be relevant to him; he knows that you lost his data, and he might leave just because he no longer feels your company is competent.

Orders
: Whatever service or product your company provides, you have some way to keep track of requests for that product or service. Again, chances are that the method is computer-based. Data loss may mean several hours, days, or even weeks of lost orders. These may be orders that your salespeople worked very hard to get!

Morale
: Think about how you would feel if you were one of the salespeople whose orders were lost. You spent days or weeks working on sales, and now they're gone forever. Maybe you should go somewhere where your hard work doesn't go to waste. The better the salesperson, the better the chance that she may jump ship if you lose her sales. What about the average employee? If your computers have a reputation for going down and a reputation for losing data, it gives the employees a feeling of helplessness. Maybe they should go somewhere where they have the proper equipment to do their jobs.

Image

What about your standing in the industry? News of a major data loss undoubtedly spreads. This news may get to competitors, whom you can trust to use it against you at any opportunity. The news may also get to a regulatory agency that is in charge of your type of company. For example, if you work for a U.S. bank, it would be a terrible thing for the Office of the Comptroller of the Currency (OCC) to find out that you had a major data loss. They may decide to take a really close look at your affairs. Nobody wants that kind of attention!

Budget

It takes only one story of lost data to give your computer department an internal reputation for data loss. Try as you might to get rid of it, that reputation may stay for a while. You're only as good as your last restore. (A friend of mine said, "You're only as good as your *worst* restore.") If people don't trust your backups, they will duplicate your backup efforts. Employees will spend time and money backing up their systems locally. Each person may decide to buy his own backup drive and backup software or even to come up with his own in-house script. Their backups will be inefficient and costly at best, and may subject them to further data loss at worst. When everybody takes matters into their own hands, you can lose quite a bit of money in people-hours and extra hardware.

Time

How many people are supporting your computers? How much of their efforts will you lose if your development system loses data? I know of many companies that have numerous contract programmers writing code all the time. If the system that houses their work loses their code, how much money will you have wasted? In fact, no matter what department you look at, if they do their work on a computer, and you lose that data, you can lose considerable time and money.

What Will Downtime Cost You?

When planning your backup and recovery program, you may have several options that affect the speed of the recovery. The faster the recovery, the more the backup system will cost you. What you must ask yourself before deciding on these types of options is, "What will downtime cost?" When thinking about this, I'm reminded of a copier machine commercial from a few years ago: "When your copier goes down, do people just say, 'That's all right, we'll just use carbon paper!'?" If one of your main systems goes down, can your people continue working, or does your entire company come to a standstill? If it comes to a standstill, are your people salaried, so that sending them home saves you no money? Here are some additional costs to consider:

Customer perception

A customer hates to hear, "Please call back; our computers are down," or "Connection not responding." Depending on your type of business, they might just decide to go elsewhere. The longer your systems are down, the more customers will hear this message.

Employee perception
: Nobody wants to work at a company where the computers are always going down. The more your employees depend on your systems, the truer this becomes. If you were a salesperson who couldn't use your contact database for a day or so, how happy would you be?

Time
: Again, you lose time. You lose headway, and your salaried employees who depend on the system that is down are effectively being paid to do nothing.

Wax On, Wax Off: Finding a Balance

Using a system that has no backups is like driving a car 100 miles an hour down a busy road the day after your insurance policy expires. Likewise, having a three-node, highly available cluster for a noncritical application is like having full coverage on your 20-year-old fifth car. Just as insurance plans have different levels of coverage and riders to cover various types of damage, different backup methodologies provide different levels of recoverability.

That Was Close

One memorable moment was when we had a 600 GB file server that hadn't been properly backed up in a while. During a particularly hot weekend, both A/Cs handling the room failed, and temperatures soared. We shut everything down, waited for the A/Cs to be fixed, and started things back up after it cooled off a bit. Sure enough, two disks, physically next to each other in the same RAID4 array, failed. We were narrowly able to avoid total data loss by finding a spare disk and swapping control boards between it and one of the failed drives, which let it spin up and be accessed. We had the vendor courier us replacement drives the next day and then spent a lot of time fixing the backup server.

—Theo Van Dinter

Don't Go Overboard

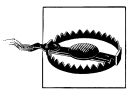
Not all environments need up-to-the-minute data recoverability. For many environments, recovering the systems up to last night's backups is acceptable. For some environments, recovering the system even up to last week or month is OK. Spending thousands of dollars and hundreds of hours implementing the greatest backup solution in the world is a waste if you don't need that level of coverage. This usually is not the problem for most sites; on the contrary, most sites don't spend nearly enough money or effort on their backup and recovery systems. In other cases, however, money may be wasted on unnecessarily elaborate systems.

Recoverability requirements also vary from machine to machine within the same company. The amount of work that would be lost, or the possibility of adversely affecting a customer, may determine these requirements. For example, it may be considered acceptable for an employee or two to lose a day's work spent on a few word processing documents. That is, unless it was the Senior Vice President's assistant who was working on the departmental budget, in which case your mileage may vary. And, it would probably be totally unacceptable for you to lose even one hour's worth of entries into a companywide sales database used by hundreds of people.

The point is that *your backup requirements are determined by your recovery requirements*. The difficulty comes in finding and using a tool capable of providing the level of recovery that you need. Consider users' home directories for a minute. If they are local to each user's workstation, a loss of one user's disk in the afternoon would mean that one user would lose a few hours of work. However, if user directories are located on an NFS file server that serves thousands of users, you could potentially lose several thousand hours of work if you use only traditional backup tools.

If the loss of a networked file server is unacceptable, you might want to consider *snapshot* technology. Snapshot software allows you to take a "picture" of your drive or filesystem at a single point in time and then use that picture to back up that drive or filesystem. If the backup references the drive or filesystem via this snapshot, it will back up a consistent picture of the drive or filesystem as it looked at the time the snapshot was taken. If this kind of functionality is interesting to you, you might consider reading Chapter 7, which describes emulating snapshot functionality with `rsync` and hard links.

Sometimes the tool you need comes with your operating system or database platform, but it's just not being used properly. Sometimes backup tools aren't being used at all. For example, if you have a production Oracle database, combining nightly hot backups with archived redo logs provides you up-to-the-minute recoverability. However, if you lose a disk that is part of a database that doesn't back up its transaction logs, you will lose all work since the last cold backup. See Part V for more information.



If you have a production instance of any kind and are not using the transaction logging feature of your database engine, turn on logging as soon as possible!

Therefore, while it is necessary to find the appropriate utility to give you the degree of recoverability that you require, it is also necessary to use it.

Get the Coverage That You Need

Some environments cannot afford even one minute of downtime, and they should pay for the best backup coverage—whatever it costs. This is because of the great loss that they will incur if they ever lose their systems for even a short period (I know of one company that claims that it loses over $1 million a minute when its systems are down). On the other hand, if you are in an environment that can afford downtime, then spending huge amounts of money for an immediately available *hot site*[*] is a complete waste of money.

Consider Table 1-1. No one should depend on a car, or a computer, without having at least the basic level of coverage. If the only car that you own is uninsured and a drunk driver runs into you and totals it, how would you recover from such a loss? Similarly, if your computer systems have critical information stored on them, how will you recover when a hard drive crashes and all that data is lost? What some people forget is that the opposite of this equation is true as well. If you have a third car that happens to be a 20-year-old (nonclassic) car, you will probably get only liability coverage on it; you could live without that car if it were destroyed today. Spending hundreds of extra dollars a year to insure a $50 car just doesn't make sense. Likewise, if the computers that you are managing are in an environment in which you can do without them for a few days, do you really need hot-swappable, mirrored drives? Pick an appropriate level of protection for your environment.

Table 1-1. Comparing automobile insurance and data protection

Types of coverage	Automobile insurance	Computer backups
Minimum coverage	Collision and liability (just keeps you from losing your shirt if you run into someone).	• Regular nightly backups (keeps you from losing your job when a disk drive dies)
Unexpected disasters	Comprehensive coverage (vandalism, acts of God, etc.).	• Journaling filesystems • Uninterruptible Power Supplies (UPSs)
Get me driving now	Rental car coverage (you get a car if your car is in the shop due to an accident).	• RAID • Mirroring • Using hot-swap drives • High-availability (HA) system

[*] A hot site is a place where you have computers standing by to do an immediate recovery of your environment.

Table 1-1. Comparing automobile insurance and data protection (continued)

Types of coverage	Automobile insurance	Computer backups
Major disasters	Another company will pick up your policy and replace your car if both your car and your insurance company are destroyed in an earthquake.	• Sending copies of your backup volumes to off-site storage, in case both your computer room and media library are destroyed • Sending your backups via a dedicated network to a large storage system at your off-site storage vendor
Maximum protection	The insurance company not only agrees to the conditions listed earlier, but also agrees to store another car of the same model in another state that you can use at any time if all cars in your state are destroyed.	• Real-time mirroring to a hot-swappable system at another of your sites • Sending your backups via either network or courier to a hot-site vendor

You need to balance the cost of a particular backup implementation against the projected monetary loss of the outage from which it protects you. For example, assume that you are evaluating two backup choices. The first option involves sending copies of your backup volumes to an off-site vendor for storage at a cost of $500 a month. The second option is an immediately available standby machine in another city that receives up-to-the-minute replication data from your production machine; let's say this option costs you $5,000 a month.

Your company is located in Utopia, where no natural disasters ever occur, your disks are all mirrored, and you have determined that a day's worth of downtime would cost only $500. Do you really want to spend $60,000 a year to protect against something that will probably never occur? If something catastrophic happened to your datacenter, wouldn't the day-old, off-site copies serve just as well? Your company would suffer an extra day or so of downtime, but you have already determined that this is affordable. The $6,000-a-year solution is probably much more appropriate for this environment.

However, are you protecting yourself from everything that you should be? Are you in an area that is prone to natural disasters and yet have no protection against that sort of event? Maybe you need to consider a different type of off-site storage. If you have a customer base that needs the data on your computers on a regular basis, have you provided for quick recovery in case of a failure? Perhaps you should be considering a hot site or multiple-site mirroring of your database servers. Table 1-1 provides a good overview of the various levels of coverage.

Why the Word "Volume" Instead of "Tape"?

Most backup utilities were originally written to back up to tape. Therefore, most books and online manuals talk about backing up to tape. However, many people are backing up to CDs, magneto-optical disks, or even disk drives. These media types have many advantages, because they act more like disk drives than tape drives. Random access of backup data is easier and you can read them using any block size you wish, because they do not record interrecord gaps like tape drives do.

Since many people no longer use tape, this book uses the more generic word *volume* whenever appropriate. You'll also find the term *backup drive* instead of *tape drive*. Again, that is because the backup drive could be a CD burner or a disk drive. The book uses the words *tape* and *tape drive* only when they are necessary and appropriate.

BackupCentral.com has a wiki page for every chapter in this book. Read or contribute updated information about this chapter at *http://www.backupcentral.com*.

CHAPTER 2

Backing It All Up

Now that the philosophy lessons of Chapter 1 are over, it's time to look at some of the important concepts behind backup and recovery, such as what to include, when to back it up, and more.

Don't Skip This Chapter!

The casual reader might assume that this chapter is an introduction to basic backup concepts. While that is, in fact, the purpose of this chapter, it is also true that many seasoned administrators are unfamiliar with the ideas presented here. One reason for this is that administrators find themselves constantly being pulled away from "mundane" activities like backups for things that are thought to be more "important," such as installing new servers and figuring out why systems are running slowly. Also, administrators may go several years without ever needing to perform a restore. The need to use your backups on a regular basis would undoubtedly change your ideas about their importance.

I wrote this book because backup and recovery has been my primary area of emphasis for several years, and I would like to share the lessons I've learned from this focused activity. This chapter provides an overview of how your backups should work. It also explains many basic yet extremely important concepts upon which any good backup plan should be based, and upon which all implementations discussed in this book are based.

The Impossible Job That No One Wants

Would anyone reading this book say that losing data is OK? I don't believe so. Then why do we treat backups so lightly? Sometimes I feel like Rodney Dangerfield when I'm arguing for better backups—"I tell ya, I don't get no respect, no respect." Backups often aren't considered during systems design. When a new server is purchased, does anyone ask for the impact on the current backup methodology? Some IT

departments do not even have control over the purchase of new systems, because they are sometimes bought by other cost centers. Have you ever tried to explain to another department manager why his terabyte-sized database server isn't going to get backed up to the standalone, gigabyte-sized tape drive that came with it?

Another often-overlooked issue is backup personnel. Have you ever tried to find the person in charge of backups? It's often an extra duty that gets passed around, in a manner similar to the way my sister and brother and I argued over whose turn it was to wash the dishes. If you are lucky enough to have a dedicated person, it's usually the most junior person in the company. I know, because that's how I got my first job. In fact, that's how many people get their first jobs. How can we give such low priority to something so important? Perhaps we should change that. Will one book change this long-standing hiring tradition? Probably not, but maybe it will help. At the very least, if the person in charge of backups has this book, that person has a complete guide to accomplishing the immense task that lies ahead.

What's the big deal, you say? With modern computer systems and reliable disk drives, why are backups still so important? Because computers still go down, that's why. Also, companies are placing more reliance than ever on computers functioning reliably. I don't care how good your Unix vendor is or how reliable your disk drives are or even if you have Dogbert himself as your network administrator, systems go down. Murphy's Law thrives in computer systems. Not only will your computer systems go down occasionally, but they will do so at the time most inconvenient to you and your customers. At that moment, and that moment will come, it is the job of the backup person to replace the data on the disk or disks that have stopped the show. "How long will it take?" is a typical question. The only acceptable response is "it's already done."

Who wants to be the person who messed up the restore and caused the customer database to be offline for three extra hours? Who wants to be the person who has to send a memo to the entire company saying that any purchase orders entered in the last two days have to be reentered? Who wants to be the person who has that in mind every day as they are checking the results of last night's backups? If you do your job well, and no data is lost, you are just doing what you're supposed to do. If you mess up, you're in big trouble. Who wants that job? No one, that's who.

You're reading this book because you've got the impossible job that nobody wants. Whether you've been doing it for a while or have just started down the backup road, you can see that the task that lies ahead is immense. The volume of data is tremendous, the nature of the data changes constantly, and the utilities at your disposal never seem to be up to the job. I know because I've been there. I've spent months trying to implement "solutions" from operating systems and database products that weren't ready. I've seen companies spend money on expensive commercial utilities, only to buy the wrong utility for their application. I've watched newer and bigger servers roll in the door without a single backup drive among them. I've also spent

long nights and weekends in computer rooms trying to recover data in a "reasonable" amount of time. Unfortunately, "reasonable" is defined by the end user who has no idea how difficult this job is.

There are now solutions to almost every backup problem out there. If you run a small shop with just a few systems, all of which run the same operating system, there's a solution for you. If you work in a huge shop with hundreds of boxes in the various flavors of Unix, Linux, Windows, and Mac OS, or just a few multiterabyte databases, there's a solution for you. The biggest part of the problem is misinformation. Most people simply do not know what is available, so they either suffer without a solution or settle for an inferior one—usually the one with the best salesperson. The six important questions that you have to continually ask yourself and others are why, what, when, where, who, and how:

Why?
: Why are you protecting yourself against disaster? Does it really matter if you lose data? What will the losses be? What different types of data do you have, and what is the value of each type?

What?
: What are you going to back up, the entire box or just selected drives or filesystems? What operating systems are you going to back up? What else, besides normal drives or filesystems, should be included in a backup?

When?
: When is the best time to back up your system? How often should you do a full backup? When should you do an incremental backup?

Where?
: Where will the backup occur? Where is the best place to store the backup volumes?

Who?
: Who is going to provide the hardware, software, and installation services to put this system together?

How?
: How are you going to accomplish it? There are a number of different ways to protect yourself against loss. Investigate the different methods, such as off-site storage, replication, mirroring, RAID, and the various levels of protection each provides. (Each of these topics is covered in detail in later sections of this book.)

Deciding Why You Are Backing Up

If you can't answer this question, there's really no point in moving forward. The good thing is that it is a really easy question to answer. Just think about all of the various things that can happen to your data, and then look at all of the types of data that you have. You should be familiar with each business unit that creates data, and how that business unit would be affected if that data was lost or damaged. All of this becomes your business justification for moving forward.

Deciding What to Back Up

Experience shows that one of the most common causes of data loss is that the lost data was never configured to be backed up. The decision of *what* to back up is an important one.

Plan for the Worst

When trying to decide what files to include in your backups, take the most pessimistic technical person in your company out to lunch. In fact, get a few of them together. Ask them to come up with scenarios that you should protect against. Use these scenarios in deciding what should be included, and they will help you plan the "how" section as well. Ask your guests: "What are the absolute worst scenarios that could cause data loss?" Here are some possible answers:

- An entire system catches fire and melts to the ground, leaving an unrecognizable mass of molten metal and blackened, smoking plastic.
- Since this machine was so important, you had it replicated to another node right next to it. Of course, that machine catches fire right along with this one.
- You have a centralized server that controls all backups and keeps a record of backup volume locations and what files are on what volumes, and so on. The server that blew up sits right next to this "backup server," and the intense heat takes this system with it.
- The disastrous chain reaction continues, taking out your DHCP and Active Directory servers, the NIS master server, the NFS and CIFS home directory servers, and the database server where you house the inventory of all your backup volumes with their respective locations. This computer also holds the telephone database listing all service agreements, vendor telephone numbers, and escalation procedures.
- You haven't memorized the number to your new off-site storage vendor yet, so it's taped to the wall next to your backup server. You realize, of course, that the flames just burned that paper beyond recognition.
- The flames set off the sprinkler system, and water pours all over your backup volumes. Man, are you having a bad day….

What do you do if one of these scenarios actually happens? Do you even know where to start? Do you know:

- What volume contains last night's backup?
- Where you stored it?
- How to get in touch with the off-site storage vendor to retrieve the copies of your backup volumes? And once you find that out, whether your server and network equipment will be available to recover?

- Who to call to get replacement equipment at 2:00 a.m. on a Saturday?
- What the network looked like before all the wires melted?

First, you need to recover your backup server, because it has all the information you need. OK, so now you found the backup company's card in your wallet, and you've pulled back every volume they had. Since your media database is lost, how will you know which one has last night's backup on it? Time is wasting.…

All right, you've combed through all the volumes, and you've found the one you need to restore the backup server (easier said than done!). Through your skill, cunning, and plenty of help from tech support, you restore the thing. It's up and running. Now, how many disks were on the systems that blew up? What models were they? How were they partitioned? Weren't some of them striped together into bigger volumes, and weren't some of them mirroring one another? Where's that information stored? Do you even know how big the drives or filesystems were? Man, this is getting complicated.…

Validated, My Eye

A biotech firm with a number of servers that were considered validated systems for FDA CFR21 purposes lost a critical database that was running on one such server. When they went to their backup server to restore it, they discovered to their horror that that server had not been backed up for approximately three months. Somehow, it had been removed from the backup schedule, so no "errors" were showing up, and they were now without anything remotely approaching a current backup. The problem escalated up to the CEO of the company.

—Jim Damoulakis

Didn't you just install that big jumbo kernel patch last week on three of these systems? (You know, the one that stopped all those network broadcast storms that kept bringing your network down in the middle of the day.) You did make a backup of the kernel after you did that, didn't you? Of course, the patch also updated files all over the OS drive. You made a full backup, didn't you? How will you restore the operating system drive, anyway? Are you really going to go through the process of reinstalling the operating system just so you can run the `restore` command and overwrite it again?

Filesystems aren't picky about size, as long as you make them big enough to hold the data that you restore to them, so it's not too hard to get those filesystems up and running. But what about the database? It was using raw partitions. You know it's going to be much pickier. It's going to want */dev/rdsk/c7t3d0s7, /dev/dsk/c8t3d0s7*, and */dev/dsk/c8t4d0s7* right where they were and partitioned just as they were before

the disaster. They also need to be owned by the database user. Do you know which drives were owned by that user before the crash? Which disks were those again?

It could happen.

Part IV covers these Catch-22 situations.

Take an Inventory

Make sure you can access essential information in the event of a disaster:

Backups for your backups
: Many companies have begun to centralize control of their backups, which I think is a good thing. However, once you centralize storage of all your backup information, you have a single point of failure for your entire backup plan. You can't restore the backup server because you don't have the database of your backups. You don't have the database of your backups because you need to restore your backup server. Restoring this server would be the first step in any multisystem outage. For things like media inventory, don't underestimate the value of an inventory printed on paper and stored off-site. That paper may just get you out of this Catch-22. Given the single-point-of-failure factor, the recovery of your backup server should be the easiest and best-documented recovery that you have. You may even want to investigate creating a special `tar`, `ntbackup`, or `rsync` backup of that data to make it even easier to recover during a disaster.

What peripheral devices did you have?
: Assuming you back up your disk drive configuration on a regular basis, you might have a list of all the disk drives, but do you know what models they are? If you have all Brand-X 500 GB drives, you have no problem, but many servers have a mixture of drives that were installed over time. You may have a collection of 40 GB, 100 GB, and 500 GB drives, all on the same system. Make sure that you are recording this in some way. Unix and Mac OS systems record this information in the *messages* file, and Windows stores it in the registry, so hopefully you're backing *those* up.

How were they partitioned?
: This one can *really* get you, especially if you have to restore the operating system drive or a database drive. Both drives are typically partitioned with custom partitions that must be repartitioned exactly the same as before for a proper restore to occur. Typically, this partition information is not saved anywhere on the system, so you must do something special to record it. On a Solaris system, for example, you can run a `prtvtoc` on each drive and save that to a file. Search on the Internet for scripts for capturing this information; a number of such free utilities exist.

How were your volume managers configured?

A number of operating system-specific volume managers are available, including Veritas Volume Manager, Windows Dynamic Drives, Solstice (Online) Disk Suite, and HP's Logical Volume Manager. How is yours configured? What devices are mirrored to what? How are your multidisk devices set up? Unbelievably, this information is not always captured by normal backup utilities. In fact, I used Logical Volume Manager for months before hearing about the `lvmcfgbackup` command (it backs up the LVM's configuration information). Sometimes if you have this properly documented, you may not need to restore at all. For example, if the operating system disk crashes, simply put the disks back the way they were and then rebuild the stripe in the same order, and the data should be intact. I've done this several times.

How are your databases set up?

I have seen many database outages. When I ask a database administrator (DBA) how her database is set up, the answer is almost always, "I'm not sure…." Find out this information, and record it up front.

Did you document how you set up DHCP, Active Directory, NFS, and CIFS?

Document, document, document! There are a hundred reasons to properly document things like this, and recovery from a disaster is one of them. Good documentation is definitely part of the backup plan. It should be regularly updated and available. No one should be standing around saying "I haven't set up NIS/AD/NFS from scratch in years. How do you do that again? Has anyone seen my copy of O'Reilly's book?" Actually, the best way to do this is to automate the creation of new servers. If your operating system supports it, take the time to write scripts that automatically install various services, and configure them for your environment. Put these together in a toolkit that is run every time you create a new server. Better yet, see if your OS vendor has any products that automate new server installations, such as Sun's Jumpstart, HP's Ignite-UX, Linux Kickstart, and Mac OS cloning features.

Do you have a plan for this?

The reason for describing the earlier horrible scenarios is so you can start planning for them now. Don't wait until there's 20 feet of snow in your front yard before you start shopping for a snow shovel! It's going to snow; it's only a question of when. Take those pessimists out to lunch, let them dream of the worst things that could happen, and then plan for them. Have a fully documented, step-by-step plan for the end of the computer world as you know it. Even if the plan needs a little modification when you actually have to use it, you will be glad you have a starting point. It's a whole lot better than standing around saying, "What do we do now? Has anyone seen my résumé?" (You did keep a hardcopy of it, right?)

Know what's on your boxes!

The best insurance against almost any kind of loss is for the backup/recovery person to be familiar with the systems he is protecting. If a particular server goes down, you should know immediately that it contains an Oracle or SQL Server database and should be running for those volumes. That way, the moment the server is ready for a restore, so are you. Become very involved in the installation of any new system or database. You should know what database platforms you are using and how they are set up. You should know about any new drives, file-systems, databases, or systems. You need to be very familiar with every box, what it does, and what's on it. This information is vital so that you can include any special backups for that type of system.

It Pays to Watch Your Logs

It was my very first gig out of college, so I was primarily supposed to be doing desktop support while learning at the feet of a high-priced Unix consultant, who we'll call *Fred*.

We were supporting a `ForEx` trading app called `Opus` that ran on SunOS. When it stored trades, half the information was in the path. For example, if someone made a USD-GBP trade on June 15 with someone from Bank of New York, the path and file would look like this:

```
/opt/app/opus/transactions/portfolio/third-party/...etc...etc.../USD/CAI/GBP/
BONY/ask/19970615120453.2372149821335
```

This insipid design was surely not Fred's fault, but he did set up the backups for it. I discovered that he had set up a `tar` job using `-v`, which produced logs so big he wasn't looking at them. Once I removed `-v` and started watching the logs, I found out that backups had been failing. The version of `tar` that shipped with SunOS at the time choked on file paths longer than 100 characters or so. The trading stubs were all about nine characters too long. Fred was basically tarring up a huge directory tree with no files at the bottom. Had he ever looked at the logs, he would have known that.

I was designated the primary Unix admin the next day. The company didn't renew Fred's contract.

—Jim "Sparky" Donnellan

Are You Backing Up What You Think You're Backing Up?

I remember an administrator at one of my previous employers who used to say, "Are we getting this on tape?" He always said it with his trademark smirk, and it was his way of saying "Hi" to the backup guy. His question makes a point. There are some global ways that you can approach backups that may drastically improve their effectiveness. Before we examine whether to back up part or all of the system, let us examine the common practice of using include lists and why they are dangerous. Also, let's consider some of the ways that you can avoid using include lists.

What are include and exclude lists? Generically speaking, there are two ways to back up a system:

- You can tell your backup system to back up everything, except what is in an *exclude list,* for example:
  - For Unix, Linux, and Mac OS servers:

```
Include: *
Exclude: /tmp, /junk1, /junk2
```

  - For Windows servers:

```
Include: *
Exclude: *.tmp, *Temporary Internet Files*, ~*.*, *.mp3
```

- You can tell your backup system to back up what is in an *include list*, for example:
  - For Unix, Linux, and Mac OS servers:

```
Include: /data1, /data2, /data3
```

  - For Windows servers:

```
Include: D:\, E:\
```

Looking at these examples, ask yourself what happens when you create */data4* or the *F:* drive? Someone has to remember to add it to the include list, or it will not be backed up. This is a recipe for disaster. Unless you're the only one who adds drives or filesystems and you have perfect memory, there will always be a forgotten drive or filesystem. As long as there are other administrators and there is gray matter in your head, something will be left out.

I Hate It When That Happens

I was working at a major publishing company when an image server died. When those involved went to the backup administrator and asked for a restore of all the images from the server, he had no record of the server. It appears that after placing the server into production a year earlier, no one had formally requested that the server be added to the backup system. They lost thousands of images.

—Chris Pritchard

However, unless your backup utility supports automated drive or filesystem discovery, it takes a little effort to say, "Back up everything." How do you make the list of what systems, drives, filesystems, and databases to back up? What you need to do is look at files such as */etc/vfstab* or the Windows registry and parse out a list of drives or filesystems to back up. You can then use exclude lists to exclude any drives or filesystems you don't want backed up.

Oracle has a similar file in Unix, called *oratab*, which can be used to list all Oracle instances on your server.* Windows stores this information in the registry, of course. You can use *oratab* to list all instances that need backing up. Unfortunately, Informix and Sybase databases have no such file unless you manually make one. I do recommend making such a file for many reasons. It is much easier to standardize system startup and backups when you have such a file. If you design your startup scripts so that a database does not get started unless it is in this file, you can be reasonably sure that any databases that anyone cares about will be in this file. This means, of course, that any important databases are backed up without any manual intervention from you. It also means that you can use the same Informix and Sybase startup scripts on every system, instead of having to hardcode each database's name into the startup scripts.

How do you know what systems to back up? Although I never got around to it, one of the scripts I always wanted to write was a script that monitored the various host databases, looking for new systems. I wanted to get a complete list of all hosts from Domain Name System (DNS) and compare it against a master list. Once I found a new IP address, I would try to determine if the new IP address was alive. If it was alive, that would mean that there was a new host that possibly needed backing up. This would be an invaluable script; it would ensure there aren't any new systems on the network that the backups don't know about. Once you found a new IP address, you could use `nmap` to find out what type of system it is. `nmap` sends a malformed TCP packet to the IP address, and the address's response to that packet reveals which operating system it is based on.

Some commercial data protection management software packages now support this functionality.

Back Up All or Part of the System?

Assuming you've covered things that are not covered by normal system backups, you are now in a position to decide whether you are going to back up your entire systems or just selected drives or filesystems from each system. These are definitely two different schools of thought. As far as I'm concerned, there are too many gotchas in the selected-filesystem option. Backing up everything is easier and safer than backing up from a list. You will find that most books stop right there and say "It's best to back up everything, but most people do something else." You will not see those words here. I think that not backing up everything is very dangerous. Consider the following comparison between the two methods.

* You can install an Oracle instance without putting it in this file. However, that instance will not get started when the system reboots. This usually means that the DBA will take the time to put it in this file. More on that in Chapter 15.

Backing up only selected drives or filesystems

Here are the arguments for and against selective backups.

Save media space and network traffic. The first argument that is typically stated as a plus to the selected-filesystem method is that you back up less data. People of this school recommend having two groups of backups: operating system data and regular data. The idea is that the operating system backups would be performed less often. Some would even recommend that they be performed only when you have a significant change, such as Windows security patches, an operating system upgrade, a patch installation, or a kernel rebuild. You would then back up your "regular" data daily.

The first problem with this argument is that it is outdated; just look at the size of the typical modern system. The operating system/data ratio is now significantly heavier on the data side. You won't be saving much space or network traffic by not backing up the OS even on your full backups. When you consider incremental backups, the ratio gets even smaller. Operating system partitions have almost nothing of size that would be included in an incremental backup, unless it's something important that should be backed up! This includes Unix, Linux, and Mac OS files such as */etc/passwd*, */etc/hosts*, *syslog*, */var/adm/messages*, and any other files that would be helpful if you lost the operating system. It also includes the Windows registry. Filesystem swap is arguably the only completely worthless information that could be included on the OS disk, and it can be excluded with proper use of an exclude list.

Harder to administer. Proponents of piecemeal backup would say that you can include important files such as the preceding ones in a special backup. The problem with that is it is so much more difficult than backing up everything. Assuming you exclude configuration files from most backups, you have to remember to do manual backups every time you change a configuration file or database. That means you have to do something *special* when you make a change. *Special is bad.* If you just back up everything, you can administer systems as you need to, without having to remember to back up before you change something.

Easier to split up between volumes. One of the very few things that could be considered a plus is that if you split up your drives or filesystems into multiple backups, it is easier to split them between multiple volumes. If a backup of your system does not fit on one volume, it is easier to automate it by splitting it into two different include lists. However, in order to take advantage of this, you have to use include lists rather than exclude lists, and then you are subject to the limitations discussed earlier. You should investigate whether your backup utility has a better way to solve this problem.

Easier to write a script to do it than to parse out the fstab, oratab, or Windows registry. This one is hard to argue against. However, if you do take the time to do it right the first time, you never need to mess with include lists again. This reminds me of another favorite phrase of mine: "Never time to do it right, always time to do it over." Take the time to do it right the first time.

The worst that happens? You overlook something! In this scenario, the biggest benefits are that you save some time spent scripting up front, as well as a few bytes of network traffic. The worst possible side effect is that you overlook the drive or filesystem with your boss's budget that just got deleted.

Backing up the entire system

The pros for backing up the entire system are briefer yet far more compelling:

Complete automation. Once you go through the trouble of creating a script or program that works, you just need to monitor its logs. You can rest easy at night knowing that all your data is being backed up.

The worst that happens? You lose a friend in the network department. You may increase your network traffic by a few percentage points, and the people looking after the wires might not like that. (That is, of course, until you restore the server where they keep their DNS source database.)

Backing up selected drives or filesystems is one of the most common mistakes that I find when evaluating a backup configuration. It is a very easy trap to fall into because of the time it saves you up front. Until you've been bitten though, you may not know how much danger you are in. If your backup setup uses include lists, I hope that this discussion convinces you to rethink that decision.

Deciding When to Back Up

This might appear to be the most straightforward topic. Everybody backs up their system every night, right? What's the big deal? Actually, this could more aptly be titled "What levels do I run when?" It's always a big question. How often do you run a full backup? How often do you run incremental backups? Do you run various levels of incrementals that back up just today's changes or continuous incremental backups that back up everything since the last full backup? Everyone has her own answers to these questions. The only thing that is a definite is that there should be *at least* some level of backup every night. Before any further discussion on the topic, let's define some terms.

Backup Levels

The following are various backup levels. These terms are not used the same way by everyone.

Full/Level 0
: A full backup.

Level 1
: An incremental backup that backs up everything that has changed since the last level 0 backup. Repeated level 1 backups still back up everything since the last full/level 0 backup.

Levels 2–9
: Each level backs up whatever has changed since the last backup of the next-lowest level. That is, a level 2 backs up everything that changed since a level 1, or since a level 0, if there is no level 1. With some products, repeated level 9 backups back up only things that have changed since the last level 9 backup, but this is far from universal.

Incremental
: Usually, a backup backs up anything that has changed since the last backup of any type.

Differential
: Most people refer to a differential as a backup that backs up everything that has changed since the last full backup, but this is not universal. In Windows, a differential is a backup that does not clear the archive bit. Therefore, if you run a full backup followed by several differential backups, they act like differential backups in the traditional sense. However, if you run even one incremental backup in Windows, it clears the archive bit, and the next differential backup backs up only those files that have changed since the last incremental backup. That's why a differential backup is not synonymous with a level 1 backup.

Cumulative incremental
: I prefer this term to differential, and it refers to a backup that backs up all files that have changed since the last full backup.

Backup products and backup administrators do not agree on these definitions. Make sure you know what your product means when it uses one of these terms!

A question that I am often asked is, "You want me to back up every night?" What the question really means is, "Even on the weekend?" Nobody's working on the weekend, right? *Right*…except for your noisiest customer last weekend. You know the customer I'm talking about: the one who calls your boss instead of the help desk when there's a problem. And if your boss isn't in or doesn't fix the problem fast

The Windows Archive Bit Is Evil!

The Windows archive bit is evil and must be stopped. At the very least, backup vendors should give us the option of not using it—without penalty. If the "ready for archiving" bit is set on a file in Windows, it indicates that a file is new or changed, and that it should be backed up in an incremental backup. Once a file is backed up, the archive bit is cleared. Therefore, the first problem with the archive bit is that it should be called the *backup bit*; backups are not archives.

The biggest problem with the archive bit, however, is that the process assumes only one application will clear the archive bit, when there could actually be several of them. The first backup program to back up the directory clears the archive bit, and the next program does not back up the same files. Suppose a user decides to use `ntbackup` to back up to CD his files that are on the company's file server. If he does that, `ntbackup` clears the archive bit, and the corporate backup system in charge of backing up those files will not back them up when it does an incremental backup. They don't appear to be in need of backup because the archive bit is not set. This means that any user can defeat the purpose of the entire backup system.

Proponents of the archive bit point out that the archive bit is set on newly installed software, even if the files are old. A backup software package that uses only modification time does not "notice" these files if they're older than the latest incremental backup, so perhaps what they should be using is a combination of the archive bit and modification time. If either has been changed, the file should be included in an incremental backup.

When backing up Unix systems, there is no archive bit, so backup applications use either `mtime` (when the contents of the file were last changed) or `ctime` (when the attributes of the file were last changed). When backing up Windows systems, different backup applications use the archive bit differently. Some use it in conjunction with `mtime` and `ctime`. Some use only the archive bit, and others do not use it at all. (Based on what I'm saying about the archive bit, that might not be a bad thing.)

Microsoft has offered an alternative to the archive bit with the *change journal*, available in Windows 2000 and later. Backup products that support the change journal can consult it to determine which files have changed instead of looking at the archive bit. The change journal is not enabled by default, but it can be enabled using the `fsutil usn createjournal` command. You need to specify a `MaximumSize` that's big enough to hold all the changes that are made in between backups. Since 30 or 40 changes are stored in a single 4 K record, you can store 500,000 changes in a 75 MB journal. (If the change journal isn't large enough, the oldest changes are deleted from the beginning of the log to make room, so it's important to make the log large enough.) I suggest you find out the largest number of files you've ever had on an incremental backup and then make the log twice that size. The additional integrity this brings to your backup system more than makes up for the space this journal takes up.

enough, this customer will call your boss's boss. Well, last weekend this customer was really behind, so she spent the entire weekend at work, working around the clock on next year's budget. She finally got it straightened out at about 1:00 a.m. Monday. At around 4:00 a.m., the disk where her home directory resides stopped working. (Everything dies Monday morning, doesn't it?) You haven't run a backup since Friday night. Your phone is ringing, and it's your boss. Any guesses as to what he wants to talk to you about? Do *you* want to be the one to tell this customer that you could have saved her file, but you don't run backups on the weekend?

Which Levels Do You Run and When?

There are several schools of thought on this question. The following are some suggested backup schedules.

Weekly schedule: All full/level 0 backups

Table 2-1 contains a backup schedule for the paranoid (not that paranoid is a bad thing). Performing a level 0 backup every day onto a separate volume. (Please don't overwrite yesterday's good level 0 backup with today's possibly corrupt level 0 backup!) If your system is really small, this schedule might work for you. If you have systems of any reasonable size, though, this schedule is not very scalable. It's also really not that necessary with today's commercial backup software systems.

Table 2-1. All full backups

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Full/0	Full/0	Full/0	Full/0	Full/0	Full/0	Full/0

Weekly schedule: Weekly full, daily level differentials/level 1s

The advantage to the schedule in Table 2-2 is that throughout most of the week, you would only need to restore from two volumes—the level 0 and the most recent level differential/level 1. This is because each differential/level 1 backs up all changes since the full backup on Sunday. Another advantage of this type of setup is that you get multiple copies of files that are changed early in the week. This is probably the best schedule to use if you are using simple utilities such as `dump`, `tar`, or `cpio` because they require you to do all the volume management. A two-volume restore is *much easier* than a six-volume restore—trust me!

Table 2-2. Weekly full backups, daily level differentials/level 1s

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Full/0	Diff/1	Diff/1	Diff/1	Diff/1	Diff/1	Diff/1

Weekly schedule: Weekly full, daily leveled backups

If your backup product supports multiple levels, you can use the schedule shown in Table 2-3. The advantage to this schedule is that it takes less time and uses less media than the preceding schedule. There are two disadvantages to this plan. First, each changed file gets backed up only once, which leaves you very susceptible to data loss if you have any media failures. Second, you would need six volumes to do a full restore on Friday. If you're using a good open-source backup utility or commercial backup utility, though the latter is really not a problem, because these utilities do all the volume management for you, including swapping tapes with an auto-changer.

Table 2-3. Weekly full backups, daily leveled backups

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Full/0	1	2	3	4	5	6

Weekly schedule: Monthly full, daily Tower of Hanoi incrementals

One of the most interesting ideas that I've seen is called the Tower of Hanoi (TOH) backup plan. It's based on an ancient mathematical progression puzzle by the same name. The game consists of three pegs and a number of different-sized rings inserted onto those pegs. A ring may not be placed on top of a ring with a smaller radius. The goal of the game is to move all of the rings from the first peg to the third peg, using the second peg for temporary storage when needed.*

A goal of most backup schedules is to put changed files on more than one volume while reducing total volume usage. The TOH accomplishes this better than any other schedule. If you use a TOH progression for your backup levels, most changed files are backed up twice—but only twice. Here are two different versions of the progression (they're related to the number of rings on the three pegs, by the way):

```
0 3 2 5 4 7 6 9 8 9
0 3 2 4 3 5 4 6 5 7 6 8 7 9 8
```

These mathematical progressions are actually pretty easy. Each consists of two interleaved series of numbers (e.g., 2 3 4 5 6 7 8 9 interleaved with 3 4 5 6 7 8 9). Table 2-4 uses a schedule to illustrate how this works.

Table 2-4. Basic Tower of Hanoi schedule

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
0	3	2	5	4	7	6

* For a complete history of the game and a URL where you can play it on the Web, see *http://www.math.toronto.edu/mathnet/games/towers.html*.

It starts with a level (full) on Sunday. Suppose that a file is changed on Monday. The level 3 on Monday would back up everything since the level 0, so that changed file would be included on Monday's backup. Suppose that on Tuesday we change another file. Then on Tuesday night, the level 2 backup must look for a level that is lower, right? The level 3 on Monday is not lower, so it references the level 0 also. So the file that was changed on Monday, as well as the file that was changed on Tuesday, is backed up again. On Wednesday, the level 5 backs up only what changed that day, because it references the level 2 on Tuesday. But on Thursday, the level 4 does not reference the level 5 on Wednesday; it references the level 2 on Tuesday.

Note that the file that changed on Tuesday was backed up only once. To get around this problem, we use a modified TOH progression, dropping down to a level 1 backup each week, as shown in Table 2-5.

Table 2-5. Monthly Tower of Hanoi schedule

Day of the week	Week one	Week two	Week three	Week four
Sunday	0	1	1	1
Monday	3	3	3	3
Tuesday	2	2	2	2
Wednesday	5	5	5	5
Thursday	4	4	4	4
Friday	7	7	7	7
Saturday	6	6	6	6

If it doesn't confuse you and your backup methodology,* and if your backup system supports it, I recommend the schedule depicted in Table 2-5. Each Sunday, you get a complete incremental backup of everything that has changed since the monthly full backup. During the rest of the week, every changed file is backed up twice—except for Wednesday's files. This protects you from media failure better than any of the schedules mentioned previously. You will need more than one volume to do a full restore, of course, but this is not a problem if you have a sophisticated backup utility with volume management.

"In the Middle of the Night…"

This phrase from a Billy Joel song indicates the usual best time to do backups. Backups should be scheduled in such a way that they do not run during normal business hours. Sometimes you cannot avoid it, but it should not be a regular occurrence. There are two main reasons for this:

* This is always the case for any recommendation in this book. If it confuses you or your backup methodology, it's not good! If your backups confuse you, you don't even want to try to restore! Always keep it simple, system administrator (K.I.S.S.).

Integrity

Unless you work in a 24/7 shop, nighttime is the time when the files are the most stable. (Of course, there could be batch jobs running that are manipulating data and customers accessing your web site, so not all files will be stable.) If you are backing up during the day, files are changing and probably also are open. Open files are more difficult to back up. Some backup packages handle open files better than others, but some cannot back them up at all. Also, if the file is changing throughout the day, you will not be sure what version you actually get on your backup.

Speed

Another reason for not doing backups during the day is that the network is much busier, hence slower, during the day. The throughput of your backups slows significantly when your network is being used for normal traffic. If this is a problem at night as well, you might consider using a special network just for your backups. Doing backups during the day can significantly affect the speed of your other applications, and it is not good practice to regularly slow down your systems while people are using them.

Of course, in today's global and Internet economy, "night" is relative. If you are in a shop in which the systems are accessed 24/7, you have to do things quite differently. You may want to look at Chapter 8 to see what vendors are doing to help meet this type of challenge.

It Does Have a Happy Ending (Almost)

I never had a server melt down to nothing, but I have lost entire servers, configuration and all. Luckily, I had their configurations saved. I also remember a time when we lost the server that contained our Informix database that listed all data about our volumes and their locations. I remember saying, "How do I get out of this one?" Luckily, we had a practice of sending a printout of each day's volumes along with our off-site shipment. I asked for all of that day's volumes, along with that printout. *Whew!*

What's that? You think that I'm a mean and vicious person who is out to give you nightmares for the next week? You have no idea how you would get that information if you needed it? You say that you're going to lose sleep for a while? Good! Better to have lost sleep than to have lost data. One of the main purposes of this book is to scare you. A complacent person in charge of backups is a dangerous thing. The preceding scenario includes several Catch-22 situations and wipes out data that is not normally caught by standard backups.

Deciding How to Back Up

Once you've decided *when* you're going to back up, you have to decide *how* you are going to back up the data. But first, look at what types of problems you are protecting yourself from.

Be Ready for Anything: 10 Types of Disasters

As stated earlier, how you want to do your restores determines how you want to do your backups. One of the questions that you must ask yourself is, "What are you going to protect yourself from?" Are the users in your environment all "power users" who use their computers intelligently and never make dumb mistakes? Would your company lose a lot of essential data if the files on your users' PCs were accidentally deleted? If a hurricane took out your whole company, would it be able to continue doing business? Make sure that you are aware of all the potential causes for data loss, and then make sure your backup methods are prepared for all of them. The most exhaustive list of potential causes of data loss that I have seen is in another O'Reilly book called *Practical Unix and Internet Security* by Simson Garfinkel and Gene Spafford. Their list, with my comments attached, follows:

User error
: This has been, by far, the cause of the biggest percentage of restores in every environment that I have seen. "Hey, I was sklocking my flambality file, and I accidentally pressed the jankle button. Can you restore it, *please*?" This one is pretty easy, right? What about the common question: "Can you restore it as of about an hour ago?" You can do this with continuous data protection systems and snapshots, but not if you're running backups once a night.

System-staff error
: This is less common than user error (unless your users have root or administrator privileges), but when it happens, oh boy, does it happen! What happens when you `newfs` your database's raw device or delete a user's document folder? These restores need to go *really fast*, because they're your fault. As far as protecting yourself from this type of error, the same is true here as for user errors: either typical nightly backups or snapshots can protect you from this .

Hardware failure
: Most books talk about protecting yourself from hardware failure, but they usually don't mention that hardware failure can come in two forms: disk drive failure and systemwide failure. It is important to mention this because it takes two entirely different methods to protect yourself from these failures. Many people do not take this into consideration when planning their data protection plan. For example, I have often heard the phrase, "I thought that disk was mirrored!"

when a drive or filesystem is corrupted by a system panic. Mirroring does not protect you from a systemwide failure. As a friend used to say, if the loose electrons floating around your system decide to corrupt a drive or filesystem when your system goes down, "mirroring only makes the corruption more efficient." Neither do snapshots protect you from hardware failure—unless you have the snapshot on a backup volume.

Disk drive failure

Protecting your systems from disk drive failure is relatively simple now. Your only decision is how safe you want to be. Mirroring, often referred to as RAID 1, offers the best protection, but it doubles the cost of your initial drive and controller hardware investment. That is why most people choose one of the other levels of Redundant Arrays of Independent Disks (RAID), the most popular being RAID 5, with RAID 6 gaining ground. RAID 5 volumes protect against the loss of a single drive by calculating and storing parity information on each drive. RAID 6 adds more protection by storing parity twice, thus allowing for the failure of more than one drive.

Systemwide failure

Most of the protection against systemwide failure comes from good system administration procedures. Document your systems properly. Use your system logs and any other monitoring methods you have at your disposal to watch your systems closely. Respond to messages about bad disks, controllers, CPUs, and memory. Warnings about hardware failures are your chance to correct problems before they cause major disasters. Another method of protecting yourself is to use a journaling filesystem. Journaling treats the filesystem much like a database, keeping track of committed and partially committed writes to the filesystem. When a system is coming up, a journaling filesystem can roll back partially committed writes, thus "uncorrupting" the filesystem.

The Windows change journal does not make NTFS a journaling filesystem in this sense. It contains only a list of files that have been changed; it does not actually contain the changes. Therefore, it cannot roll back any changes.

Software failure

Protecting yourself from software failure can be difficult. Operating system bugs, database bugs, and system management software bugs can all cause data loss. Once again, the degree to which you protect yourself from these types of failures depends on which type of backups you use. Frequent snapshots or continuous data protection systems are the only way to truly protect against losing data, possibly a lot of data, from software failure.

Electronic break-ins, vandalism, and theft

There have been numerous incidents of this in the past few years, and many have made national news. If you do lose data due to any one of these, it's very different from other types of data loss. While you may recover the data, you can never be sure of what happened to the data while it wasn't in your possession. Therefore, you need to do everything you can to ensure that this never happens. If you want to protect yourself from losing data in this manner, I highly recommend reading the book from which I borrowed this list, *Practical Unix and Internet Security*, by Simson Garfinkel and Gene Spafford (O'Reilly).

Natural disasters

Are you prepared for a hurricane, tornado, earthquake, or flood? If not, you're not alone. Imagine that your entire state was wiped out. If you are using off-site storage, is that facility close to you? Is it prepared to handle whatever type of natural disasters occur in your area? For example, if your office is in a flood zone, does your data storage company store your backups on the first floor? If they're in the flood zone as well, your data can be lost in one good rain. If you really want to ensure yourself against a major natural disaster, you should explore real-time, off-site storage at a remote location, discussed later in this chapter in the section "Off-Site Storage."

Other disasters

I remember how we used to test our disaster recovery plan at one company where I worked: we would pretend that some sort of truck blew up on the street that ran by our data center. The plan was to recover to an alternate building. This would mean that we would have to have off-site storage of media and an alternate site that was prepared to accommodate all our systems. A good way to do this is to separate your production and development systems and place them in different buildings. The development systems can then take the production systems' place if the production systems are damaged, or if power to the production building is interrupted.

Archival information

It is a terrible thing to realize that a rarely used but very important file is missing. It is even more terrible indeed to find out that it has been gone longer than your retention cycle. For example, you keep your backups for only three months, after which you reuse the oldest volume, overwriting any backups that are on that volume. If that is the case, any files that have been missing for more than three months are *impossible* to recover. No matter how insistent the user is about how important the files are, no matter how many calls he makes to your supervisors, you will *never* be able to restore the files. That is why you should keep some of your backups a little bit longer. A normal practice is to set aside one full backup each month for a few years. If you're going to keep these backups for a long time, make sure you read the following sidebar "Are You Keeping Your Archives Too Long?" and the "Backup and Archive" section in Chapter 24.

"How Were the Backups Last Night?"

I suppose I've heard thousands of administrator-error horror stories, like people typing `rm -r /*`. I remember a guy who wanted to delete a junk file in */bin* called *?*&(&^JI($SF))FS%$#T*, or something like that. He typed `rm /bin/?*` (which deleted all the files starting with any character—that's right—all of them). But there's one story that I witnessed firsthand that still makes me laugh.

A consultant was given the task of cleaning up our home directories. Apparently, my company was very good about deleting logins for people who had left the company, but we weren't very good about deleting their home directories. The consultant wrote a program that basically did the following:

1. `cd` into */home1*
2. `find`, looking for directories that did not match an entry in the password file and were not owned by root or administrator
3. `rm -r` that directory

Each user's home directory was located under a directory that was the first letter of her login. For example, the home directory for *cpreston* was in */home1/c/cpreston*. The scenario went something like this. The idea was that */home1/c* would be owned by root and thus would not be deleted. Unfortunately, over the years, an administrator or two would `cd` into */home1/c/cpreston* and try to correct an ownership problem. To do that, the administrator would type `chown cpreston .*`. Well, if you've ever done that as root, you know that .* includes **..**, which in this case would be */home1/c*. Thus, the */home1/c* ends up being owned by me!

The consultant did not foresee this and so would interpret */home1/c* as a user's home directory and look for the user called "c" in the password file. Of course, there was no such user, so the program said `rm -r /home1/c`. I'm not sure when my friend realized what was happening, but I do remember being on my way out the door and getting a weird phone call. "How were the backups of */home1* last night?" my friend asked—very sheepishly and very mysteriously. "Fine, as always," was my response, "Why?" There's something beautiful about the power that the backup guy yields at that magic moment when someone *really* needs some files restored. Up to that point, you're the guy who comes in early and stays late, watching the backup drives spin. In one moment, you're transformed into the most important person he knows! *Cool*.

Automate Your Backup

If you work in a shop with a modest budget, you probably looked at this heading and said, "Sure, if I could afford it." Although automation that involves expensive jukeboxes and autochangers is nice, that is not the type of automation I am talking about. There are two types of automation. One type allows your backups to complete an entire cycle without requiring any manual intervention from you, such as

ejecting and loading new volumes. This type of automation can make things much easier but can also make them much more expensive. If you can't afford it, a less expensive alternative is to have your backup system notify you when you need to do something manually. At the very least, it should notify you if you need (or forgot) to change a volume. If things aren't going right, you need to know. Too many times people look at their backup logs only when they need to do a restore. That's when they find out that their backups have failed for days or weeks. A slightly intelligent backup system could email you or page you if things don't go the way you expect them to go.

The second type of automation is actually much more important. This type of automation refers to how your backups "think." Your backup process should know what to back up without you telling it. If a DBA installs a new database, your backups should know about it. If a system administrator installs a new drive or filesystem, your backups should automatically include it. This is the type of automation that is essential to safe backups. A good backup system should not depend on a human brain to remember to do something.

Are You Keeping Your Archives Too Long?

Some governments have laws and regulations that govern how long certain types of data are allowed to be kept in a company's files. We're not talking about regulations that say you must keep data for a certain number of years. We're talking about a regulation that says you must *delete* data after a certain number of years. For example, you may be told that your personnel department can keep disciplinary paperwork for only two years. If an employee believes that her chances for a promotion are reduced because of a disciplinary action that is more than two years old, she can sue for damages. Many lawsuits have been filed based on these laws.

What happens when the disciplinary action "paperwork" is actually a file on someone's computer? The laws extend to the computers too, and the files must be deleted. But what if that file is on an archive volume that is being kept forever? Many companies have backup policies that dictate that one volume per system per year is kept "forever." In recent years, some companies have lost lawsuits because of policies like this.

The only way around this is to exclude from regular backups any directories that contain this type of information and archive them using a different schedule that conforms to the document retention laws of your state. I admit this is a pain. You will never read that I think that doing something special for anything is a good thing, but in these litigious times, this issue should not be overlooked.

Plan for Expansion

Another common problem happens as a backup system grows over time. What works for one or two boxes doesn't necessarily work for 200. As the volume of data grows, the need for a standardized backup system becomes greater and greater. This is a problem because most administrators, as they are writing their shell script to back up five or six boxes, do not think ahead to the time when there may be many more. I can remember my early days as the backup guy. I had 10 or 11 systems, and the "monster" was an Ultrix box. It was "huge," we said in those days. (It was almost 8 gigabytes!) The smallest tape drive we had was a 10 GB (with compression) Exabyte. We used the big 10 GB tape drive for the 8 GB system. We had what I considered to be a pretty good in-house backup script that worked without modification for two years.

Then came the HPs. The *smallest* system was 20 GB, and the biggest was much bigger than that. But these big systems came with a little 2 GB (4 with compression) DDS drive. Our backup script author never dreamed of a system that was bigger than a tape. One day I woke up, and our system was broken. I then spent months and months hacking up that shell script to support splitting the drive or filesystem into two tapes. Eventually, I gave up and bought a commercial product. My point is that if I had thought of that ahead of time, I might have been able to overcome the limitation without losing so much sleep.

When you are designing your backup system—or your data center, for that matter—plan on your systems getting bigger and more numerous. Plan for what you will do when that happens—trust me, it *will* happen. It will be much better for your mental health (not to mention your job security) if you can foresee the inevitable and plan for it when you design the system the first time. Your backup system is something that should be done right the first time. And if you spend a little time dreaming about how to break it *before* you design it, you can save yourself a lot of money in antacids and sleeping pills.

Don't Forget Unix mtime, atime, and ctime

Unix, Linux, and Mac OS systems record three different times for each file. The first is `mtime`, or modification time. The `mtime` value is changed whenever the contents of the file have changed, such as when you add lines to a logfile. The second is `atime`, or access time. The `atime` value is changed whenever the file is accessed, such as when a script is run or a document is read. The last is `ctime`, or change time. The `ctime` value is updated whenever the attributes of the file, such as its permissions or ownership, are changed.

Administrators use `ctime` to look for hackers because they may change permissions of a file to try to exploit your system. Administrators also monitor `atime` to look for large files that have not been accessed for a long time. (Such files can be archived and deleted.)

Backups change atime

You may be wondering what this has to do with backups. You need to understand that any backup utility that backs up using the filesystem modifies `atime` as it reads the file to back it up. Almost all commercial utilities, as well as `tar`, `cpio`, and `dd`,* have this feature. `dump` reads the filesystem via the raw device, so it does not change `atime`.

The atime can be reset—with a penalty

A backup program can look at a file's `atime` before it backs it up. After it backs up the file, the `atime` obviously has changed. It can then use the `utime` system call to reset `atime` to its original value. However, changing `atime` is considered an attribute change, which means that it changes `ctime`. This means that when you use a utility such as `cpio` or `gtar` that can reset `atime`, you change `ctime` on every file that it backs up. If you have a system that is watching for `ctime` changes, it will think that it's found a hacker for sure!

Make sure that you understand how your utility handles this issue.

Don't Forget ACLs

Windows files stored on an NTFS filesystem and some files stored on modern Linux filesystems use access control lists (ACLs) to grant or restrict permissions to users. ACLs say who can read, write, execute, modify, or have full control over a file. Figure 2-1 shows an example of such ACLs.

You need to investigate how your backup product is handling ACLs. The proper answer is that they are backed up and restored. This is a feature common with commercial products, but unfortunately, not all open-source products do this. Make sure you look into this when evaluating open-source tools.

Don't Forget Mac OS Resource Forks

Mac OS files stored in MFS, HFS, or HFS Plus filesystems have two forks: the data fork and the resource fork. The *data fork* contains the actual data for the file, such as its text. The *resource fork* contains related structured data, such as offsets, menus,

* `dd` has this feature when you're using it to copy an individual file in a filesystem, of course. When using `dd` to copy a raw device, you will not change the access times of files in the filesystem.

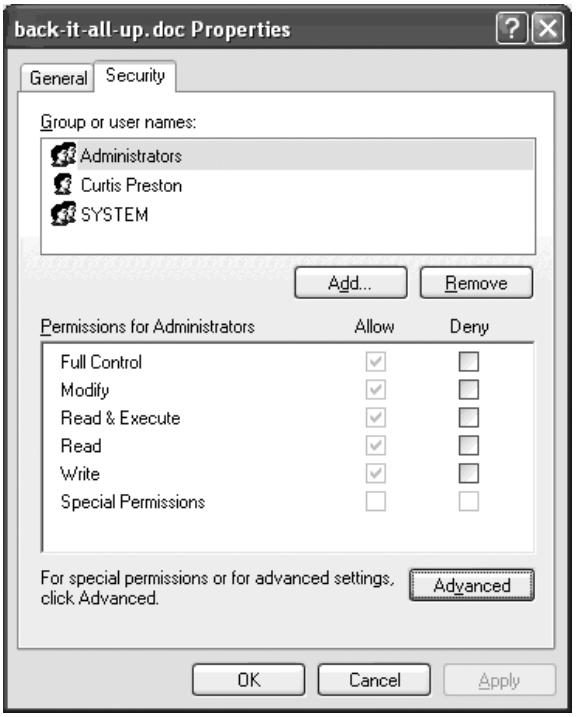


Figure 2-1. Access control list example

dialog boxes, and icons for the file. The two forks are tightly bound into a single file. While they are typically used by executables, every file can have a resource fork, and other applications can use it as well. For example, a word processing program may store a file's text in the data fork and the file's images in the file's resource fork.

These resource forks, like Windows ACLs, need to be backed up, and not all backup products back them up properly. Make sure you investigate what your backup system does with the data fork and resource fork.

Keep It Simple, SA

K.I.S.S. Have you seen this acronym before? It applies double or triple to backups. The more complicated your backup scheme is, the more likely it is to fail. If you do not understand it, you cannot implement it. Remember this every time you consider adding a new bell or whistle to your backup system. Every change puts your data at risk. Also, every change might make your backup system that much more complex—and more difficult to explain to the new backup person. One of the heads of support for a commercial backup product said that he sees the same thing over and over again. One person gets to know the software really well and writes various scripts to

automate this and that. Backups become a well-oiled machine—until they are turned over to the trainee. The trainee doesn't understand all the bells and whistles, and things start breaking. All of a sudden, your data is in danger. Keep that in mind the next time you think about adding some cool new feature to your backup script.

This next comment also relates to the previous section about "thinking big." One of the common judgment errors is to not automate in the beginning. It's so much easier to just put a hardcoded include list in a file somewhere or put it in the `cron` or scheduled task entry itself. However, that creates many different backup methods. If each box has its own special customized backup system, it is very hard to monitor your backups and explain them to the new person.

Remember, special is bad. Just keep saying it over and over again until you believe it.

It's not such a big deal when you have two or three systems, but it is when you grow to 200 systems. If you have to remember every system's idiosyncrasies every time you look at your logs, things inevitably get out of control. Exceptions for each system also can mean that things get overlooked. Do you remember that nine months ago you excluded */home** on *apollo*? I hope so, if *apollo* just became your primary NFS server, and it now has seven home directories.

If you cannot explain your backups to a stranger in less than a few hours, things are probably too complex. You should look at implementing things like centralized logging, standardized backup scripts, and some level of automation.

Read Those Manuals

The IP address of the backup server for a large software company was constantly changing to different, seemingly random IP addresses. The only identifiable pattern was that each new IP address the backup server would be assigned would be an IP address of one of the backup clients. Support cases were opened with vendors and all engineers were working 24/7 to resolve it, yet nobody could figure it out.

It turned out that a backup operator assigned to resolving backup issues was troubleshooting using the standard troubleshooting procedures for the group. But the new backup operator mixed up a few commands, so when trying to do basic name resolutions for the backup hosts (`nslookup` *`hostname`*), the command issued became `ifconfig -a` *`hostname`* instead. This changed the IP address of the backup server to whatever host was having backup issues, at random times of the day, and only on the days that operator was working.

—Jorgen Lie

Storing Your Backups

It doesn't do any good to make really good backups only to have your backup volumes destroyed, lost, or misplaced. You need to have a well-defined process for storing your media.

Storage in General

If you've read this far, you know that I consider your backups very important. If your backups are important, isn't the media on which they reside just as important? That goes without saying, right? Well, you'd never know it from most volume "libraries." Volume "piles" is probably a more accurate term. How many computer rooms have you seen that have volumes spread out all over the place? They get stacked, piled, fall behind the systems, and a tape cartridge works really well as a coaster for a coffee mug. (We wouldn't want to get any coffee rings on the new server, right?)

Have you ever really needed a volume and couldn't find it? I've been there. It's a horrible feeling to know that you've got the file on a volume, but can't find the darn volume! Why, then, do we treat our backup volumes like so much dirty laundry? Organize your backup volumes! Label them, catalog them, give them unique names or numbers, and put them in some sort of logical order in some kind of storage container. Do it, or the backup demon will come to haunt you!

Your ability to perform a large recovery quickly is directly related to how well you organize your media.

On-Site Storage

What about that media cabinet that you're using for your on-site volume storage? You don't have one, you say? You're using a file cabinet, you say? Well, use something, but if you can afford it, a number of companies make storage containers for media. They also make cabinets that can withstand fire. Spend the money; you'll be glad you did. Doing a restore is so much less stressful when you can find the volume with no problem. Remember, though, that fireproof does not mean heat-proof. These types of media safes are meant to withstand brief fires that are quickly extinguished by a sprinkler system. If a fire burns for a long time right next to the container or raises the temperature in the room significantly, the volumes may be no good anyway. (This is another good reason why you also must store volumes off-site.)

Have the most well-organized person in your office design your media storage system. Here's an idea. Ask your best administrative person to take a look at your storage system and compare it to his filing cabinet. Explain that you want an honest evaluation.

12,000 gold pieces

A financial institution where I once worked had an inventory of more than 12,000 pieces of media, and we never lost one. How did we do it, you ask? We treated every volume as if it were a piece of gold. Our inventory system was built on a number of things:

- Each volume had a unique numeric identifier.
- This number was in the form of a bar code placed on *every volume*. (Labeling more than 500 5 ¼-inch original installation floppies that came with our AT&T 3b2/1000s was no joy, I assure you, but we did it with the help of a team of temps!)
- Each volume's number, name, purpose, media type, date used, and location were stored in an Informix database.
- Every volume movement was tracked by that database. When a volume was taken to another building for a backup or restore, that movement was recorded in the database. If a volume was sent to our off-site storage vendor, that was stored in the database. If an administrator borrowed a backup volume or installation CD, it was recorded in a field called "Loaned to:".
- There was a manual log for when we moved media out of the media library momentarily for restores. For daily, high-volume moves, we used a bar code scanner with a shell script that automatically updated the database.
- We did a complete inventory every other quarter and a spot-check inventory once a month. If the spot-check inventory turned up too many errors, it was time for another full inventory.
- During the inventory we checked every volume against a printout of the database and every entry in the printout against an actual volume. (The latter half of the inventory consisted of hunting down errant administrators who had squirreled away backups or installation media in their drawers.)
- The volumes were stored in Wrightline media cabinets and were behind locked doors. Only the backup operators had access to the volumes. (These were the same operators who were held responsible if something came up missing.)
- The inventories were called "self-audits," and there was also an annual internal audit by the audit department, as well as the external audit by the Office of the Comptroller of Currency. They would comb through our logs, looking for inconsistencies. They had a knack for finding entries that looked a little weird and saying, "Let me see this one…."
- This entire process was thoroughly documented, and the institution is still following these procedures to this day, although it has probably improved them a bit.

The OCC takes this whole issue of data protection very seriously. (The OCC, by the way, is the group that has the power to say you can no longer be a bank. You want to make sure that they are happy with your procedures.)

Off-Site Storage

Once you have organized the media that you are storing on-site, it's time to consider off-site storage. There are two ways to store your data off-site:

- Media vaulting (they hold your tapes)
- Electronic vaulting (no tapes)

The latter can be expensive, but not as expensive as some people think. It is also much easier to use during a disaster, and you can't lose tapes if there aren't any tapes to lose. That is, of course, what off-site storage is meant to prepare you for—the destruction of your media and/or the building that holds it. If you have a complete set of backups in another location, you will be able to recover from even the worst local disaster.

Choosing a media vaulting vendor

Choosing a media vaulting vendor is as important a task as choosing your backup software. Choosing the wrong vendor can be disastrous. You depend on that vendor as your last line of defense, which is why you are paying them. Therefore, their storage and filing procedures need to be above reproach. They need to be better than the scenario I described in the "12,000 gold pieces" section earlier in this chapter. Their movement-tracking procedure must be free of holes. Here is a list of things to consider when choosing an off-site storage vendor:

Individual media accountability
: The first media vaulting vendor I ever used stored all of my volumes inside cases. They never inventoried the individual pieces of media. It was up to me to know which volume was in which case. When I needed a volume from one of the cases, they had to go in and get it. Once that was done, there was no log of where that volume actually existed. This is referred to as *container vaulting*. Most media vaulting companies also offer individual media vaulting. This method ensures that every volume is being tracked.

Bar-coded, location-based inventory
: Again, each volume should have a bar code that allows your storage vendor to scan every volume in and out. They should scan volumes into their vault when they arrive and scan them out when they give them back to you.

Electronic double check
: If you are keeping track of every volume's location, and your vendor is too, you should double-check each other. One or both of you can print out an export of your database that shows volume locations. You can write a program that cross-checks the location of every volume against the other inventory. I can't tell you how many times such a program has saved me. It's great to find an error when it happens, instead of weeks later when you need a volume that got misplaced.

Testing your chosen vendor

See if your vendor is on their toes. One tricky thing you can do is to see if they leave you alone in the vault. You are a customer of this company, so ask them if you can do an inventory of your media alone. See if they allow you unrestricted access to the inside of the vault. If they leave you alone inside the vault with no supervision, you have access to other companies' media. That means that at certain times, other companies may have access to your media. Run, don't walk, away from this company.

Make surprise inspections. Make spot checks. Ask for random volumes back, and see how quickly they can find them. Ask for volumes you just sent them. Volumes in the process of being inventoried are the hardest to find, but they should be able to do it. If you regularly send them five volumes a day with an inventory, put four volumes in one day, but list five on the inventory. See if they notice. If they don't, raise a ruckus! Their procedures should protect you from these types of human errors. If they don't, those procedures need to be improved. Be unpredictable. If you become predictable, you may be overlooked. Keeping them on their toes will make them remember you—and how important you think your volumes are. (By the way, your ability to make surprise inspections and spot checks should be spelled out in your contract. Make sure that it is OK for you to do this. If it is not…well, you know what to do.)

Vendors store two types of volumes: those that rotate in and out and those that stay there indefinitely. As you rotate the cyclical volumes in and out, they are inventoried. Your archive volumes are another story. If a volume has been there for two years and has never been touched, how do you know that it's OK? You should make a full inventory of those volumes at least once, preferably twice, every year.

Send the original, keep the copies. One of the things that you should regularly test is your copy procedure. If you are sending volumes off-site, some backup products give you the option of sending the originals or copies. If you can, send the originals. When it comes time for a restore, use your copy. If things go wrong, you can always go get the original. This process validates your copy procedure every time you do a restore. You can correct flaws in the process before disaster strikes. I remember several instances when a volume was eaten in a drive, or had soda spilled on it, and we needed that off-site copy really badly. That is the wrong time to find out your copy procedure is no good!

Electronic vaulting

Electronic vaulting is becoming quite popular. It can be expensive, but it's a beautiful thing. If you can afford it, I highly recommend it. The premise is that your backups are sent directly to a storage system at the electronic vaulting vendor. One question you need to ask yourself is, "What happens if *they* burn to the ground?" All your data could be lost. Don't let this happen. Make sure that this storage company

is not the only location for your backed-up data. In addition, make sure that you know how you're going to do a large restore. While a small network link may be large enough to do a continuous incremental backup, it's probably not large enough to do a 100 GB restore. If this is a concern, ask your electronic vaulting vendor about a local recovery appliance.

Testing Your Backups

I wish there were enough to say about this to make it a separate chapter, because it's that important. I can't tell you how many stories I have heard about people who waited until they needed a major restore before they tested their backups. That's when they found out that they'd been using the wrong device or the wrong blocking factor, or that the device had I/O errors. This point cannot be stated strongly enough. If you don't test your backups, you are guaranteed to get a surprise sooner or later.

Test Everything!

It is important to test every type of restore. If you are testing filesystem backups, make sure you:

- Restore many single files. Can you find the needle in the haystack?
- Restore an older version of a file.
- Restore an entire drive or filesystem, and compare your results with the original. Are they the same size, and so on?
- Pretend that an entire system is down, and try to recreate it.
- Pretend that a particular volume is bad, and force yourself to use an alternate backup.
- Retrieve a few volumes from your off-site storage vendor.
- Pretend that your backup server is destroyed, and try to recover from that. (This one's tough!) This test is extremely important if you are using an open-source or commercial backup utility. Some products do not plan for this well, and you can find yourself in a real Catch-22 situation.

If you are testing database restores, make sure you:

- Restore part of your database, pretending that you lost only one data file or disk drive, if this option is available.
- Restore the entire database onto another server; this is where you learn about files that you are not including.

- Restore the database up to a point in time, earlier than the present time (this is helpful practice for recovering from a DBA or user error).
- Pretend that last night's backup failed, and force yourself to use an older backup. Theoretically, if you have saved all your transaction logs to a backup volume, you should be able to use a backup that is weeks old and roll it forward to the present time using those logs. This is another strong argument for using transaction logs.

Test Often

As I said earlier, sit around one day with some really pessimistic people, and ask them to dream up scenarios for you to test. Test your ability to recover from each of these scenarios *on a regular basis*. What works this month might not work next month. The only thing that is guaranteed to remain constant is change. One suggestion is to create a list of recovery procedures and randomly test a subset of them every month. Management changes, hardware changes, networks change, and OS and database versions change. Every change you know about should make you want to perform a test of the affected area.

Monitoring Your Backups

If you are not monitoring your backups, they are not doing what you think they are doing—guaranteed. This is one pot that will not boil if you don't watch it. Every backup should have a log that is examined daily. This can be automated as well. Here are some examples:

Give me a summary. `dump` gives a whole bunch of messages that I couldn't care less about, `Pass I`, `Pass II`, `% done`, and so on. When I'm monitoring the `dump` backups of hundreds of drives or filesystems, most of that is so much noise. What I really want to see is what got dumped, where it went, when it went, what level it was, and the ever-popular `DUMP IS DONE` message. To get a summary of just these lines, the first thing I do is use `grep -v` to exclude the phrases I don't want, leaving only a few lines. This is much easier to review. This technique can also be applied to other Unix, Linux, and Mac OS backup commands.

You can do something similar with a Windows scheduled task. You can cause the task to create a log somewhere on the *C:* drive, and then have that log emailed to you via a command line SMTP mailer such as `blat`. (`blat` is a free command line utility that sends email using SMTP or a post to Usenet using NNTP. You can download it from *http://www.blat.net*.)