Web Mapping Illustrated
By Tyler Mitchell
.....
Publisher: **O'Reilly**
Pub Date: **June 2005**
ISBN: **0-596-00865-1**
Pages: **368**

Table of Contents | Index | Errata

Overview

With the help of the Internet and accompanying tools, creating and publishing online maps has become easier and rich with options. A city guide web site can use maps to show the location of restaurants, museums, and art venues. A business can post a map for reaching its offices. The state government can present a map showing average income by area.

Developers who want to publish maps on the web often discover that commercial tools cost too much and hunting down the free tools scattered across Internet can use up too much of your time and resources. Web Mapping Illustrated shows you how to create maps, even interactive maps, with free tools, including MapServer, OpenEV, GDAL/OGR, and PostGIS. It also explains how to find, collect, understand, use, and share mapping data, both over the traditional Web and using OGC-standard services like WFS and WMS.

Mapping is a growing field that goes beyond collecting and analyzing GIS data. Web Mapping Illustrated shows how to combine free geographic data, GPS, and data management tools into one resource for your mapping information needs so you don't have to lose your way while searching for it.

Remember the fun you had exploring the world with maps? Experience the fun again with Web Mapping Illustrated. This book will take you on a direct route to creating valuable maps.

Web Mapping Illustrated
By Tyler Mitchell
.....
Publisher: **O'Reilly**
Pub Date: **June 2005**
ISBN: **0-596-00865-1**
Pages: **368**

Table of Contents | Index | Errata

Copyright
Foreword
Preface
- Youthful Exploration
- The Tools in This Book
- What This Book Covers
- Organization of This Book
- Conventions Used in This Book
- Safari Enabled
- Comments and Questions
- Acknowledgments

Chapter 1. Introduction to Digital Mapping
- Section 1.1. The Power of Digital Maps
- Section 1.2. The Difficulties of Making Maps
- Section 1.3. Different Kinds of Web Mapping

Chapter 2. Digital Mapping Tasks and Tools
- Section 2.1. Common Mapping Tasks
- Section 2.2. Common Pitfalls, Deadends, and Irritations
- Section 2.3. Identifying the Types of Tasks for a Project

Chapter 3. Converting and Viewing Maps
- Section 3.1. Raster and Vector
- Section 3.2. OpenEV
- Section 3.3. MapServer
- Section 3.4. Geospatial Data Abstraction Library (GDAL)
- Section 3.5. OGR Simple Features Library
- Section 3.6. PostGIS
- Section 3.7. Summary of Applications

Chapter 4. Installing MapServer
- Section 4.1. How MapServer Applications Operate
- Section 4.2. Walkthrough of the Main Components
- Section 4.3. Installing MapServer
- Section 4.4. Getting Help

Chapter 5. Acquiring Map Data
- Section 5.1. Appraising Your Data Needs
- Section 5.2. Acquiring the Data You Need

Chapter 6. Analyzing Map Data
- Section 6.1. Downloading the Demonstration Data
- Section 6.2. Installing Data Management Tools: GDAL and FWTools
- Section 6.3. Examining Data Content
- Section 6.4. Summarizing Information Using Other Tools

Chapter 7. Converting Map Data

Section 7.1. Converting Map Data
Section 7.2. Converting Vector Data
Section 7.3. Converting Raster Data to Other Formats
Chapter 8. Visualizing Mapping Data in a Desktop Program
Section 8.1. Visualization and Mapping Programs
Section 8.2. Using OpenEV
Section 8.3. OpenEV Basics
Chapter 9. Create and Edit Personal Map Data
Section 9.1. Planning Your Map
Section 9.2. Preprocessing Data Examples
Chapter 10. Creating Static Maps
Section 10.1. MapServer Utilities
Section 10.2. Sample Uses of the Command-Line Utilities
Section 10.3. Setting Output Image Formats
Chapter 11. Publishing Interactive Maps on the Web
Section 11.1. Preparing and Testing MapServer
Section 11.2. Create a Custom Application for a Particular Area
Section 11.3. Continuing Education
Chapter 12. Accessing Maps Through Web Services
Section 12.1. Web Services for Mapping
Section 12.2. What Do Web Services for Mapping Do?
Section 12.3. Using MapServer with Web Services
Section 12.4. Reference Map Files
Chapter 13. Managing a Spatial Database
Section 13.1. Introducing PostGIS
Section 13.2. What Is a Spatial Database?
Section 13.3. Downloading PostGIS Install Packages and Binaries
Section 13.4. Compiling from Source Code
Section 13.5. Steps for Setting Up PostGIS
Section 13.6. Creating a Spatial Database
Section 13.7. Load Data into the Database
Section 13.8. Spatial Data Queries
Section 13.9. Accessing Spatial Data from PostGIS in Other Applications
Chapter 14. Custom Programming with MapServer's MapScript
Section 14.1. Introducing MapScript
Section 14.2. Getting MapScript
Section 14.3. MapScript Objects
Section 14.4. MapScript Examples
Section 14.5. Other Resources
Section 14.6. Parallel MapScript Translations
Appendix A. A Brief Introduction to Map Projections
Section A.1. The Third Spheroid from the Sun
Section A.2. Using Map Projections with MapServer
Section A.3. Map Projection Examples
Section A.4. Using Projections with Other Applications
Section A.5. References
Appendix B. MapServer Reference Guide for Vector Data Access
Section B.1. Vector Data
Section B.2. Data Format Guide
ESRI Shapefiles (SHP)
PostGIS/PostgreSQL Database
MapInfo Files (TAB/MID/MIF)
Oracle Spatial Database
Web Feature Service (WFS)
Geography Markup Language Files (GML)

VirtualSpatialData (ODBC/OVF)
TIGER/Line Files
ESRI ArcInfo Coverage Files
ESRI ArcSDE Database (SDE)
Microstation Design Files (DGN)
IHO S-57 Files
Spatial Data Transfer Standard Files (SDTS)
Inline MapServer Features
National Transfer Format Files (NTF)
Colophon
About the Author
Colophon
Index

Web Mapping Illustrated

by Tyler Mitchell

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

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (safari.oreilly.com). For more information, contact our corporate/institutional sales department: (800) 998-9938 or corporate@oreilly.com.

Editor:	Simon St.Laurent
Production Editor:	Mary Anne Weeks Mayo
Cover Designer:	Ellie Volckhausen
Interior Designer:	David Futato
Printing History:	
June 2005:	First Edition.

Nutshell Handbook, the Nutshell Handbook logo, and the O'Reilly logo are registered trademarks of O'Reilly Media, Inc. Web Mapping Illustrated, the image of a snipe, and related trade dress are trademarks of O'Reilly Media, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

ISBN: 0-596-00865-1

[C]

Foreword

For novices and geospatial experts alike, mapping technologies are undergoing as significant a change as has been seen since mapping first went digital. The prior introduction of Geographic Information Systems (GIS) and other digital mapping technologies transformed traditional map making and introduced an era of specialists in these new geographic technologies. Today, an even newer set of technological advancements are bringing an equally massive change as digital mapping goes mainstream. The availability of Global Positioning Systems (GPS), broadband Internet access, mass storage hard drives, portable devices, and—most importantly—web technologies are accelerating the ability to incorporate geographic information into our daily lives. All these changes have occurred simultaneously and so quickly that the impact of only a fraction of the full potential of spatial technologies has yet been felt.

In parallel with the exciting opportunities that modern technologies are providing the digital geospatial universe, a less broadly known but perhaps far more important phenomenon has emerged: a new world of open source collaboration. Open source development and user communities, along with a healthy commitment from industry, are filling the growing need and demand for spatial technologies for making better decisions and providing more information to the growing mapping needs of technology users. In a multidimensional world, geography forms a common framework for disseminating information. The open source community and industry is filling that need at a growth rate unmatched in the industry.

In an age when web technologies have erased the distances between peoples of different continents and nationalities, this book and the technologies behind it remind us of the continued importance of place in the world in which we live. Mapping has always highlighted the differences and variations that occur over space; but at the same time it has reminded us that we share this world with our neighbors, and our actions have impact beyond ourselves. Hopefully, web mapping technologies will help to bring this powerful information to all of us for our common future good.

If you are reading this book without ever having heard of Geographic Information Systems or Remote Sensing, you are not alone. It is for you that the publishing of this book is so timely; it is now that mapping technologies are for the first time becoming readily accessible to the broader IT world. The incredible wealth of information provided in this book will allow you to interact with the open source mapping community as so many have already done, and will one day allow you to help the many others that will follow.

I hope that this book will, if nothing else, engage you in understanding the power that mapping information can bring to your web presence and other IT needs—regardless of whether you are with an NGO, a small or large corporation, or a government organization. The importance of this book cannot be overstated. It comes at a critical stage, when two phenomena with tremendous momentum are coming together: the emergence of Open Source mapping technology, and the availability of technologies enabling digital mapping to become accessible by the masses.

Dave McIlhagga

President, DM Solutions Group

Preface

What is it about maps? For some of us, maps are intriguing no matter where we are. I've spent hours poring over them learning about foreign places. There is a sense of mystery surrounding maps. They contain information that can only be revealed through exploration.

Digital maps allow a user to explore even further by providing an interactive experience. Most maps have traditionally been static. Now digital maps allow users to update information and customize it for their particular needs.

Youthful Exploration

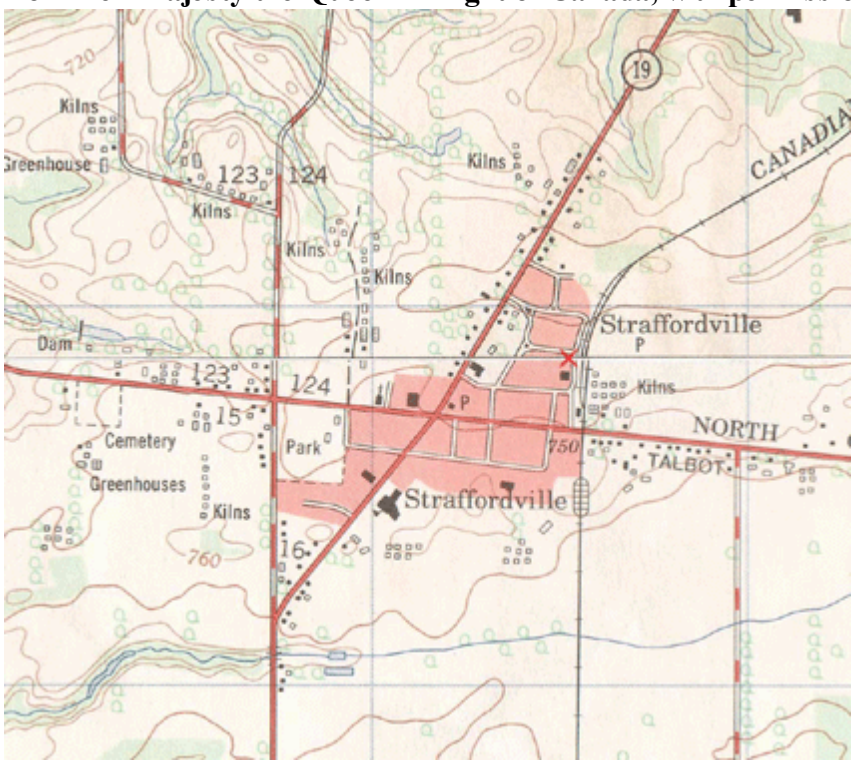
For me, map-based exploration started at a young age. I remember the thrill of finding our Scout camp on a topographic map. Part of the map is shown in Figure P-1. I found my home town, local roads, and even greenhouses. It was hard to believe that someone bothered to map the streets I used to play on or the tobacco fields I worked in during summer vacation. Yet there they were, drawn on this fascinating map that hangs on my office wall 20 years later.

These maps opened the door to planning hiking adventures and bicycle trips. I blame maps for luring me further and further away from home—to see what a map symbol or town looked like on the ground.

When I wasn't exploring, I was often on the computer learning more about the digital world. My combined interest in computers and exploration naturally led me to the field of computerized mapping and geographic information systems (GIS). It never occurred to me that my enjoyment of maps and computers would become a career.

Whether showing a friend where you live or displaying the path of a pending hurricane, maps play an important role in lives of people everywhere. Having the tools

Figure P-1. Part of the topographic map of my home town in southern Ontario, Canada; my home was located at the X symbol. Portions of NTS map sheets 40l/15c and 10c/15c ©2005. Produced under licence from Her Majesty the Queen in Right of Canada, with permission of Natural Resources Canada.



and ability to map the world you live in is incredibly powerful. The following quote is from the Mayan Atlas:

Maps are power. Either you will map or you will be mapped. If you are mapped by those who desire to own or control your land and resources, their map will display their justifications for their claims, not yours.

Having open access to mapping tools further enables mapping efforts. Being able to share those maps with the world through web mapping makes the effort all the more worthwhile.

The Tools in This Book

The tools in this book are the results of a handful of open source projects. They are a critical set of tools in my professional data management toolbox. From visualizing map data to converting between formats, I have come to depend on many of them daily. They are also an important part of my future goals for mapping and data management.

These tools are a subset of what is available today, both in the mainstream commercial market and in the open source realm. Because these are open source, they are free for you to use and adopt as you see fit. Many of them are pushing the envelope of what even the commercial products can do.

I began using many of these tools shortly after finishing university. My day job was in mapping and geospatial data analysis, and I had access to some of the latest commercial tools. However, when I wanted to pursue projects at home on my own time, the traditional tools were simply not available. The licensing restrictions and costs forced me to find alternatives; eventually, the open source tools took over. Any gaps in my array of tools will likely be filled within a year of this book being published.

There is a lot of active development going on across the spectrum of open source mapping and GIS projects. Many projects use the latest open standards for interoperability and tend to implement them much faster than the commercial products.

My initial motivation for writing was to fill in the gaps of existing documentation and answer the new user's common questions. I hope it does this and more. I hope you become as excited about these tools as I am. Years of programming have given us a powerful toolkit for mapping, data management, and even youthful exploration.

What This Book Covers

This book introduces several concepts and tools. They can be grouped into the following four categories:

- Mapping and data-management concepts
- Command-line data-management tools
- Command-line and web-based mapping tools
- Spatial database management

You will study the following tools:

Geospatial Data Abstraction Library (GDAL) with OGR

This tool includes application programming interfaces (APIs) and command-line utilities for raster and vector data. GDAL's web site is http://www.gdal.org.

OpenEV

For basic desktop GIS and imagery analysis; includes tools to draw new map features for use in other programs.

UMN MapServer

This tool includes command-line tools to build CGI web applications and uses the MapServer API, called MapScript, to custom-script mapping applications.

PostGIS

This tool is an extension to the PostgreSQL database management system that allows you to store and manipulate spatial data alongside tabular data.

Organization of This Book

This book is organized into 14 chapters and 2 appendixes:

Chapter 1, Introduction to Digital Mapping

This chapter introduces digital mapping, including web mapping, and presents some of the barriers to using the technology. It also includes a list of web sites providing web mapping services and outlines the technology required to do web mapping.

Chapter 2, Digital Mapping Tasks and Tools

This chapter outlines the goals of digital mapping and the common types of tasks involved including viewing, analysis, creating/manipulating, conversion, and sharing.

Chapter 3, Converting and Viewing Maps

This chapter introduces the concepts of raster and vector data types, and the main tools used in this book: OpenEV, MapServer, GDAL, OGR, and PostGIS.

Chapter 4, Installing MapServer

In this chapter, we walk through the main components of MapServer applications. You'll find detailed instructions for installing binaries or compiling MapServer from source. The chapter also provides a list of MapServer support contacts.

Chapter 5, Acquiring Map Data

This chapter discusses how to assess your data needs and acquire data to meet those needs. It provides a list of resources for finding free mapping data.

Chapter 6, Analyzing Map Data

This chapter covers setting up the FWTools package and using GDAL/OGR utilities for examining raster and vector datasets. Here you'll find examples that combine these utilities with command-line text processing tools to produce customized reports and summaries.

Chapter 7, Converting Map Data

This chapter shows how to convert raster and vector data between formats using GDAL/OGR utilities. You'll learn how to convert between formats such as ESRI shapefiles, GML, DGN, and PostGIS formats.

Chapter 8, Visualizing Mapping Data in a Desktop Program

This chapter provides a list of desktop mapping programs. It also introduces OpenEV as a desktop mapping program and walks through common tools in OpenEV. Here, you'll find examples of color-theming and preparing 3D views.

Conventions Used in This Book

Italic is used for:

- New terms where they are defined
- Emphasis in body text
- Pathnames, filenames, and program names; however, if the program name is also the name of a Java class, it is written in constant width font, like other class names
- Host and domain names (e.g., http://www.maptools.org)

Constant width is used for:

- Code examples and fragments
- Anything that might appear in an XML document, including element names, tags, attribute values, entity references, and processing instructions
- Anything that might appear in a program, including keywords, operators, method names, class names, utilities, and literals

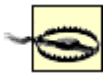
Constant width bold is used for:

- User input
- Emphasis in code examples and fragments

Constant width italic is used for:

- Replaceable elements in code statements

This icon indicates a tip, suggestion, or general note.



This trap icon indicates a warning or caution.

Case-sensitive filenames and commands don't always allow authors to adhere to standard English grammar. It is usually possible to rewrite the sentence so the two don't conflict, and when possible I have endeavored to do so. However, on rare occasions when there is simply no way around the problem, I let standard English come up the

Safari Enabled

When you see a Safari® Enabled icon on the cover of your favorite technology book, it 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 technology 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://safari.oreilly.com.

Comments and Questions

Please address comments and questions concerning this book to the publisher:
O'Reilly Media, Inc.1005 Gravenstein Highway NorthSebastopol, CA 95472(800) 998-9938 (in the United States or Canada)(707) 829-0515 (international or local)(707) 829-0104 (fax)

There's a web page for this book that lists errata, examples, and any additional information. You can access this page at:
http://www.oreilly.com/catalog/webmapping

To comment or ask technical questions about this book, send email to:
bookquestions@oreilly.com

For more information about our books, conferences, Resource Centers, and the O'Reilly Network, see our web site at:
http://www.oreilly.com

Acknowledgments

Several fellow passengers on this writing roller coaster deserve special mention. This project would never have happened without the support and patience of my editor, Simon St.Laurent. He helped me through several proposals, refining and focusing the content of this book.

Regardless of a successful book proposal, I would never have pursued this project without the support of my loving wife. Her encouragement, patience, and enthusiasm kept me going to the end.

Technical reviewers for this book helped catch my mistakes and improve the content substantially. Thank you very much to all of them: Bart van den Eijnden, Darren Redfern, Jeff McKenna, Paul Ramsey, and Tom Kralidis. A handful of others helped answer my endless stream of questions. If you helped with even a yes/no question, it was appreciated.

I'm thankful for the support and encouragement I received from Dave McIlhagga and DM Solutions Group in general. It was a continual reminder that this book was necessary.

The developers of these tools deserve special recognition for their contributions to open source GIS and mapping. Without them, there wouldn't be much to write to about! I would also like to acknowledge the long-term support of the University of Minnesota and their willingness to let the MapServer community grow beyond their borders.

A significant amount of documentation already exists for MapServer. Without the efforts of many companies and volunteers, I would never have learned as much as I have about these great tools. Thank you, fellow authors. In no way does this book mean to downplay your efforts.

Several friends and colleagues have helped encourage me over the years. Without their encouragement to think outside the box and strive for something better, I doubt I'd be writing this today.

I spent way too many hours on IRC channels picking the brains of other chatlings. When email was just too slow, this help was much appreciated and allowed real-time assistance from the broader community.

Chapter 1. Introduction to Digital Mapping

Not long ago, people drew and colored their maps by hand. Analyzing data and creating the resulting maps was slow and labor intensive. Digital maps, thanks to the ever-falling cost of processing power and storage, have opened up a whole new range of possibilities. With the click of a mouse or a few lines of code, your computer analyzes, draws, and color-themes your map data. From the global positioning system (GPS) in your car to the web site displaying local bus routes, digital mapping has gone mainstream.

Of course, learning to produce digital maps requires some effort. Map data can be used incorrectly, resulting in maps with errors or misleading content. Digital mapping doesn't guarantee quality or ethics, just like conventional mapping.

1.1. The Power of Digital Maps

When you contrast the methods of conventional and digital mapping, the power of digital mapping becomes evident. The process of conventional mapping includes hand-drawn observations of the real world, transposed onto paper. If a feature changes, moves, or is drawn incorrectly, a new map needs to be created to reflect that change. Likewise if a map shows the extent of a city and that city grows, the extent of the map will need to be changed and the map will need to be completely recreated.

These problems are reduced with digital mapping. Because features are stored as distinct layers in a computer file, you can modify a map without starting from scratch. Once a feature is modified, the computer-based map instantly reflects the change the next time the feature is viewed. Interactive maps allow the user to view the precise area they are interested in, rather than be confined by the dimensions of a printed page. The user can also choose to view only certain pieces of content. The mapmaker doesn't have to guess which information the viewer wants to see but can make it possible for the reader to choose.

Instead of focusing on the details of a particular area of the world to map, the digital mapmaker can focus on how to best present information. This is much like the difference between an author and a web page designer. When you move into the digital realm, the focus is more on helping others find information rather than presenting static representations of information, as on a printed page. Today's mapmaker is often a web site developer, programmer, or some sort of geographic information analyst. Her focus is on managing and presenting information to a specific audience, be it in finance, forestry, or national defense, for instance.

1.2. The Difficulties of Making Maps

If you've worked with maps, digital or conventional, you'll know that despite my enthusiasm, mapping isn't always easy. Why do we often find it so difficult to make maps of the world around us? How well could you map out the way you normally drive to the supermarket? Usually, it's easier to describe your trip than it is to draw a map. Perhaps we have a perception of what a map must look like and therefore are afraid to draw our own, thinking it might look silly in comparison. Yet some maps drawn by a friend on a napkin might be of more use than any professional city map could ever be.

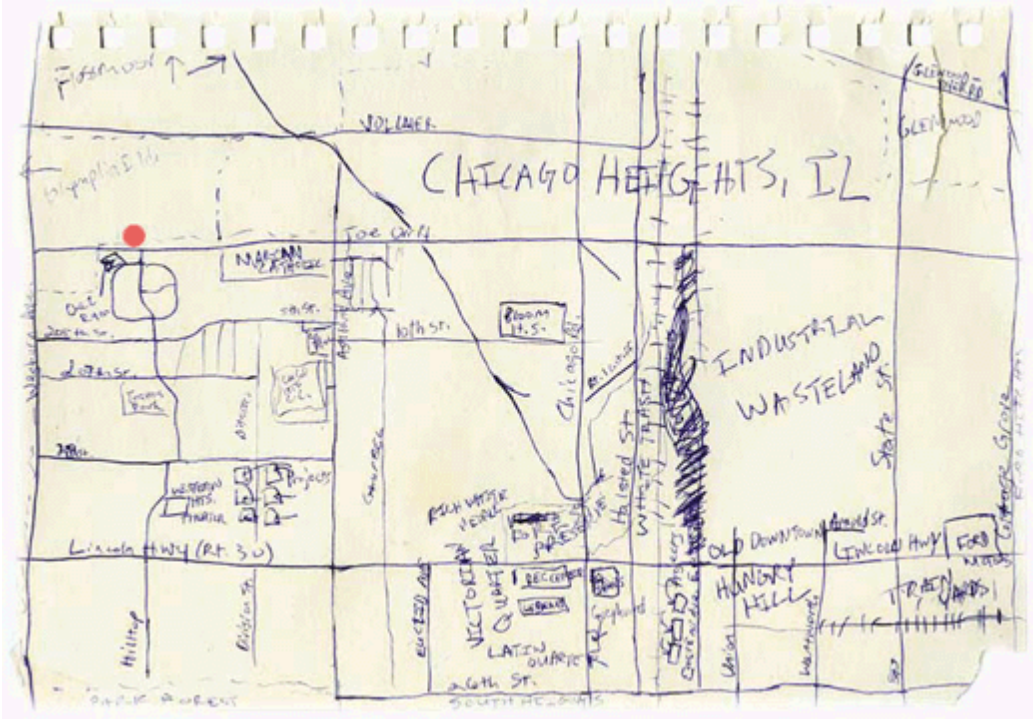
1.2.1. Personal Maps

The element of personal knowledge, rather than general knowledge, is what can make a somewhat useful map into one that is very powerful. When words fail to describe the location of something that isn't general knowledge, a map can round out the picture for you. Maps can be used to supplement a verbal description, but because creating a map involves drawing a perspective from your head, it can be very intimidating. That intimidation and lack of ownership over maps has created an interesting dilemma. In our minds, maps are something that professionals create, not the average person. Yet a map like the one shown in Figure 1-1 can have much more meaning to someone than a professional map of the same area. So what are the professional maps lacking? They show mostly common information and often lack personal information that would make the map more useful or interesting to you.

1.2.2. Technology Barriers

Digital mapping isn't a new topic. Ever since computers could create graphic representations of the earth, people have been creating maps with them. In early computing, people used to draw with ASCII text-based maps. (I remember creating ASCII maps for role-playing games on a Tandy color computer.) However, designing graphics with ASCII symbols wasn't pretty. Thankfully, more sophisticated graphic techniques on personal computers allow you to create your own high-quality maps.

Figure 1-1. A personal map drawn by Ryan Mendenhall showing Chicago Heights, Illinois, U.S.A.; this map is courtesy of Lori Napoleon's maps project web site: http://www.subk.net/maps.html



You might already be creating your own maps but aren't satisfied with the tools. For some, the cost of commercial tools can be prohibitive, especially if you just want to play around for a while to get a feel for the craft. Open source software alleviates the need for immediate, monetary payback on investment.

1.3. Different Kinds of Web Mapping

One very effective way to make map information available to a group of nontechnical end users is to make it available through a web page. Web mapping sites are becoming increasingly popular. There are two broad kinds of web mapping applications: static and interactive.

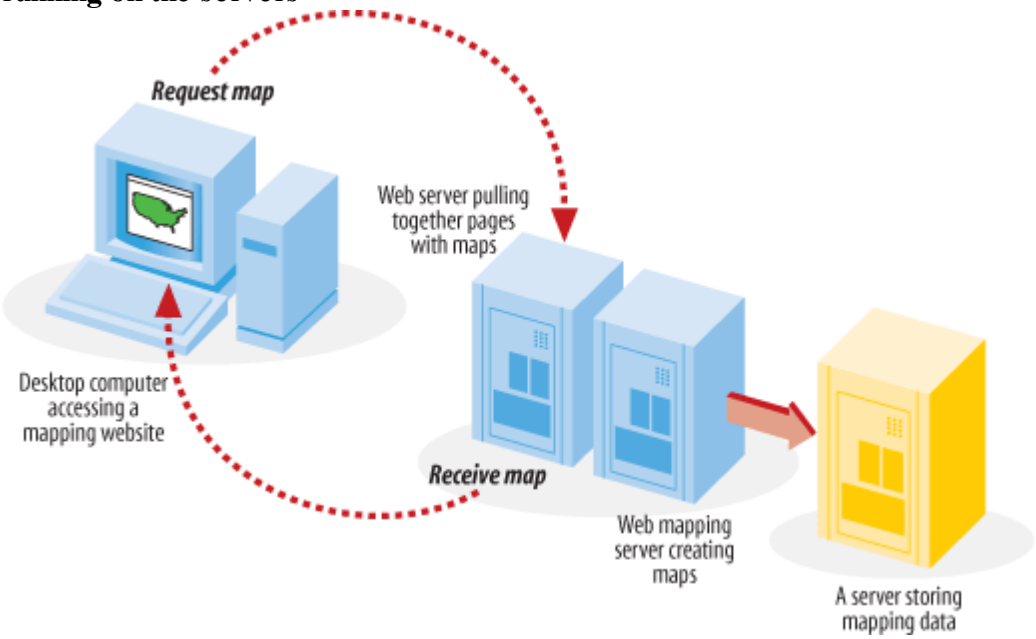
Static maps displayed as an image on a web page are quite common. If you already have a digital map (e.g., from scanning a document), you can be up and running very quickly with a static map on your web page. Basic web design skills are all you need for this because it is only a single image on a page.

This book doesn't teach web design skills. O'Reilly has other books that cover the topic of web design, from basic to advanced, including: Learning Web Design, Web Design in a Nutshell, HTML and XHTML: The Definitive Guide, and many more.

Interactive maps aren't as commonly seen because they require specialized skills to keep such sites up and running (not to mention the potential costs of buying off-the-shelf software). The term interactive implies that the viewer can somehow interact with the map. This can mean selecting different map data layers to view or zooming into a particular part of the map that you are interested in. All this is done while interacting with the web page and a map image that is repeatedly updated. For example, MapQuest is an interactive web mapping program for finding street addresses and driving directions. You can see it in action at http://www.mapquest.com.

Interactive maps that are accessed through web pages are referred to as web-based maps or simply web maps. These maps can be very powerful, but as mentioned, they can also be difficult to set up due to the technical skills required for maintaining a web server, a mapping server/program and management of the underlying map data. As you can see, these types of maps are fundamentally different from static maps because they are really a type of web-based program or application. Figure 1-2 shows a basic diagram of how an end user requests a map through a web mapping site and what happens behind the scenes. A user requests a map from the web server, and the server passes the request to the web mapping server, who then pulls together all the data. The map is passed all the way back to the end user's web browser.

Figure 1-2. A diagram of how a mapping web site interacts with the end user and the back-end programs running on the servers



1.3.1. Web Map Users

Chapter 2. Digital Mapping Tasks and Tools

Maps can be beautiful. Some antique maps, found today in prints, writing paper, and even greeting cards, are appreciated more for their aesthetic value than their original cartographic use. The aspiring map maker can be intimidated by these masterpieces of science and art. Fortunately, the mapping process doesn't need to be intimidating or mystical.

Before you begin, you should know that all maps serve a specific purpose. If you understand that purpose, you've decoded the most important piece of a mapping project. This is true regardless of how the map is made. Traditional tools were pen and ink, not magic. Digital maps are just a drawing made up of points strung together into lines and shapes, or a mosaic of colored squares.

The purpose and fundamentals of digital mapping are no different and no more complex than traditional mapping. In the past, a cartographer would sit down, pull out some paper, and sketch a map. Of course, this took skill, knowledge, and a great deal of patience. Using digital tools, the computer is the canvas, and software tools do the drawing using geographic data as the knowledge base. Not only do digital tools make more mapping possible, in most cases digital solutions make the work ridiculously easy.

This chapter explores the common tasks, pitfalls, and issues involved in creating maps using computerized methods. This includes an overview of the types of tasks involved with digital mapping—the communication of information using a variety of powerful media including maps, images, and other sophisticated graphics. The goals of *digital mapping* are no different than that of traditional mapping: they present geographic or location-based information to a particular audience for a particular purpose. Perhaps your job requires you to map out a proposed subdivision. Maybe you want to show where the good fishing spots are in the lake by your summer cabin. Different reasons yield the same desired goal: a map.

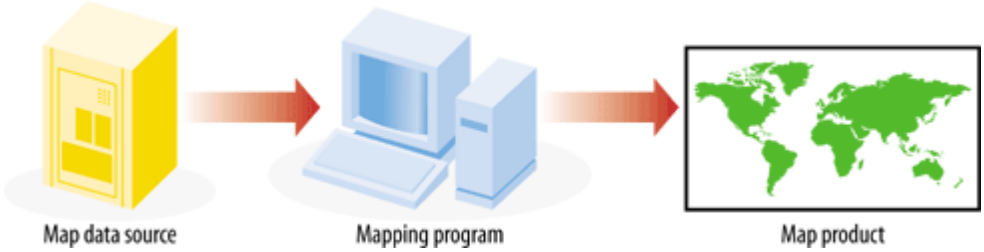
For the most part, the terms geographic information and maps can be used interchangeably, but maps usually refer to the output (printed or digital) of the mapping process. Geographic information refers to digital data stored in files on a computer that's used for a variety of purposes.

When the end product of a field survey is a hardcopy map, the whole process results in a paper map and nothing more. The map might be altered and appended as more information becomes available, but the hardcopy map is the final product with a single purpose.

Digital mapping can do this and more. Computerized tools help collect and interact with the map data. This data is used to make maps, but it can also be analyzed to create new data or produce statistical summaries. The same geographic data can be applied to several different mapping projects. The ability to render the same information without compiling new field notes or tracing a paper copy makes digital mapping more efficient and more fun.

Digital mapping applies computer-assisted techniques to a wide range of tasks that traditionally required large amounts of manual labor. The tasks that were performed are no different than those of the modern map maker, though the approach and tools vary greatly. Figure 2-1 shows a conceptual diagram of the digital mapping process.

Figure 2-1. Digital maps are made using a mapping program that accesses mapping data and gives the resulting map back to the user



2.1. Common Mapping Tasks

The process that produces a map requires three basic tasks: quantifying observations, locating the position of your observations, and visualizing the locations on a map. Digital tools have made these tasks more efficient and more accurate.

Quantifying observations

Measuring equipment such as laser range finders or imaging satellites provide discrete measurements that are less affected by personal interpretations. Traditional observations, such as manual photo interpretation or drawing features by hand, tend to introduce a biased view of the subject.

Locating positions of observations

Geographic referencing tools such as GPS receivers link on-the-earth locations to common mapping coordinate systems such as latitude and longitude. They calculate the receiver's location using satellite-based signals that help the GPS receiver calculate its location relative to satellites whose positions are well known. They act as a type of digital benchmark rather than using traditional survey or map referencing (best guess) methods. Traditional astronomical measurements or ground-based surveying techniques were useful but we now have common, consistent, and unbiased methods for calculating location.

Visualizing these locations on a map

Desktop mapping programs allow the user to compare location information with digital base map data. Traditional hand-drawn paper maps can't compete with the speed and flexibility of digital desktop mapping programs. Of course, digital mapping data is needed to do the job, but once data is available, infinite renditions of maps using the same base data is possible.

If a tool described here isn't effective, other tasks are affected. For example, poor recording of observations can still produce a map, but its accuracy is questionable.

The end goal of mapping is to present information about our observations of the world. The better that can be done, the better the goal has been met. When these tools are working together, the cartographic process can be very efficient.

2.2. Common Pitfalls, Deadends, and Irritations

Many maps can now be created in the safety of a home or office without the need to sail the seas to chart new territories. The traditional, nondigital methods of surveying to produce maps held certain dangers that are quite different from those of today. However, digital mapping has its own problems and irritations. The pitfalls today may not involve physical danger but instead involve poor quality, inappropriate data, or restricted access to the right kind of information.

2.2.1. Finding Good Source Data

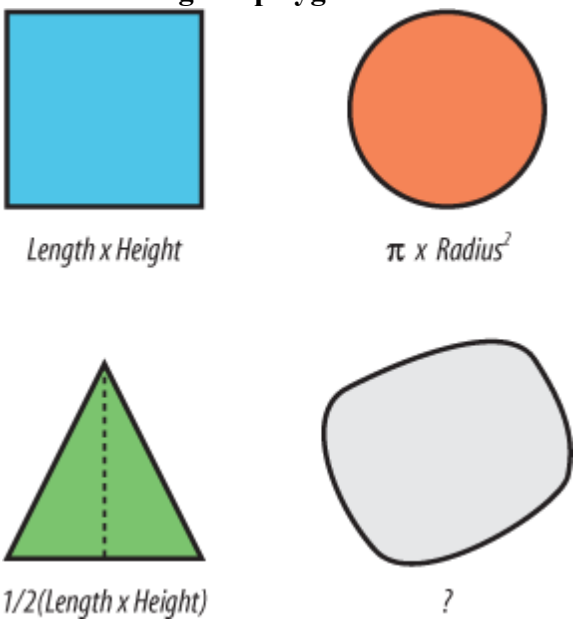
A bad map isn't much better than a blank map. This book doesn't discuss good map design (the art of cartography), but an equally important factor plays into digital mapping: the quality of source data. Because maps are based on some sort of source data, it is imperative to have good quality information at the beginning. The maxim garbage in, garbage out definitely applies; bad data makes for bad maps and analysis. This is discussed in more detail in Chapter 5.

2.2.2. Dependency on Digital Tools

With the advent of digital mapping has come the loss of many traditional mapping skills. While digital tools can make maps, there are some traditional skills that are helpful. You might think that training in digital mapping would include the theory and techniques of traditional mapping processes, but it often doesn't. Today, many who do digital mapping are trained to use only a specific piece of software. Take away that software or introduce a large theoretical problem, and they may be lost. Map projection problems are a good example. If you don't understand how projections work, you can severely degrade the quality of your data when reprojecting and merging datasets as described in Chapter 8 and discussed in Appendix A.

Another example would be the ignorance of geometric computations. It was quite humiliating to realize that I didn't know how to calculate the area of an irregular polygon; nor was it even common knowledge to my most esteemed GIS colleagues. Figure 2-2 shows a range of common shapes and the formulae used to calculate their area.

Figure 2-2. How to calculate the area of common shapes, including a square, circle, triangle, and the infamous irregular polygon



Many of the calculations were taught in elementary school but not for the irregular polygon. It doesn't use a simple formula, and there are multiple ways to calculate the area. One method is to triangulate all the vertices that make up the polygon boundary and then calculate and sum the area of all those triangles. You can see why we let the computer do this kind of task!

2.3. Identifying the Types of Tasks for a Project

Just like carpenters, map makers know the value of using the right tool for the job. The digital map maker has a variety of tools to choose from, and each tool is designed for a certain task. Many tools can do one or two tasks well, and other tasks moderately well or not at all. There are five different types of tools used in digital mapping and its related disciplines. These are general categories which often overlap.

2.3.1. Viewing and Mapping

Viewing and mapping data aren't necessarily the same thing. Some applications are intended only for visualizing data, while others target map production. Map production is more focused on a high-quality visual product intended for print. In the case of this book, viewing tools are used for visually gathering information about the map data—how the data is laid out, where (geographically) the data covers, comparing it to other data, etc.

Mapping tools are used to publish data to the Internet through web mapping applications or web services. They can also be used to print a paper map. The concepts of viewing and mapping can be grouped together because they both involve a graphic output/product. They tend to be the final product after the activities in the following categories are completed.

2.3.2. Analysis

Just viewing maps or images isn't usually the final goal of a project. Certain types of analysis are often required to make data visualization more understandable or presentable. This includes data classification (where similar features are grouped together into categories), spatial proximity calculations (features within a certain distance of another), and statistical summary (grouping data using statistical functions such as average or sum). Analysis tends to summarize information temporarily, whereas manipulating data can change or create new data.

2.3.3. Creating and Manipulating

This category can include creating features, which uses a process often referred to as digitizing. These features may be created as a result of some sort of analysis. For example, you might keep features that are within a certain study area.

You can manipulate data with a variety of tools from command-line programs to drag and drop-style graphical manipulation. Many viewing applications can't edit features. Those that can edit often create new data only by drawing on screen (a.k.a. digitizing) or moving features. Some products have the ability to do more, such as performing buffering and overlap analysis or grouping features into fewer, larger pieces. Though these are common in many commercial products, open source desktop GIS products with these capabilities are just starting to appear.

2.3.4. Conversion

Certain applications require data to be in certain file or database formats. This is particularly the case in the commercial world where most vendors support their own proprietary formats with marginal support for others. This use of proprietary data formats has led to a historic dependency upon a vendor's product. Fortunately, recent advances in the geomatics software industry have led to cross-application support for more competitor formats. This, in turn, has led to interoperable vendor-neutral standards through cooperative organizations such as the Open Geospatial Consortium (OGC). The purpose of the OGC and their specifications are discussed in more detail in Chapter 12.

Source data isn't always in the format required by viewing or manipulating applications. If you receive data from someone who uses a different mapping system, it's more than likely that conversion will be necessary. Output data that may be created by manipulation processes isn't always in the format that an end user or client may require. Enter the role of data conversion tools that convert one format into another.

Data conversion programs help make data available in a variety of formats. There are some excellent tools available,

Chapter 3. Converting and Viewing Maps

While presenting maps on the Web is fantastic, the data for those maps has to come from somewhere. You'll also want to have a toolkit for creating or modifying maps to fit your needs, especially if you're developing in an environment that isn't already GIS-oriented. This chapter introduces end-user applications for viewing and sharing data as well as low-level tools for data access and conversion.

While many other open source GIS and mapping tools exist, this chapter covers the small selection used throughout the remainder of this book. While many other excellent options exist, the sample of tools described here are robust enough for professional use. These free and open source GIS/mapping products are successfully used throughout industry, government, and academia.

For more general information and links to other tools, see the following reference web sites:

If you have the funds or already have the tools, you can, of course, use proprietary GIS software to create the data you'll be presenting with open source software. One of the best features of the open source tools is their ability to work with data created by proprietary applications and stored in proprietary formats.

3.1. Raster and Vector

The terms raster and vector are used throughout this chapter. They both refer to specific types of data. *Raster data* is organized as a matrix or grid that has rows and columns; each row/column intersection is a cell or pixel. Each cell has a value, for example, an elevation. Images and digital elevation models are rasters. They are a specific number of pixels high and wide, with each pixel representing a certain size on the ground; for example, Landsat satellite images are 185 x 185 km in size. Each pixel is 30 x 30 m in size.

Vector data is represented as coordinates that define points or points that are strung together to make lines and polygons. This data often has an associated table of information, one for every feature (point, line, or polygon) in the dataset. Keeping these distinctions in mind will help you better understand the remaining parts of this chapter.

3.2. OpenEV

OpenEV is a powerful open source desktop viewer. It allows users to explore data in any of the image and vector data formats supported by the Geospatial Data Abstraction Library (GDAL/OGR), which will be introduced later in the chapter. OpenEV can be used for custom image analysis as well as drawing simple maps and creating new data. OpenEV comes as part of the FWTools package, available at http://fwtools.maptools.org.

Many users overlook OpenEV's ability to view and navigate in real time through images draped over 3D landscapes. Figure 3-1 shows an example of an image being viewed with OpenEV, and Figure 3-2 shows the same image but modified with one of the OpenEV image enhancement tools.

OpenEV is a good example of how GDAL/OGR capabilities can be accessed through other programming languages. OpenEV is built using Python, which makes OpenEV extensible and powerful because it has the flexibility of that language running behind it. The most powerful feature of OpenEV may be its ability to use Python capabilities within OpenEV; this allows access to GDAL/OGR functions and to any other Python module you need.

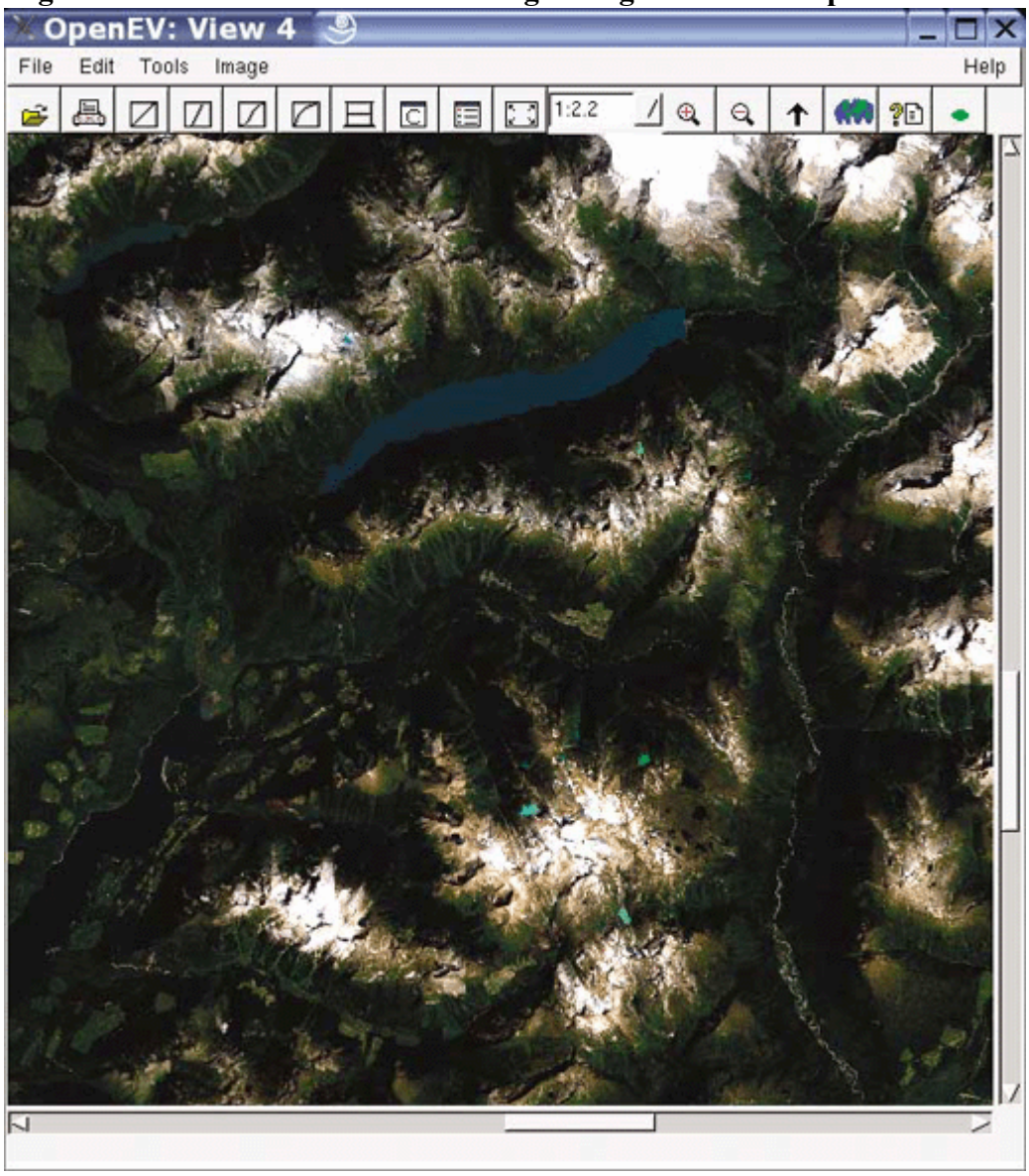
3.3. MapServer

MapServer is the primary open source web mapping tool used in this book. The main MapServer site is at http://mapserver.gis.umn.edu.

There are numerous reasons people decide to use MapServer. One is the ability to make their mapping information broadly accessible to others, particularly over the Internet. Many GIS and mapping analysts need to create custom mapping products for those they support or work for; MapServer makes it possible for users to create maps without needing particular tools installed or assistance from mapping analysts. This in turn reduces the pressure on specialized staff.

Others come to MapServer because it is one of few solutions available for those with diverse data formats. MapServer, through the use of libraries such as GDAL/OGR, can access various data formats without data conversion.

Figure 3-1. A raw Landsat satellite image being viewed with OpenEV



Consider that you could have a collection of 10 different sets of mapping data, all of which need to appear on the same map simultaneously without any of the data being converted from its native format. The native formats can include those used by different commercial vendors. ESRI shapefiles, Intergraph Microstation design files (DGN), MapInfo TAB files, and Oracle spatial databases can all be mapped together without conversion. Other nonproprietary formats can be used as well, including the OGC standards for Geography Markup Language (GML), Web Map Server (WMS), Web Feature Server (WFS), and PostGIS and other databases. The ability to have

3.4. Geospatial Data Abstraction Library (GDAL)

GDAL is part of the FWTools package available at http://fwtools.maptools.org. GDAL's home page (http://www.gdal.org) describes the project as:

...a translator library for raster geospatial data formats... As a library, it presents a single abstract data model to the calling application for all supported formats.

GDAL (often pronounced goodle) has three important features. First, it supports over 40 different raster formats. Second, it is available for other applications to use. Any application using the GDAL libraries can access all its supported formats, making custom programming for every desired format unnecessary. Third, prebuilt utilities help you use the functionality of the GDAL programming libraries without having to write your own program.

These three features offer a powerhouse of capability: imagine not worrying about what format an image is in. With GDAL supporting dozens of formats, the odds are that the formats you use are covered. Whether you need to do data conversion, display images in your custom program, or write a new driver for a custom image format, GDAL has programming interfaces or utilities available to help.

3.4.1. Raster Formats Supported by GDAL

GDAL supports dozens of raster formats. This list is taken from the GDAL web site formats list page found at http://www.gdal.org/formats_list.html.
Arc/Info Binary Grid (*.adf*)Microsoft Windows Device Independent Bitmap (*.bmp*)BSB Nautical Chart Format (*.kap*)VTP Binary Terrain Format (*.bt*)CEOS (Spot, for instance)First Generation USGS DOQ (*.doq*)New Labelled USGS DOQ (*.doq*)Military Elevation Data (*.dt0*, *.dt1*)ERMapper Compressed Wavelets (*.ecw*)ESRI *.hdr* labeledENVI *.hdr* labeled RasterEnvisat Image Product (*.n1*)EOSAT FAST FormatFITS (*.fits*)Graphics Interchange Format (*.gif*)Arc/Info Binary Grid (*.adf*)GRASS RastersTIFF/GeoTIFF (*.tif*)Hierarchical Data Format Release 4 (HDF4)Erdas Imagine (*.img*)Atlantis MFF2eJapanese DEM (*.mem*)JPEG, JFIF (*.jpg*)JPEG2000 (*.jp2*, *.j2k*)NOAA Polar Orbiter Level 1b Data Set (AVHRR)Erdas 7.x .LAN and .GISIn Memory RasterAtlantis MFFMulti-resolution Seamless Image databaseNITFNetCDFOGDI BridgePCI *.aux* labeledPCI Geomatics database filePortable Network Graphics (*.png*)Netpbm (*.ppm*, *.pgm*)USGS SDTS DEM (**CATD.DDF*)SAR CEOSUSGS ASCII DEM (*.dem*)X11 Pixmap (*.xpm*)

This list is comprehensive but certainly not static. If a format you need isn't listed, you are encouraged to contact the developers. Sometimes only a small change is required to meet your needs. Other times it may mean your request is on a future enhancement waiting list. If you have a paying project or client with a particular need, hiring the developer can make your request a higher priority. Either way, this is one of the great features of open source software development—direct communication with the people in charge of development.

All these formats can be read, but GDAL can't write to or create new files in all these formats. The web page shown earlier lists which ones GDAL can create.

3.4.2. Programming Libraries

As mentioned earlier, an important feature of GDAL is its availability as a set of programming libraries. Developers using various languages can take advantage of GDAL's capabilities, giving them more time to focus on other tasks. Custom programming to support formats already available through GDAL isn't necessary: reusability is a key strength of GDAL.

GDAL's application programming interface (API) tutorial shows parallel examples of how to access raster data using C, C++, and Python. You can also use the Simplified Wrapper and Interface Generator (SWIG) to create interfaces for other programming languages such as Perl, Java, C#, and more. See http://www.swig.org/ for more information on SWIG.

The ability to directly link to GDAL libraries has helped add features to an array of GIS and visualization programs

3.5. OGR Simple Features Library

OGR is part of the FWTools package that's available at http://fwtools.maptools.org. The OGR project home page (http://www.gdal.org/ogr) describes OGR as:

... a C++ open source library (and command line tools) providing read (and sometimes write) access to a variety of vector file formats including ESRI Shapefiles, S-57, SDTS, PostGIS, Oracle Spatial, and Mapinfo mid/mif and TAB formats.

The historical definition of the acronym OGR is irrelevant today, but it's used throughout the code base, making it difficult to change.

OGR supports more than 16 different vector formats and has utilities similar to GDAL's raster utilities.

3.5.1. Vector Formats Supported by OGR

The following list of the vector data formats supported by OGR was taken from the OGR formats web page at http://www.gdal.org/ogr/ogr_formats.html. The web page also shows which formats can be written or only read by OGR.
Arc/Info Binary CoverageESRI shapefileDODS/OPeNDAPFMEObjects GatewayGMLIHO S-57 (ENC)Mapinfo fileMicrostation DGNOGDI vectorsODBCOracle SpatialPostgreSQLSDTSUK .NTFU.S. Census TIGER/LineVRT: Virtual Datasource

OGR is part of the GDAL/OGR project and is packaged with GDAL. GDAL deals with raster or image data, and OGR deals with vector data. GDAL is to painting as OGR is to connect-the-dot drawings. These data access and conversion libraries cover the breadth of mapping data.

3.5.2. OGR Utilities and Examples

Like GDAL, OGR consists of a set of libraries that can be used in applications. It also comes with some powerful utilities:

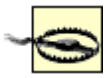
ogrinfo

Interrogates a vector dataset and gives information about the features. This can be done with any format supported by OGR. The following code shows ogrinfo being used to show information about a shapefile:

```
# ogrinfo -summary placept.shp placept

Had to open data source read-only.

INFO: Open of `placept.shp'

using driver `ESRI Shapefile' successful.


Layer name: placept

Geometry: Point

Feature Count: 497

Extent: (-140.873489, 42.053455) - (-52.808067, 82.431976)
```

3.6. PostGIS

PostgreSQL is a powerful enterprise-level relational database that is free and open source but also has commercial support options. It is the backbone of data repositories for many applications and web sites. Refractions Research (http://www.refractions.net) has created a product called PostGIS that extends PostgreSQL, allowing it to store several types of geographic data. The result is a robust and feature-rich database for storing and managing tabular and geographic data together. Having this ability makes PostgreSQL a spatial database, one in which the shapes of features are stored just like other tabular data.



PostgreSQL also has several native geometry data types, but according to Refractions, these aren't advanced enough for the kind of GIS data storage they needed. The PostGIS functions handle the PostGIS geometry types and not the native PostgreSQL geometry types.

This description is only part of the story. PostGIS isn't merely a geographic data storage extension. It has capabilities from other projects that allow it to manipulate geographic data directly in the database. The ability to manipulate data using simple SQL sets it ahead of many commercial alternatives that act only as proprietary data stores. Their geographic data is encoded so that only their proprietary tools can access and manipulate the data.

The more advanced PostGIS functions rely on an underlying set of libraries. These come from a Refraction project called Geometry Engine Open Source (GEOS). GEOS is a C++ library that meets the OGC specification for Simple Features for SQL. GEOS libraries can be used in custom applications and were not designed solely for use with PostGIS. For more information on GEOS, see http://geos.refractions.net/.

3.6.1. GIS Analysis with SQL

PostGIS allows you to use SQL statements to manipulate and create geographic data—for example, to buffer points and create circles. This is just the tip of the iceberg. PostGIS can be a GIS in and of itself while at the same time, all the power of PostgreSQL as a tabular database is available. GIS overlays, projecting and reprojecting of features into other coordinate systems, and spatial proximity queries are all possible using PostGIS. It is possible to have all the standard GIS overlay and data manipulation processes available in a server-side database solution. Example 3-1 illustrates the marriage of SQL and GIS capabilities by selecting points contained by another shape. More examples are shown in Chapter 13.

Example 3-1. An SQL command that takes a polygon shape from one table and finds all the points that are within a feature in another table

```
> SELECT town_name

  FROM towns, ontario

    WHERE Contains(ontario_polygon,#first feature

                town_points);     #containing the others


     town_name
```

3.7. Summary of Applications

Table 3-1 summarizes the functions of each application discussed in this chapter: one checkmark shows peripheral use; two checks denotes common use.

Table 3-1. Summary of the types of functions each application generally plays

	GDAL	OGR	PostGIS	OpenEV	MapServer
Viewing and mapping	✓			✓✓	✓✓
Analysis	✓		✓✓	✓	✓
Manipulation	✓	✓	✓✓	✓	
Conversion	✓✓	✓✓			
Sharing			✓		✓✓

Though the applications may fit more than one category, this table is intended to show the most common ways they are used. For some tasks, there are several applications that can meet your needs. Ultimately, your end goal will help determine which is best to use.

Chapter 4. Installing MapServer

Whether you are preparing static map images for a web page or publishing an interactive web site, MapServer can do the job. Interactive maps allow users to zoom in to particular areas and turn map layers on or off. Applications can be made that let the user select a feature on the map and link it to other data. The possibilities are endless, but the final product is usually a point, click, and view process. The user identifies an area to view or a layer to look at, and the web page is updated with the requested content. If you need a static map, you can save the map for later use.

You can use MapServer from the command line or, for interested programmers, through an API. MapServer also can be used as a common gateway interface (CGI) application or scripted using common web programming languages such as PHP, Perl, Python, and Java. Whether using the CGI version or scripting your own, MapServer's runtime configuration files control what layers are shown and how they are drawn. Mapping data can easily be added to an existing application by editing the configuration file.

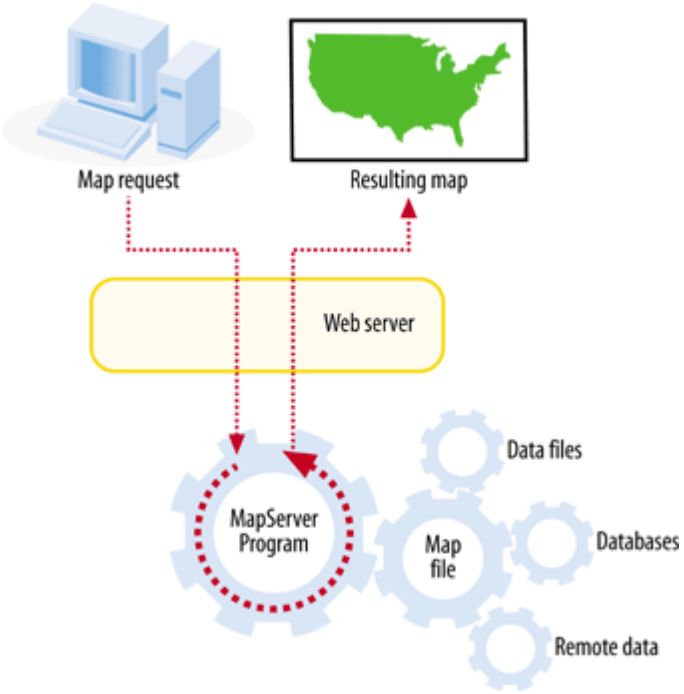
This chapter discusses what MapServer is, how it can be used, and how it works. It also covers how to set up the underlying programs that make up MapServer, so it can be used in custom applications. Examples of MapServer application development are given in Chapters 10 and 11.

4.1. How MapServer Applications Operate

MapServer usually works behind a web server application. The web server receives requests for maps and passes them to MapServer to create. MapServer generates the requested map image and hands it to the web server, which transmits it back to the user. Figure 4-1 shows how the user interacts with the web server which, in turn, makes requests to the MapServer program.

MapServer's primary function is reading data from various sources and pulling these layers together into a graphic file, also known as the *map image*. One layer may be a

Figure 4-1. A diagram showing the basic operation of a MapServer application



satellite image, another the outline of your country or points showing a major city. Each layer is overlaid or drawn on top of the others and then printed into a web-friendly graphic for the user to see. A good example of the results of the overlapping and mapping process can be seen in Figure 4-2. You can see a satellite image (from a remote server), road lines, and city locations; the city labels are dynamically generated by MapServer.

This drawing process (a.k.a. *rendering*) occurs each time a request for a new map is made to MapServer, for instance, when a user zooms into the map for a closer look. This process also occurs when a user manually requests a redraw, such as when the content of one data layer changes, and the user wants to see the change.

4.2. Walkthrough of the Main Components

MapServer produces output graphic files based on the input requests from the user and how the map is defined. Key components include the MapServer executable or CGI program, a map file, data sources and output images. Figure 4-3 shows how all these components work together: after a user request, the MapServer CGI program accesses a map file, draws information from the data sources, and returns an image of the map.

Figure 4-2. A map showing various layers of information

4.2.1. MapServer Executable

The simplest form of MapServer runs as an executable CGI application on a web server. Technically, MapServer is considered an HTTP-based stateless process. Stateless means that it processes a request and then stops running. A CGI application receives requests from a web server, processes them, and then returns a response or data to the web server. CGI is by far the most popular due to its simplicity: no programming is required to get it working. You edit the text-based, runtime configuration file, create a web page, and then set them up to be served by a web server.

If you are a programmer, you don't need to use MapServer in CGI mode. Instead, you can create custom applications that use the MapServer API. However, this is considered an advanced topic, and an overview is discussed in Chapter 14.

Figure 4-3. Main MapServer application components

4.3. Installing MapServer

MapServer runs on a variety of platforms, and the details for installing it vary depending on where you want to run it and how you want to integrate it with the rest of your system.

4.3.1. Platforms

Many operating systems and processors can run MapServer successfully. A recent survey of MapServer application developers showed dozens of different implementations running on many processor speed and type combinations. Included in the survey results were the following operating systems:

- Windows 2000, XP, 95
- RedHat/SuSE/Debian/Mandrake Linux, Versions 6 through 9
- Solaris
- Mac OS X Panther
- Vmware running Windows and Linux
- FreeBSD
- SCO Open Server
- SGI IRIX

Reported processor speeds were as low as 120 MHz and as little as 64 MB of memory. Others use the latest processing and memory resources available. With the diversity and flexibility required to meet these cross-platform requirements many developers have found MapServer to be the only option for serving web-based maps.

4.3.2. Acquiring MapServer Binaries

Many documents or how-tos include information on compiling MapServer, which can lead you to assume that compiling MapServer from source code is required. For most users it isn't required. Acquiring binaries refers to the process of downloading executables and libraries that are ready to run on your operating system of choice, without compiling from source code.

4.3.2.1 Windows versions

MapServer application designers wanting to use Microsoft Windows to run MapServer can download standard Windows packages that include everything you need to get started. The packages include all the required MapServer programs zipped into a single file for download. Packages are available from a few locations, including the main MapServer web site (http://mapserver.gis.umn.edu/win32binaries.html), the MapTools.org site (http://dl.maptools.org/dl/), or from other users via mailing list discussions.

The MapTools.org web site has the easiest-to-use package and is highly recommended for inexperienced

4.4. Getting Help

During setup and installation of MapServer, there are different ways to get some help. These include commercial support, mailing lists, and an Internet relay chat (IRC) channel.

MapServer web site

- The main web site with documentation is at http://mapserver.gis.umn.edu.
- An FAQ is available at http://mapserver.gis.umn.edu/doc/mapserver-FAQ.html.
- Bug tracking/reporting can be found at http://mapserver.gis.umn.edu/bugs/index.cgi.

MapServer mailing lists

- Look for MapServer users at http://lists.umn.edu/archives/mapserver-users.html.
- Find older archives at http://mapserver.gis.umn.edu/cgi-bin/wilma/mapserver-users.
- Check out a searchable GMANE archive at http://dir.gmane.org/gmane.comp.gis.mapserver.user.
- Find MapServer-dev at http://lists.gis.umn.edu/mailman/listinfo/mapserver-dev.

MapServer IRC channel

- Look for #mapserver at irc.freenode.net.

Other references

- A map of other users/consultants is at http://moximedia.com/imf-ows/imf.jsp?site=ms_users.
- A support site that includes MapServer category is at http://www.findopensourcesupport.com.
- Check out the Mapping Hacks book (O'Reilly) at http://mappinghacks.com.

Table 4-2 lists commercial options for support or development of MapServer applications.

Chapter 5. Acquiring Map Data

You are probably eager to start on a project and try out all of these great tools. However, before you can jump in, you have some homework to do. The first step in any mapping project is identifying your data needs. Finding the information you require is your next step.

5.1. Appraising Your Data Needs

What you need will vary greatly depending on your goal, so you should first determine the goal or final product of your work. Along with this comes the need to assess what kind of map information you want to portray and at what level of detail. The following sections are a guide to thinking through all your requirements before starting a project.

5.1.1. Vector Maps or Raster Maps?

If you want to make a custom map, you need to determine what kind of map data you require. There are two kinds of maps, each requiring different kinds of map data.

A vector map shows features made up of points, lines, or polygon areas, as in Figure 5-1. For instance, this can be a local road map or a global map showing locations of power stations and their respective countries.

An image or raster map is made from data sources such as aerial photography, satellite imagery, or computer-generated images of Earth's surface. Figure 5-2 shows an example of a raster map made from weather satellite data.

One variant of the raster map is the scanned map. When a paper map is scanned, it is converted into a digital raster map and can then be used as a layer on other maps. This is one way to make mapping data more readily available without having to recreate the maps using digital sources.

Figure 5-1. A vector map showing roads, country borders, and some major cities in North America

Figure 5-2. A raster map showing cloud cover over North America

5.2. Acquiring the Data You Need

Once you have determined your mapping and data needs, you have to find the appropriate data. To find such data, you can, for example, search the Internet, trade with friends, or download from a government site. This chapter takes you through several examples from searching for premade maps to finding data to do your own mapping and conversion.

5.2.1. Finding Premade Maps

Sometimes all you want is a premade, basic map of the world or a specific country. The Internet can be a good place to find these. One excellent site is the CIA World Factbook at http://www.cia.gov/cia/publications/factbook/geos/ag.html. Here you will find a basic political map for every country and also a detailed demographic, cultural, and political summaries. Large-size, print-quality reference maps are also available in Portable Document Format (PDF).

Another good resource is the University of Texas Online Map Library at http://www.lib.utexas.edu/maps/. With a brief walk through some of the links, you can find diverse maps that show, for example, a street map of Baghdad or a shaded relief map of the United Kingdom, nicely showing the Scottish Highlands. The library has scanned many of these maps and made them available to the public domain.

5.2.2. Finding Satellite Images or Shaded Surface Maps

Premade maps from images or shaded elevation models are also available. The CIA Factbook (http://www.cia.gov/ciapublications/factbook) has small political maps for each country. Other reference maps that show more physical data (e.g., bathymetry and elevation) or even photographic examples of the earth can be found. The U.S. National Geophysical Data Center (NGDC) includes some beautiful premade gallery images of their more detailed source data; look for them at http://www.ngdc.noaa.gov/seg/topo/globegal.shtml.

5.2.3. Finding Map Data

The Internet contains a wealth of geographic information for your projects. It is a natural place to begin your quest for mapping data. Some government agencies make their mapping data freely available on the Internet. Certain organizations have even made it their mission to make mapping data available for others for free. Other groups have created catalogs of mapping data you can search.

Plenty of Internet map data sources are available for the United States. North American data is also widely accessible, with Canada having some excellent products. Global data is more of a challenge. International organizations such as the United Nations or even American-based organizations that observe global events, such as the National Oceanic and Atmospheric Administration (NOAA), are some of the better sources for global data.

There are a variety of data repositories on the Internet to choose from, and they each have a certain target scale of application that can't be ignored. For example, a satellite image can show pictures of where you live. Figure 5-6, for example, shows the different colors of surfaces such as roads, fields, and waterways, but only to a certain level of detail. If you hope to use it for planning a walk to the corner store, you will be sorely disappointed. However, if you want to see how your city is laid out, you will probably be satisfied. Figure 5-7 shows a closer look at the same image as Figure 5-6. The maximum amount of detail has been reached, and the image looks bad. This is an attempt to use the image beyond its intended scale of application.

There are multiple ways to hunt for good data. You can announce your search to email mailing lists or review web sites that summarize free data sources. Local mapping companies and government agencies are also a good place to start. Many states, provinces, and municipalities have their own GIS and mapping offices. Your search can also include local consulting companies, industries, or nonprofit organizations related to environmental or natural resource management and use. Just ask to speak to someone in the mapping or GIS department. Most people in these organizations

Chapter 6. Analyzing Map Data

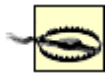
Finding data is helpful, but sometimes you need to explore it or do basic manipulation before you're ready to use it in your own maps. This chapter will lead you through acquiring the tools and data for some sample projects. Some of this sample data will be used in examples in other chapters as well. The techniques shown here don't produce lovely maps by themselves, but they can save you immense frustration when you go to use data to create maps.

6.1. Downloading the Demonstration Data

This section will walk you through the process of acquiring some fairly simple data for the U.S. state of Minnesota. In order to keep things simple, you will use a well-known MapServer demonstration dataset. The MapServer web site always has copies of the latest versions of MapServer software available as well as demonstration datasets and applications.

First, go to http://mapserver.gis.umn.edu/dload.html. Beside the latest download version file, you will see a link to demo. This links to a ZIP file called workshop.zip.

The full path is http://maps.dnr.state.mn.us/mapserver_demos/workshop.zip. Now, download, and unzip the contents of this file. All you really need is the workshop/data folder.



Where you unzip doesn't matter, but you need to ensure that the folders contained in the archive are recreated when unzipped. This is usually the default for your archive software. In Winzip, this option is called Use Folder Names.

6.2. Installing Data Management Tools: GDAL and FWTools

If you examine the contents of the data folder you will find 49 files. These are map data files in ESRI shapefile format and GeoTIFF images. To learn more about them, you will use a tool from the GDAL project. GDAL comes bundled with the FWTools package, as does the OpenEV viewer and other utilities used in this book.

You can acquire the latest version of FWTools for your operating system or distribution from http://fwtools.maptools.org/. Examples used in this book are for the Linux operating system, but for Windows users, I have included notes where steps or usage varies.

Windows users

FWTools comes as an installation/setup file. First, download and execute the file. It should walk you through the setup steps. The setup program should place an FWTools Shell icon on your desktop. Launching this icon gives you a command prompt window for running FWTools commands.

Linux users

First, download the most recent tarball version of FWTools (e.g., *FWTools-linux-0.9.8.tar.gz*), and unzip/untar it into any folder you have handy. Linux users will be working from the command prompt/shell console to enter commands.

Now, open a console, and move into the folder that the file was saved in. This can be done using the cd command followed by the path to the folder. You can unzip the file by running:

```
> tar -xzvf FWTools-linux-0.9.8.tar.gz.
```

Run the installer by typing ./install.sh. This prepares the installation and creates a folder called bin_safe. Now, run the command:

```
> . /fwtools_env.sh
```

(Note the space after the period.) This command loads all the environment variables and settings, even though it might appear to not do anything. You are now ready to use FWTools commands.

You will be running some of programs to manipulate, examine, and convert the MapServer demo data. In the examples, commands that you type are prefixed by the greater than symbol, >. This isn't part of the command and isn't typed, but it indicates that everything after it is. The > symbol is how the command prompt tells you it is ready for another command. The command prompt symbol varies depending on operating system.

6.3. Examining Data Content

Examining data content is an important part of any project. Understanding information specific to your dataset will help you use it more effectively. Each piece of spatial data will have some geographic component to it (coordinates describing the location of real features), but it will also have what are called attributes.These are non-geographic data about the geographic feature, such as the size of population, the name of a building, the color of a lake, etc. You will often hear the geographic coordinate data described as spatial data and the attribute information referred to as tabular, attribute, or nonspatial data. It is equally valid to call any dataset spatial if it has some geographic component to it.

6.3.1. Viewing Summary Information About Airports

The MapServer demo data includes a variety of vector spatial files; therefore you will use the ogrinfo tool to gather information about the files. At the command prompt, change into the workshop folder, and run the ogrinfo command to have it list the datasets that are in the data folder. The output from the command will look like Example 6-1.

Example 6-1. Showing a list of the layer names available in a folder containing shapefiles

```
> ogrinfo data

INFO: Open of 'data'

using driver 'ESRI Shapefile' successful.

1: twprgpy3 (Polygon)

2: rmprdln3 (Line String)

3: lakespy2 (Polygon)

4: stprkpy3 (Polygon)

5: ctyrdln3 (Line String)

6: dlgstln2 (Line String)

7: mcd90py2 (Polygon)

8: twprdln3 (Line String)

9: plsscpy3 (Polygon)

10: mcdrdln3 (Line String)

11: majrdln3 (Line String)

12: drgidx (Polygon)

13: airports (Point)

14: ctybdpy2 (Polygon)
```

This shows that there are 14 layers in the data folder (the order of the listing may vary on other systems). You can also see that the folder contains ESRI shapefile format files. Each shapefile is a layer in this listing. If you look at the files located in the data folder, you will see that there are way more than 14 files. This is because a shapefile consists of more than one file: one holds spatial data, another holds tabular data, etc.

A summary of more information for each layer can be seen by adding the name of the layer to the ogrinfo command and a -summary parameter, as shown in Example 6-2.

6.4. Summarizing Information Using Other Tools

While ogrinfo and other ogr utilities are powerful tools, basic text-processing tools such as sort, uniq, wc, and sed can give them an extra bit of flexibility. The tools here are readily available for Unix-type operating systems (like Linux) by default. They are also available for other operating systems but you may need to download a package (e.g., from http://gnu.org) to get them for your system.

Each command can receive text streams. In this case, the text stream will be the lines of information coming from ogrinfo and listed on the screen. These commands take in those lines and allow you to, for example, show only certain portions of them, to throw away certain lines, reformat them, do a search/replace function or count items. Many types of functions can be done using the ogrinfo -sql parameter, but the ultimate formatting of the results isn't always what is desired. These examples show some common patterns for extracting specific information and generating more custom stats.

6.4.1. Setting Up Processing Tools for Non-GNU Platforms

These text-processing tools are sometimes packaged together, but are usually separate projects in and of themselves. Most of them were formed as part of the GNU/Free Software Foundation and are registered with the GNU free software directory at http://www.gnu.org/directory/. The targets of GNU software are free operating systems, which can cause some problems if you are dependent on an operating system such as Microsoft Windows. Some operating system don't normally include these tools, but they can often be acquired from Internet sources or even purchased.

A very comprehensive set of these tools for Windows is available at http://unxutils.sourceforge.net/. You can download a ZIP file that contains all the programs. If you unzip the file and store the files in a common Windows folder, such as C:\Windows\System32 or C:\winnt\System32, they will be available to run from the command prompt.

If the tool or command you want isn't included, the next place to look is the GNU directory (http://gnu.org). This is where to start if you are looking for a particular program. A home page for the program and more information about it are available. Look for the download page for the program first to see if there is a binary version of the tool available for your operating system. If not, you may need to download the source code and compile the utility yourself.

Another resource to search is the Freshmeat web site at http://freshmeat.net. This site helps users find programs or projects and also provides daily news reports of what is being updated. Many projects reported in Freshmeat are hosted on the Sourceforge web site at http://sourceforge.net.

One source that is commonly used on Windows is the Cygwin environment, which can be found at http://www.cygwin.com. The web site describes Cygwin as "a Linux-like environment for Windows." Cygwin can be downloaded and installed on most modern Windows platforms and provides many of the text-processing tools mentioned previously. Furthermore, it also provides access to source-code compilers such as GCC.

Mac OS X includes many of the same kinds of text-processing tools. They may not be exactly the same as the GNU programs mentioned here, but similar alternatives are available in the Darwin core underlying OS X. For ones that aren't available natively in OS X, they can be compiled from the GNU source code or acquired through your favorite package manager such as Fink.

6.4.2. Using ogrinfo to List Data in a Shapefile

The standard output of ogrinfo reports are a set of lines displaying information about each feature. As earlier, this output is quite verbose, showing some summary information first, then sections for each feature. In the case of the airport data, each airport has its own section of seven lines. Example 6-10 shows a couple of these sections covering 2 of the 12 features (the rest were removed to reduce unnecessary length).

Example 6-10. Basic output listing about the airports shapefile

```
> ogrinfo data airports
```

Chapter 7. Converting Map Data

Chapter 2 outlined the various tasks of digital mapping; one of those tasks involved converting data between different formats. In a perfect world there might only be one or two formats, but in reality there are dozens of different ways to store spatial information. Add to that both raster and vector data formats, and you've got quite an array of options. Like Chapter 6, this chapter by itself won't likely create beautiful maps, but the steps presented here are even more important because they are a key part of the path to producing useful maps.

The examples in this chapter use the GDAL/OGR utilities that are introduced in Chapter 6. They are part of the FWTools package.

7.1. Converting Map Data

Different data formats have sprung up because of the need for various capabilities and the desire for proprietary software control. Creating a better data format may be necessary to handle more complex information or improve performance, but having various data formats just because each vendor creates one for its own product is hardly productive. Fortunately, there are some excellent tools available for both raster and vector data conversion and for coordinate reprojection. By their natures, raster and vector data need to be handled differently, though the general concept is similar: take data in one format, and convert it to another format.

Data conversion can be necessary for a number of reasons:

- The source data may be in a format you aren't familiar with.
- Your mapping program may not know how to handle a particular format.
- You may just want to get a quick image snapshot in a common, non-GIS format.

7.2. Converting Vector Data

Quite often data isn't in a format that makes it readily available for both mapping software and people to read. Many data formats store the geographic data in a binary format. This format is normally readable only by computers and is designed for software to use. Some spatial data formats are in a simple text format, which is easier to explore.

7.2.1. Geography Markup Language (GML)

One example of a text-based format is Geography Markup Language (GML). It uses a text syntax to encode coordinates into a text file. GML can then be read or manually edited without needing a special piece of software or at least nothing more than a common text editor. Creating GML from scratch isn't very pleasant. Fortunately, another OGR utility exists that can convert OGR-supported data formats into and out of other formats, including GML.

GML has three different versions: GML1, GML2, and GML3. There are many sub-versions as well. The differences between versions can be a problem because certain features may not be directly transferable to another. The tools introduced in this chapter use ogr2ogr, which outputs to GML2 format.

GML was designed to be suitable for data interoperability, allowing the exchange of spatial data using a common format. This has opened up the possibilities for various web services that operate using different software yet can communicate using this standard format. See Chapter 12 to learn more about this example.

GML's downside is its size. Text files can't hold as much raw computer data as a binary file, so GML files tend to be much larger than other data formats. It isn't ideal to store all your data as GML, but it's perfect for sending data to someone who may not be able to support other formats. Because GML is text, it compresses very well using tools such as gzip and WinZip.

Extensible Markup Language (XML) is a popular standard for general data exchange, especially through the Internet. XML-based datafiles (such as GML) typically have an associated schema document. Schema documents aren't discussed, but you should know that they do exist. They describe how the data is structured in the XML data file—what attributes they have, what version of XML is being used, what type of information is in an attribute, etc. Schema documents have a filename suffix of .xsd. Therefore, a GML file called airports.gml also has an associated airports.xsd file describing the contents of airports.gml.

7.2.2. Converting a Shapefile to GML

Example 7-1 shows how to convert the airports data from shapefile format to GML using the ogr2ogr conversion utility.

Example 7-1. Using ogr2ogr to convert a shapefile into GML format

```
> ogr2ogr -f "GML" airports.gml data/airports.shp airports

> more airports.gml

<?xml version="1.0" encoding="utf-8" ?>

<ogr:FeatureCollection

     xmlns:xsi="http://www.w3c.org/2001/XMLSchema-instance"

     xsi:schemaLocation=". airports.xsd"

     xmlns:ogr="http://ogr.maptools.org/"

     xmlns:gml="http://www.opengis.net/gml">

  <gml:boundedBy>
```

7.3. Converting Raster Data to Other Formats

Raster data, like vector data, can be stored in numerous formats. Some formats come from certain software requirements or from certain satellite image sources. Some are more general while others are better suited for more complex tasks. In this section, you will see how to use the gdal_translate raster translation utility and its various parameters to help you draw the most value out of an image.

7.3.1. Translating an Image to Another Format

Just as ogr2ogr has several conversion options, so does gdal_translate. The syntax for the command is slightly different but similar in concept. Here is the usage:

```
    gdal_translate <options> <input_image> <output_image>
```

The options available can be listed by running the command without any parameters, as in Example 7-5.

Example 7-5. Checking the options for gdal_translate

```
> gdal_translate

Usage: gdal_translate [--help-general]
       [-ot {Byte/Int16/UInt16/UInt32/Int32/Float32/Float64/
             CInt16/CInt32/CFloat32/CFloat64}] [-not_strict]
       [-of format] [-b band] [-outsize xsize[%] ysize[%]]
       [-scale [src_min src_max [dst_min dst_max]]]
       [-srcwin xoff yoff xsize ysize] [-a_srs srs_def]
       [-projwin ulx uly lrx lry] [-co "NAME=VALUE"]*
       [-gcp pixel line easting northing]*
       [-mo "META-TAG=VALUE"]* [-quiet]
       src_dataset dst_dataset


GDAL 1.2.1.0, released 2004/06/23
```

This first part of the output of Example 7-5 shows the various options. Each item enclosed with brackets [] is optional. Without them, the program simply converts one image to another, creating the output in GeoTIFF format by default. Some options are used in the examples of the following sections and in more depth in other chapters of this book. Example 7-5 also shows the software version number and the official release date of the GDAL project.

Example 7-6 shows a list of all available image formats. This list will vary depending on the operating system and (only if you compiled it yourself) the options you specified during the configuration process. This lists many formats, though it may actually be able to read more image types. The list shows only potential output formats that can be used with the -of option. You supply this option with the abbreviated name for that format as shown in Example 7-6. For the latest capabilities, be sure to see the GDAL support formats page http://www.gdal.org/formats_list.html.

Example 7-6. Listing supported raster output formats

```
The following format drivers are configured and support output:
```

Chapter 8. Visualizing Mapping Data in a Desktop Program

For many people, the prospect of creating a map is very exciting. Viewing a graphical representation of the world, especially a familiar part of the world, can kick-start your imagination and resolve unanswered questions. Where would we be if we couldn't view a map while planning a vacation and reserving the hotel? What would gardeners do if they couldn't look up the planting zone index for their area? How would trade prosper without a clear picture of other countries and trade routes? The questions can go on and on. So can the maps!

This chapter will show you how to use the open source spatial data viewer OpenEV to look at the data we used in the previous chapter. If you follow along, you will be able to create a customized view of Itasca County in Minnesota, U.S.A. By the end of the chapter, you should be comfortable using the tool for basic viewing, color-theming a map, and creating a simple 3D view.

8.1. Visualization and Mapping Programs

This chapter focuses on OpenEV, but there are many other GIS and mapping programs available. Some are open source and some are commercial.

8.1.1. Other Open Source Mapping Programs

Here is a list of some of the free and open source desktop GIS and mapping programs available today:

Quantum GIS (QGIS)

http://www.qgis.org

Java Unified Mapping Platform (JUMP)

http://jump-project.org

User-friendly Desktop GIS (UDIG)

http://udig.refractions.net

Thuban

http://thuban.intevation.org

OpenMap

http://openmap.bbn.com

OpenEV

http://fwtools.maptools.org

For more comprehensive lists of open source GIS and mapping tools, see:

8.1.2. Other Free-to-Use Programs

Many of the major vendors distribute their own free viewers. These aren't considered freeware because you may not be able to freely distribute the programs. However, they are freely available to use:

ArcExplorer

http://www.esri.com/software/arcexplorer/

GeoMedia Viewer

http://imgs.intergraph.com/gviewer/

8.2. Using OpenEV

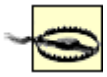
OpenEV was designed as an example of the kinds of tools that can be built on top of various open source GIS libraries. It has a variety of powerful features, is open to complete customization, and includes the ability to add in custom tools written in Python. For the purposes of this chapter, only some of the basic functions (not including Python scripting) will be demonstrated. These include the ability to view various formats of raster and vector data layers, change their colors, create labels, etc. It is also possible to create your own data; check for that in Chapter 9.

Follow along with the examples, and try them yourself to build up your familiarity and comfort with this tool. You will first test your installation of OpenEV, then create some custom maps using the demonstration datasets. Once the basic viewing capabilities are explored, you will learn to create some more sophisticated color classifications, and ultimately produce a 3D model to navigate through.

8.2.1. Installing OpenEV

Chapters 5 and 6 walked you through how to acquire sample data and review it using some simple tools. If you don't already have the MapServer demonstration data and the FWTools downloaded and installed, please refer back to these sections in Chapter 6.

You should have two things: a sample MapServer application known as *Workshop* and a copy of OpenEV that comes with FWTools. To test that you have OpenEV installed, launch the program. Windows users will launch the OpenEV_FW desktop shortcut. This will open a command window and create some windows.



Don't close the command window that appears. OpenEV needs this to stay open. It is safe to minimize the window while using OpenEV.

Linux users will launch the openev executable from the FWTools *bin_safe* folder.

Linux users: if you don't have a *bin_safe* folder, run the installer by typing:

```
> ./install.sh
```

This prepares the installation and creates a folder called bin_safe. Now, run the command:

```
> . /fwtools_env.sh
```

(Note the space after the period.) This loads all the environment variables. Now you should be able to run openev from anywhere.

When starting, the main window titled "OpenEV: View 1" should appear with a black background. The Layers windows will also appear.

8.2.2. Loading Sample Map Data into OpenEV

Starting from the default View 1 screen, you can add in sample data by selecting Open from the File menu, or by clicking on the folder icon on the left side of the main tool bar. The File Open window allows you to find a dataset on your filesystem and load it as a layer in OpenEV. All files are listed in the File Open window, whether or not they are spatial datafiles. OpenEV doesn't attempt to guess what they are until you actually try to open one.

8.3. OpenEV Basics

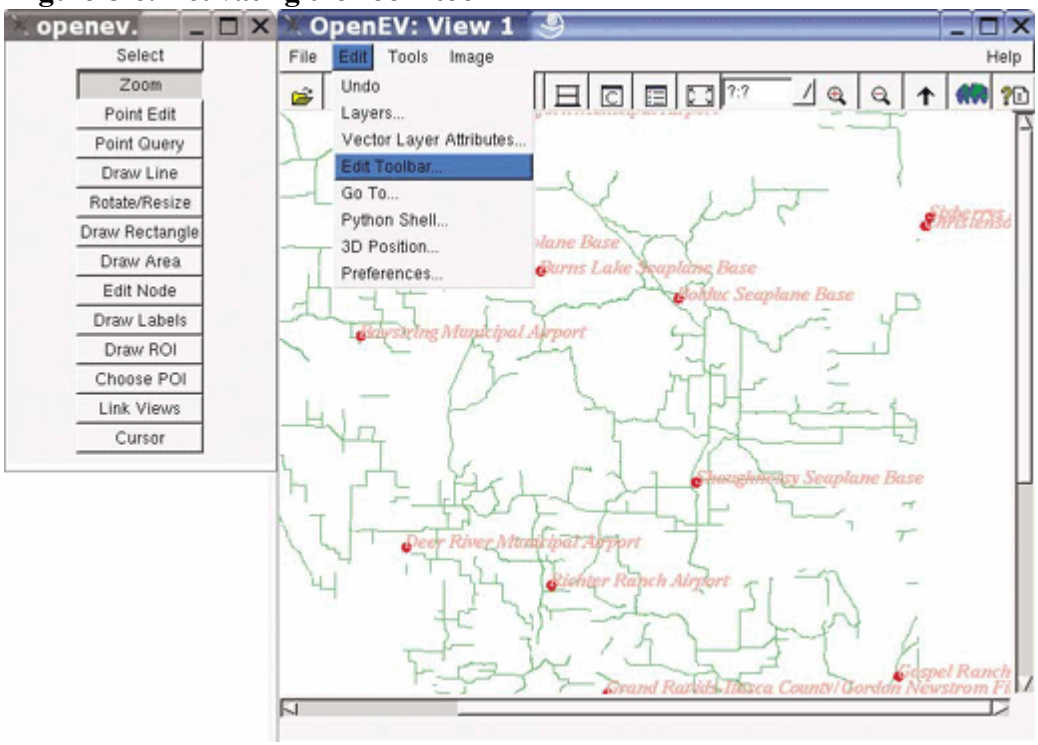
OpenEV has a variety of simple and complex tools. Many of the advanced tools have been designed with raster data in mind, but the basic toolsets and controls include many of the features needed to do quick visualization of both raster and vector data.

8.3.1. Zooming

One the first things you will want to do is zoom in to certain portions of your map. There are a couple ways to do this. The first is to use the plus and minus buttons on the toolbar of the view. The icons are a magnifying glass with a + and - symbol inside them. Clicking + zooms into the center of the screen, and clicking - zooms out from the center of the screen.

The more efficient way is to use the Zoom tool, which allows you to drag a rectangle on the screen and zoom in to that portion of the map. To enable the zoom tool, select the Edit Toolbar from the Edit menu, then choose Zoom from the list, as shown in Figure 8-6.

Figure 8-6. Activating the zoom tool



Now you can return to the view and click and hold the left mouse button to zoom in. You can also drag a rectangle around a portion of the screen and have it zoom in directly to that location. To do this, click and hold the left mouse button, then drag the mouse to another location. Watch as the rectangular zoom box appears, as in Figure 8-7. When the box covers the area you want, let go of the mouse button.

Zooming out can be done a few ways as well. As mentioned earlier, you can use the zoom-out magnifying glass button, but you can also hold the right mouse button down and slowly zoom out. Another method to use, if appropriate, is the Fit All Layers button on the main view tool bar. This button is a rectangle with four internal arrows pointing toward the corners of the rectangle, as shown in Figure 8-8. Clicking this button zooms out so far that all the data in all the layers can be seen.

If you use this tool and find that you can no longer see your data, it is probably because one or more layers are in a different map projection. If you know the layer, remove it from the Layers window, and try Fit All Layers again. There is no tool to return you to the previous

Chapter 9. Create and Edit Personal Map Data

Some projects have more than enough data available to create a useful map, but there are often times when you need to create your own data. This chapter will help guide you through the process of creating your own data from scratch. A few simple tools are all you need to get started. You will learn to use OpenEV with a simple base map to create your own map layers.

9.1. Planning Your Map

Planning your map at the outset can save you tremendous amounts of time later. Evaluating what you want to do, what you have to work with, and what you need to collect will help you make realistic decisions about your project.

9.1.1. Choosing a General Scale and Extent

The size and scale of the map you are making will affect various parts of your project. If you want a wall-size map covering your neighborhood, you will have very different requirements than someone planning a world map for a small book. You need to ensure that the data you create will work well with your intended map. Here are some questions to consider:

What part of the world do you want your map to cover?

You may need global, regional, or local-scale data.

What size of map do you want?

The larger the size, the more detailed the information you will probably want. A small map doesn't show a lot of detail. A large one does.

Will it be interactive or designed for a hardcopy print?

An interactive map gives the reader more options and flexibility. You have to make sure your data can support those options. If you have a global-scale map and the reader can zoom in to their town, will you have detailed data available at that scale? A hardcopy print is less flexible but easier to manage because you work with one scale all the time.

How accurate does the data need to be?

Will the product need to be used for precise navigation, or can you handle some error in your data? You need to know your limits so that you don't disappoint your readers.

9.1.2. Identifying Data Requirements for Your Base Map

When creating new or customized data it helps to have a base map to start with. You need data in the correct scale, covering the right area. You can then draw features on top of the base to create custom data. The data itself is the goal of this chapter, not a final map product.

The base map gives you a reference to the real world. You may have an air photo you can scan or a topographic map image downloaded from the Internet. Your base data doesn't necessarily need to look good if you are just using it as a reference for creating data. For example, highway data can be the reference for roughly locating towns. Since all highways won't be visible at a country-wide map scale, they won't end up on your final map.

When considering your base map requirements, keep in mind that global scale datasets are available but are often very coarse. There are a lot of detailed air photo and road line files available for the United States. You can probably even locate your house. Canada has a handful of satellite images available for most of the country but not to a scale you can use to locate a home. Other countries have varying levels of data available.

Chapter 5 has a list of data sources that can help you find useful base map data.

9.1.3. What Are Your Sources?

9.2. Preprocessing Data Examples

Projects in this chapter focus on the city of Kelowna, Canada. In the summer of 2003, there were massive fires in the area. It was a very destructive fire season, destroying many homes. This area is used as a case study throughout the rest of this chapter as you work through a personal mapping project in the area. Many of the typical problems, like trying to find good base map data, are discussed.

In order to use base map data, you need to do some data preprocessing. This often involves clipping out areas of interest or projecting information into a common coordinate system.

9.2.1. Clipping Out an Area of Interest

The satellite image used for this project covered a much broader area than was needed. Unnecessary portions of the image were clipped off, speeding up loading and making the image a more manageable size.

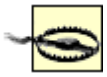
Satellite imagery of Canada is readily available from the government's GeoGratis data site (http://geogratis.cgdi.gc.ca/). The scene for Kelowna is path 45 and row 25; you can grab the image file from the Landsat 7 download directory (/download/landsat_7/ortho/geotiff/utm/045025/). The files are organized and named based on the path, row, date, projection, and the band of data in the file. The file *...743_utm10.tif* contains a preprocessed color composite using bands 7, 4, and 3 to represent red, green, and blue colors in the image.

I used OpenEV to view the file and determine the area of interest. Figure 9-1 shows the whole image or scene.

The large lake in the bottom half of the image is Okanagan Lake. After zooming in to the general area of interest, the Draw Region Of Interest (ROI) tool is used to select a rectangle. The Draw ROI tool is on the Edit Toolbar, which can be found in the Edit menu. To use the ROI tool, select the tool and drag a rectangle over the area. It creates an orange rectangle and leaves it on the screen, as shown in Figure 9-2.

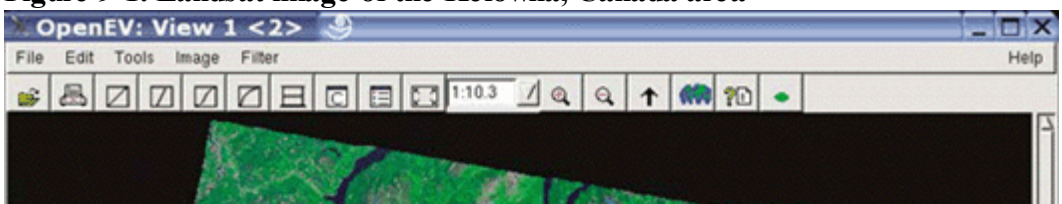
This area is then exported to a new file. To do this, use the File ⟶ Export tool. The Export window provides several options as shown in Figure 9-3. There is a small button near the bottom of the window that shows Advanced Options.

The filename of the image that will be created is entered into the Set Output box. This tool automatically exports the whole image to a new file, but only the selected region needs to be exported. To do this, select the Window Input File button. The four text entry boxes in the Input Window area show numbers that correspond to the pixel numbers of the image that will be exported. Only the pixels in the portion of the image identified as the ROI should be used.



A bug with the software sometimes requires you to redefine your ROI after the Window Input File option is selected. If you want to use the ROI that is already drawn, you can click on one edge of the ROI box in the view; the text boxes are then updated.

Figure 9-1. Landsat image of the Kelowna, Canada area



Chapter 10. Creating Static Maps

Putting map data into an Internet page or interactive web site may be easier than you think. MapServer is often all you need to make it happen. This chapter shows you how to use MapServer to get your data up on a web site as quickly as possible. Even if what you really want is interactive maps, using MapServer to create static maps may be useful at times, and it's a good way to learn about the program.

MapServer is best known for its web mapping capabilities, but it also comes with a set of standalone command-line tools for personal use. These can be used to create static map images for web pages, documents, or email. This chapter starts with a description of each utility, then walks you through several common mapping tasks using a simple map file. Chapter 4 discussed the process of installing or compiling these utilities while Chapter 11 focuses on using MapServer as an interactive web mapping solution.

10.1. MapServer Utilities

MapServer comes with much more than the standard *mapserv* executable file. The following programs are compiled at the same time as the *mapserv* file and are usually included in any MapServer installation package. MapServer is more than just a web mapping program. To demonstrate this, the chapter begins with an introduction to some command-line tools. They can be used as standalone applications for making maps without a web server. Details about each program are included.

legend

Creates a graphic image of a legend based on the symbols used to draw layers specified in a map file.

scalebar

Produces a graphic image of a scalebar based on the options in a map file, using the map extent as set in the map file.

shp2img

Produces a map image based on a map file. Map extents and desired layers can be set. The options are robust enough that a simple desktop custom application could use this tool to create maps. Similar to *shp2pdf*, which outputs a PDF rather than just an image.

shp2pdf

Produces an output PDF file that includes a map based on the input map file. This requires that MapServer was compiled with PDF support and has similar syntax as shp2img.

shptree

Adds an index (quad-tree based) to a shapefile that can improve performance when creating maps from large shapefiles. More information about the different options can be obtained from the mailing list archives or online documentation, but are outside the scope of this chapter.

tile4ms

Creates an overview shapefile that points to the location of individual files and their extent rectangles. This output shapefile can be used by MapServer to reference a large number of shapefiles (or other support data formats) that are in multiple pieces. This is helpful because you don't have to append them all together or add dozens of layers in the map file. Instead, the tile index tells MapServer what layers are available and where they are geographically. It is more efficient because the extent rectangles are used as an index to find only those tiles of interest within the current map extent. It is recommended to also run shptree on the index that you create. ogrtindex and gdaltindex, part of the GDAL/OGR packages, are similar. These are highly recommended because they can index more than just shapefiles.

sortshp

Sorts a shapefile using a field in the table, rearranging the physical order of features in the file. This can be helpful when order is important for display or query optimization.

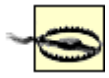
For historical reasons, the prefix shp is used for some of the programs. They were originally created to convert

10.2. Sample Uses of the Command-Line Utilities

When MapServer runs as a web mapping application it produces various graphic images. A map image is one of them. It can also produce an image of a scale bar, a reference map, and a legend. The command-line utilities let you create these manually (or scripted through your favorite scripting language). The following examples give a feel for the process of creating some basic results using these programs. These examples use the utilities from a typical MapServer installation as described in Chapter 4.

10.2.1. Acquire Some Mapping Data

A simple world map boundary dataset will be used for this example. This shapefile contains boundaries of the country borders for the world and are available from the FreeGIS (http://www.freegis.org) web site at http://ftp.intevation.de/freegis/worlddata/freegis_worlddata-0.1_simpl.tar.gz.



This version contains simplified, lower-resolution boundaries and are about 3.7 MB in size. The full resolution boundaries take more effort for MapServer to display and are a larger file at over 30 MB.

You can use the full resolution data for the exercises in this chapter, but the files are large enough to require optimization to be effective. The exercises will run more slowly than with the simplified boundaries. For further exploration of MapServer, the higher resolution boundaries make a great dataset to work with. You can download them from http://ftp.intevation.de/freegis/worlddata/freegis_worlddata-0.1.tar.gz.

The first thing to do is check out the data. This is done using the ogrinfo tool as described in Chapter 6. This isn't a MapServer utility but is a valuable tool for getting started, as shown in Example 10-1.

Example 10-1. Using ogrinfo to examine the contents of the world countries shapefile

```
> ogrinfo countries_simpl.shp -al -summary

INFO: Open of 'countries_simpl.shp'

using driver 'ESRI Shapefile' successful.


Layer name: countries_simpl

Geometry: Polygon

Feature Count: 3901

Extent: (-179.999900, -89.999900) - (179.999900, 83.627357)

Layer SRS WKT:

(unknown)

gid: Integer (11.0)

cat: Integer (11.0)

fibs: String (2.0)

name: String (255.0)
```

10.3. Setting Output Image Formats

MapServer can produce map images, legends, and scale bars in different image formats. Throughout this chapter, the PNG format is shown in the examples because it is the default image format. MapServer packages can have different default image formats. PNG format is commonly supported by many MapServer packages, but in some cases MapServer may not be able to produce PNG files or doesn't produce them by default.

MapServer can be compiled using many different options. The term "MapServer package" refers to the set of programs, e.g., shp2img, that you are using. The capabilities of these programs depend on how they are compiled. Some packages will have more features enabled than others.

The output image format can be explicitly set in the map file. This requires adding a single line specifying the output format you are requesting. Example 10-11 shows the additional line added to the earlier example map file.

Example 10-11. Specifying the output image format as JPEG

```
MAP

  SIZE 600 300

  EXTENT -180 -90 180 90

  IMAGECOLOR 180 180 250

  IMAGETYPE JPEG


  UNITS DD

...
```

This setting can be overridden when using the shp2img command with the -i option. It can't be overridden when using commands such as scalebar or legend, as mentioned earlier in "Creating Your First Map Image."

The IMAGETYPE specified must be one supported by the MapServer package you are running. To check what output formats your installation supports, run the mapserv file (or mapserv.exe on Windows) with the -v option. This option tells you what input and output formats are supported by the utilities in your MapServer package. The following example shows the output from the MapServer used in this chapter, with the output image formats highlighted:

```
> mapserv -v

MapServer version 4.4.1 OUTPUT=PNG OUTPUT=JPEG OUTPUT=WBMP SUPPORTS=PROJ

SUPPORTS=FREETYPE SUPPORTS=WMS_SERVER SUPPORTS=WMS_CLIENT SUPPORTS=WFS_SERVER

SUPPORTS=WFS_CLIENT INPUT=EPPL7 INPUT=POSTGIS INPUT=OGR INPUT=GDAL INPUT=SHAPEFILE
```

This listing shows that this installation of MapServer can only output PNG, JPEG, or WBMP format files. That means the IMAGETYPE could be set to either of those three. Therefore it can't output GIF, TIFF, PDF, or any other formats.

Chapter 11. Publishing Interactive Maps on the Web

MapServer is typically used through a web server such as Apache or IIS. Any web server that supports CGI applications can use MapServer. The web server passes along the map-related requests and returns the results to the browser. Chapter 4 describes how to compile or download binary versions of the MapServer CGI program called mapserv. This is the main MapServer program discussed in this section. The mapserv executable, not the command-line tools described earlier (Chapter 10), is used for web server applications. When this book refers to "integrating MapServer with a web server," it means using the mapserv executable (mapserv.exe on Windows) with a web server.

MapServer also has a scripting environment called MapScript. MapScript provides application programming interfaces for several programming languages. This is covered in Chapter 14. If you are a programmer and want to incorporate MapServer functionality into an application, MapScript is for you.

The CGI version of MapServer discussed in this chapter is for those who want MapServer up and running right away, without programming.

11.1. Preparing and Testing MapServer

A few things need to be set up before using MapServer. There are some differences between operating systems such as the name of your MapServer executable, where files are stored, etc.

11.1.1. MapServer for Windows (MS4W)

Chapter 4 described how to access and install the MS4W package. You might want to look at the MS4W portion of that chapter and review how to start the web server. A custom application will be built later in this chapter, but first you need to understand how MapServer works in the MS4W environment.

MS4W comes with a version of the Apache HTTP web server. The MapServer program is stored in Apache's cgi-bin folder, the main location for many web programs. If you were compiling your own mapserv.exe file, you install it by copying it into the cgi-bin folder. By default, this folder is located in the \ms4w\Apache\cgi-bin folder on the drive MS4W is installed on (e.g., *C:* or*D:*). There may be several versions of MapServer located there. For example, mapserv_36.exe is MapServer Version 3.6 (very out of date) and mapserv_44.exe is the most recent Version 4.4. Usually the more recent version will just be called mapserv.exe.

11.1.2. MapServer on Linux with Apache

Preparing MapServer on Linux is also described in Chapter 4. Installing the MapServer CGI program is similar to how you do so for MS4W. The mapserv executable is simply copied (or linked) to Apache's cgi-bin folder. For example, on SuSE Linux, the web server's cgi-bin folder is located at /srv/www/cgi-bin. The mapserv application must be copied into this folder so it is accessible to the web server. These instructions also apply to non-MS4W installations of Apache on other platforms.

11.1.3. Configuring Apache

Configuring Apache to use MapServer doesn't have to be complicated. If you put the mapserv/mapserv.exe program into the web server's default cgi-bin location, there may not be much more to do to get the program running. The next section gives a couple of example tests to see if it is running properly.

There are other settings you will need in order to access maps produced by MapServer. When MapServer runs, it needs a temporary area to create the images for maps, legends, scale bars, etc. This temporary folder must also be accessible to the web so that users can view the images created by MapServer.

11.1.3.1 Windows temporary image folder location

MS4W comes with a temporary folder already set up. This folder is c:\ms4w\tmp\ms_tmp. The Apache configuration file at c:\ms4w\Apache\conf\httpd.conf sets this folder up with an *alias* called /ms_tmp. This means that when a web application requests a file in /ms_tmp, Apache looks for the file in c:\ms4w\tmp\ms_tmp. For example, if MapServer created a map image at c:\ms4w\tmp\ms_tmp\mymap.png, the web application would request to see the image by using a URL like http://localhost/ms_tmp/mymap.png.

The key thing to remember is that MS4W creates all the temporary map image files in this folder.

11.1.3.2 Linux temporary image folder location

If you are using a fresh install of Apache, you need to set up your temporary location and alias in the configuration files manually. There are several ways to configure settings. One method that uses Apache 2 on SuSE Linux is discussed here. You can set up your MapServer temporary folder anywhere on your filesystem. To keep it simple, you can put it directly under the htdocs root folder. This means that you don't need to create an alias for Apache because the folder is a child of the web server's root folder.

For example, the root web folder may be set in the /etc/apache2/httpd.conf file and specified as DocumentRoot

11.2. Create a Custom Application for a Particular Area

MapServer has many different settings you will want to learn more about as you develop your skills. A very small, core subset of options are used in the examples in this chapter. For a complete listing of map file settings, see the map file reference document at http://mapserver.gis.umn.edu/doc/mapfile-reference.html.

Also note that there are several MapServer CGI variables used in this chapter. For more information see the current MapServer CGI reference document at http://mapserver.gis.umn.edu/doc/cgi-reference.html.

11.2.1. Changing the Initial Extent of the Map

In Chapter 10 you learned how to change the extents of the map when using the shp2img command-line utility. With an interactive map the extents can be changed by zooming in, just like the test example with the global map application. The initial map shown by that application covers the whole world. If you are interested only in a map covering a certain area, you will want to change the initial extent of the map to suit your area of interest.

You will continue to build on the previous examples in this chapter, including the global.map file. The EXTENT line in the map file specifies what the initial extent of the map will be when MapServer starts up. This was set to:

```
EXTENT -180 -90 180 90
```

which covers the whole world. Now you will set this to zoom in to a new area, to make a map of Canada. You will do so by continuing to use the countries_simpl.shp shapefile, and also use the ogr2ogr and ogrinfo tools to assess what extent you need. Please see the procedures for using ogr2ogr and ogrinfo in Chapter 10. Example 10-5 shows the same methodology for determining the extent of Bulgaria.

These command-line programs aren't part of MapServer. They are part of the GDAL/OGR package described in Chapter 3. The ogr2ogr utility extracts the shapes of Canada and saves them in to a new shapefile. ogrinfo is then run to see what the coordinate extents of Canada are.

The OGR utilities are available as part of the FWTools package. You can download it from http://fwtools.maptools.org.

The new shapefile is used only as a temporary means of determining the coordinates of the area of interest. These coordinates are then set in global.map so that MapServer focuses on Canada.

The extent of Canada, as returned from ogrinfo, is:

```
Extent: (-141.000000, 41.675980) - (-52.636291, 83.110458)
```

You can put this extent into the global.map file and see how things look. Simply copy/paste this text right into the map file and then clean it up, removing brackets and commas. Example 11-4 shows how the first few lines of the changed application look.

Example 11-4. A modified map file to include the extent of Canada instead of the whole world

```
MAP

  SIZE 600 300

  EXTENT -141 42 -52 83
```

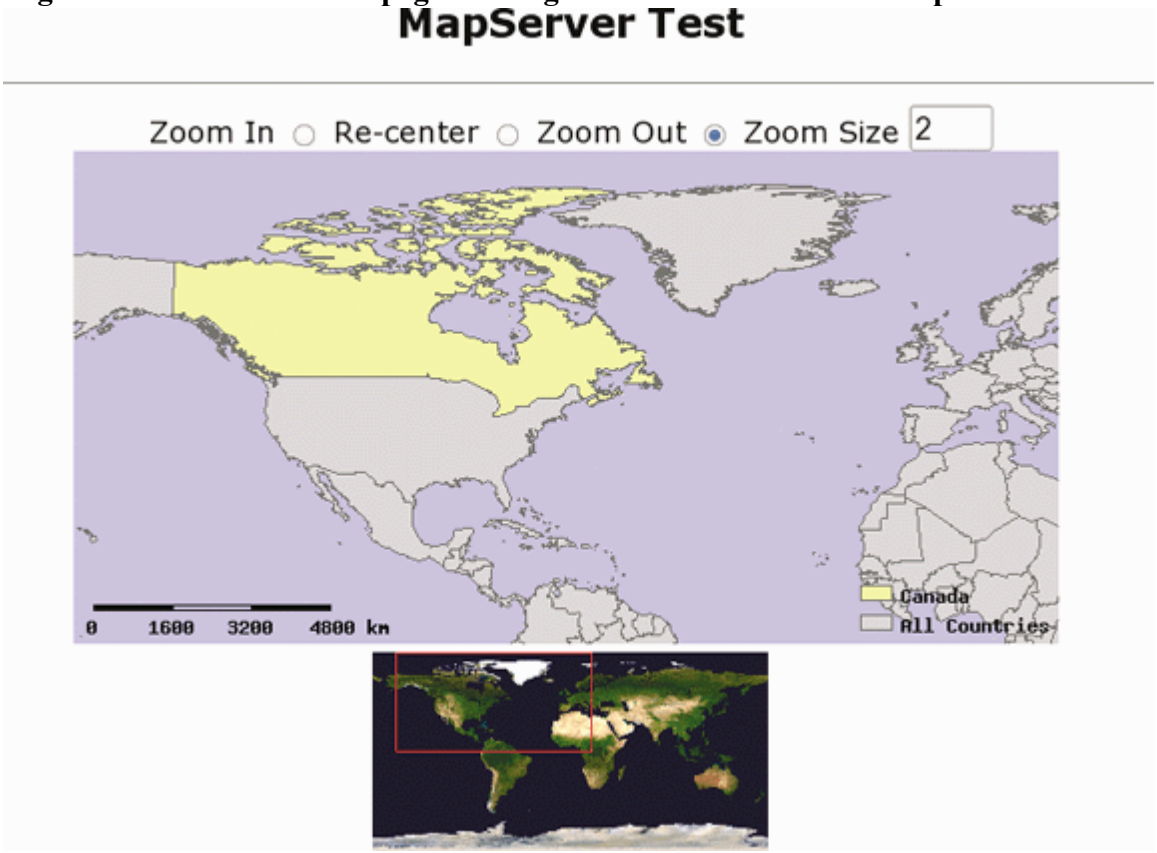
11.3. Continuing Education

The examples in this chapter barely scratch the surface of MapServer capabilities. You can get an idea of the other kinds of functions that are available by looking at other MapServer applications. The MapServer web site maintains an application gallery at http://mapserver.gis.umn.edu/gallery.html.

It shows several examples of applications people have built, including sites that allow querying, data entry, searching, etc.

There are also quite a few developments under way that will make MapServer applications easier to develop. These are often called application development frameworks. They require some more advanced setup to get started, because of their

Figure 11-18. The final web page showing an interactive reference map



dependency on web programming languages (e.g., PHP). Here are a few application development tools:

- Chameleon and MapLab (http://www.maptools.org)
- Flexible Internet Spatial Template (FIST) (http://datashare.gis.unbc.ca)
- MapBender (http://www.mapbender.org)

There has also been increasing interest in integrating MapServer functionality with a content management system (CMS), groupware, or wiki. Tikiwiki (http://tikiwiki.org/tiki-map.phtml) is one example of this.

Chapter 12. Accessing Maps Through Web Services

While publishing maps to the traditional Web for human consumption is definitely one of MapServer's most attractive features, you may also want to share your data so that other programs can use it. To do this, you'll likely want to take advantage of web services. Ethan Cerami gives a good definition of a web service in the introduction to his book Web Services Essentials (O'Reilly):

A web service is any service that is available over the Internet, uses a standardized XML messaging system, and isn't tied to any one operating system or programming language.

Similarly, Venu Vasudevan in his A Web Services Primer, defines a web service as "XML plus HTTP." See http://webservices.xml.com/pub/a/ws/2001/04/04/webservices/index.html.

The term web service typically refers to an Internet communication protocol between applications. Web services are built on the standard web HTTP protocol used for web browsing. HTTP is the transfer protocol coupled with XML as a communication language. These communication specifications allow the Internet to act as a communication network between applications—not just as a service for sharing web pages.

Web services are important because they play the integral part of interoperability between applications which were developed independently of each other and yet are able to communicate. The web services standards are independent of programming languages, operating systems, and platforms. They provide a standardized method of communicating between web-accessible applications, and are especially important to mapping applications that use the Internet to share data.

One of the great things about web services is that you don't need to understand all their ins and outs to use them. To use mapping-related web services, you just need to know how to point your application at a data source across the Internet. If you want to share your data with others, you need to know how to set up your applications so they are accessible.

12.1. Web Services for Mapping

While the internal details of web services—XML-RPC, UDDI, WSDL, SOAP (all covered in Web Services Essentials, from O'Reilly)--are interesting to developers, they aren't the focus here. There are services particular to mapping and, even though they may operate in a similar fashion, they are completely different from the general services a web developer may be familiar with. This chapter focuses on the web services MapServer supports.

12.2. What Do Web Services for Mapping Do?

Mapping web services has been developed independently of the Web. They use similar approaches and follow some common communication protocols, but the web services for mapping are maintained independent of web services in general.

Web services for mapping are the future of web mapping. Computing and network resources have reached the level of sophistication to make this possible. As data sources continue to expand, the demand to have access to them increases. Meeting this demand will only be possible using mapping web services.

Web services for mapping essentially fill two roles: accessing remote data sources as a consumer and serving up or sharing data as a provider for others. Web services for mapping are all about sharing information.

Accessing remote data sources requires some specific knowledge about the data source. Particulars may include the server address to connect to, the data layers available, or the format the data can be sent to you in. Once you know the details, you can set up your application to request exactly what you want. MapServer will add in its own pieces of information to each request so you get something logical back on a map.

Sharing (a.k.a. publishing) your own data is relatively easy to do. A few additions to your application will allow others to query it for such things as available layers and data format options. Once you've made a few modifications to your map file, other users can add your layers to their applications.

Web services for mapping don't require you to know a bunch of details about how requests are made or sent. MapServer will take care of the guts of the operation for you, such as sending HTTP or XML-based requests and receiving responses and data. You usually won't have to read XML when using these services. However, you might occasionally need to review an XML-based document to learn more about a service.

The examples in this chapter walk you through the steps of building your own request manually. You can then incorporate this technology into your own applications. If you are just planning to have MapServer broker requests to a WMS server, then you won't need to pay as much attention to the manual examples.

Just as telephones or CD players use common methods for sending or receiving information, so do mapping web services. Therefore, other products or applications can work together as long as they use the common standards or protocols.

12.2.1. Why Use These Services?

There are various reasons for sharing and accessing map data using web services. A government or large corporation may make map data available so that other divisions or departments don't need to store their data locally. Consider the cost savings often touted for centralizing data storage. If data is centralized, the users still need to access it. Web services for mapping make the data available.

Perhaps someone is willing to share data with you but can't allow you to have a complete copy, for whatever reason. A web service can facilitate that. Some map data, such as hurricane locations, is constantly changing. Downloading that data might not be possible or timely. Integrating a layer in your map that accesses a web service in realtime might be the answer. Some data repositories are so large that it would be unrealistic to copy them to your location. Web services can make that data available for you at the time you need it. This helps reduce the need for continual data storage upgrades. Figure 12-1 illustrates the basic function of a MapServer application accessing remote data through a web services request.

Whether sharing or accessing, it is all about making the most of our time and resources. Continually changing an application to point to new data can be inefficient. Storing large amounts of data locally may also be an unreasonable

12.3. Using MapServer with Web Services

It isn't necessary to program applications to use web services. MapServer is already able to access and provide data using a web services framework. This chapter focuses on setting up your MapServer application to use these services and will show you how to define a layer in your map file (which uses data retrieved from a remote data source). You will also learn how to set up your own service to make your data available to others.

The web services specifications for mapping come from the Open Geospatial Consortium (formerly known as the Open GIS Consortium). The OGC has developed standards or specifications for web services for mapping. The goal of the organization is to improve interoperability between applications by creating some common interchange languages through common standards. The membership list is quite impressive, as are the goals. Here is a brief description from the About page at http://opengeospatial.org/about/:

The Open Geospatial Consortium, Inc. (OGC) is an international industry consortium of 270 companies, government agencies and universities participating in a consensus process to develop publicly available interface specifications. OpenGIS® Specifications support interoperable solutions that "geo-enable" the Web, wireless and location-based services, and mainstream IT. The specifications empower technology developers to make complex spatial information and services accessible and useful with all kinds of applications.

The focus here isn't on creating applications, but on specifications for sharing data. This makes sense when you consider that major mapping, GIS, and IT players sit together on this consortium (e.g., Autodesk, ESRI, MapInfo, Intergraph, and Oracle).

MapServer can implement many OGC specifications, in varying degrees of completeness, as summarized in Table 12-1.

Table 12-1. OGC specifications implemented by MapServer

Abbreviation	Name	Purpose
WMS	Web Map Service	Share and request vector and raster map data in plain image format
WFS	Web Feature Service	Share and request vector map data and attributes in GML format
WCS	Web Coverage Service	Share image/raster data with original data values
WMC	Web Map Context	Save and load views of a WMS application as XML
SLD	Styled Layer Descriptors	Request particular symbolization and styling from a WMS
GML	Geography Markup Language	XML-based format for spatial and tabular data interchange
Filter	Filter Encoding	XML-based format for defining queries on spatial data

Two of these specifications are supported by Web Map Service and Web Feature Service. Other

12.4. Reference Map Files

This section includes the complete listing of map files (Examples 12-13 through 12-16) used in this chapter. Additions to the files are highlighted. Lines to be removed are commented out by the preceding # symbols.

Example 12-13. The global_mosaic WMS layer added to the global.map file

```
MAP
  SIZE 600 300
  EXTENT -180 -90 180 90
  IMAGECOLOR 180 180 250
 IMAGETYPE PNG24


  UNITS DD


  WEB
    TEMPLATE global.html
    IMAGEPATH "/srv/www/htdocs/tmp/"
    IMAGEURL "/tmp/"
  END


  LAYER
    NAME global_landsat
    TYPE RASTER
    STATUS DEFAULT
    CONNECTIONTYPE WMS
    CONNECTION "http://wms.jpl.nasa.gov/wms.cgi?"
    MINSCALE 20000
    METADATA
      "wms_server_version" "1.1.1"
      "wms_srs" "EPSG:4326"
      "wms_format" "image/jpeg"
      "wms_styles" ""
      "wms_name" "modis,global_mosaic"
    END
  END
```

Chapter 13. Managing a Spatial Database

The previous chapters explained how to interrogate and convert data, and how to select subsets of it for more customized use. This chapter is for those who need a spatial database for creating, managing, and manipulating data. Spatial databases combine the power of a relational database with geographic and tabular information. The focus of this chapter is on getting some spatial data into a database, querying it, and using it in a map. It will also cover how to use PostGIS data in a few different applications and how to export it into other formats.

13.1. Introducing PostGIS

PostGIS is a spatial (or geographic) extension of the popular open source PostgreSQL relational database. This chapter focuses on using the normal tabular components of PostgreSQL in conjunction with the spatial data management functions of PostGIS.

For more in-depth help with PostgreSQL, I recommend O'Reilly's Practical PostgreSQL (http://www.oreilly.com/catalog/ppostgresql/) and the PostgreSQL web site http://www.postgresql.org.

PostGIS is an open source extension developed by Refractions Research. Their web site (http://postgis.refractions.net) gives this summary of the product:

PostGIS adds support for geographic objects to the PostgreSQL object-relational database. In effect, PostGIS "spatially enables" the PostgreSQL server, allowing it to be used as a backend spatial database for geographic information systems (GIS), much like ESRI's SDE or Oracle's Spatial extension.

13.2. What Is a Spatial Database?

A database is a tool for storing and accessing tables of information. Traditional databases store information in fields and records (columns and rows or attributes and values). The types of data that fields can hold varies across different types of databases but, generally speaking, they hold numeric and text data. The main feature of a database is that of querying, where you can retrieve information that meets your specific criteria. Relational databases allow you to join information from multiple tables using a common piece of information that is in both tables.

To learn more about relational database management systems (RDBMS), see http://en.wikipedia.org/wiki/RDBMS.

A spatial database is much the same, but it can also store geographic data. Several databases and GIS products use the term spatial database to mean slightly different things. For example, ESRI's Spatial Database Engine (SDE) isn't a spatial database, but is advertised as an interface between client software and a normal database. It allows spatial data to be stored in SDE's format, in the database. To load and manipulate the spatial data, you need to have an ESRI product or access to an ESRI service.

ESRI's SDE product isn't the only option for spatially enabling a database. Despite the limited amount of marketing, there are other databases available that better fit the description of spatial database. Oracle has an extension called Oracle Spatial and IBM has a Spatial Extender for the DB2 database. MySQL database also has a spatial extension. All these are similar to PostGIS; they store and access the spatial data using database tools, without requiring specialized GIS software to access or modify the data.

PostGIS has several features most commercial spatial databases don't, some of which initially draw people to PostGIS as an enterprise spatial database management system. PostGIS is actively developed and supported. Support for PostGIS is built into an increasing number of applications including MapServer, JUMP, QGIS, FME, and more. There are extensive functions for interacting with spatial data, without needing a GIS client application. This has inspired many to use PostGIS.

PostGIS can be considered an advanced spatial database because it has the ability to both store and manipulate spatial data. PostGIS isn't simply a data storage repository, but also an environment for interacting with spatial data. The OGC has created a specification for storing and querying spatial data in an SQL database—the Simple Features Specification for SQL (SFSQL). OGC specifications aren't just nice to have; they are becoming an integral requirement for geospatial data interoperability. When sharing data or enabling open access to geospatial information is required, these open standards become critical.

PostGIS has one of the most robust implementations of the SFSQL specification, according to the OGC resources page at http://www.opengeospatial.org/resources/?page=products. Because PostGIS implements all SFSQL specifications, you can access standardized functions without proprietary twists and turns over the lifespan of your projects. Some other applications only implement subsets of the SFSQL specification.

The way geographic data is presented in PostGIS (as returned in a query) is very intuitive, as you will see from the examples in this chapter. You can get access to coordinates of spatial features through a variety of methods, depending on your need. You can query from a command-line SQL tool or view them graphically with mapping software. Either way, you aren't bound to using proprietary tools to get access to your information.

13.2.1. Server-Based GIS

PostgreSQL is a database server product. When requests are made to the database, the server processes the request, prepares the data, and returns the results to your application. All the heavy-duty work is done on the server, and only the results are sent back to you. For many applications, this is critical. Even with the horsepower of modern computers, most PCs aren't designed to handle the intense workload of database queries. If all the data had to be sent across a network to be processed by your application on the user's machine, the network and client program

13.3. Downloading PostGIS Install Packages and Binaries

Install packages or binary (precompiled) versions are available for a few different computing platforms.

PostgreSQL and PostGIS are both in active development, and some links and references for getting them installed can become quickly out of date. You should always refer to the main web sites for the latest pointers and instructions:

13.3.1. PostGIS for Windows

Native Windows versions of PostgreSQL have recently become available. These allow you to install the standard PostgreSQL database and, depending on the package, may also include PostGIS functionality. For packages that don't include all the PostGIS functionality, the PostGIS libraries and scripts need to be installed and run. Some Windows installation packages have been created to help automate this.

The official install package for PostgreSQL is available at ftp://ftp.postgresql.org/pub/binary/. Under the most recent version, there is a Windows folder.

One version was developed by Mark Cave-Ayland and is available from his web site, http://www.webbased.co.uk/mca/. He includes links to the main PostgreSQL installers as well as his own package containing PostGIS and other spatial libraries such as GEOS and Proj. This package is used by many people who frequent the PostGIS mailing list (which is the place to ask questions when you run into difficulty). See this link to search the archives of the mailing list or to subscribe to it: http://postgis.refractions.net/support.php.

PostGIS is also available as a MapServer for Windows package. See the MS4W download page for notes and the package to download: http://www.maptools.org/ms4w/. MS4W is easy to install and start using.

Another PostGIS installer is available as part of the DC Maintenance Management System (DCMMS) at http://dcmms.sourceforge.net/.

13.3.2. PostGIS for Linux

Different Linux distributions tend to have different easy-to-install versions of PostgreSQL and PostGIS. RPM and Debian packages are probably two of the most common.

French readers may be interested in this page, which describes some installation sources in French: http://www.01map.com/download/.

13.3.2.1 RPM packages

If you are looking for simplicity (and your operating system supports them), RPM packages are probably your best bet. Packages are available from a couple of sources. As with the Windows sources mentioned previously, you will get the PostgreSQL database package and then either compile your own PostGIS portion or install a PostGIS package. If you just want a PostgreSQL database, try going directly to a PostgreSQL download mirror and searching under the binary and rpms folders for your platform. RedHat 9 and Fedora Core 1 and 2 are both available at ftp://ftp.postgresql.org/pub/binary/v8.0.1.

If you intend to compile your own PostGIS support using these packages, you will need to install the source RPM packages as well. PostGIS compilation needs access to files that come only with the source PostgreSQL packages.

Refractions Research hosts a comprehensive set of RPM packages including a PostGIS package. They also have packages for Fedora, Mandrake, RedHat, SuSE, and source packages as well. You can find these packages at http://postgis.refractions.net/rpms/.

13.4. Compiling from Source Code

Compiling PostGIS from source code isn't covered here. Source code can be downloaded from the PostGIS web site at http://postgis.refractions.net/download.php. This site includes hourly snapshots of source code from the CVS repository, as well as official releases. Source RPMs are also available.

Compilation instructions are provided as a README file that comes with the source:
http://postgis.refractions.net/docs/ch02.html http://postgis.refractions.net/README.postgis.txt

13.5. Steps for Setting Up PostGIS

The PostGIS documentation is the best source for walking through all the steps of setting up PostGIS. This section provides a very brief run-through of steps. Depending on your skill and understanding of PostgreSQL and SQL scripts, these instructions may or may not be enough. If not, please refer to the main PostGIS documentation at http://postgis.refractions.net/docs.

Depending on the method you have used to get the PostgreSQL database set up and PostGIS installed, there will be different steps required to get started. Some installation packages include all the steps required to jump right into using PostGIS. Others don't, which means you still need to set up the final stages of PostGIS functionality. If you compile PostGIS from source code, you will always have the following steps to walk through. In a nutshell, these steps involve:

1. Getting PostgreSQL up and running
2. Enabling pl/pgsql language support
3. Loading the postgis.sql (or lwpostgis.sql) script
4. Loading the spatial_ref_sys.sql script
5. Creating a database

Other steps involving loading and accessing data are covered in later sections.

13.5.1. Getting PostgreSQL Up and Running

You must have a PostgreSQL database service running before setting up PostGIS. PostgreSQL runs in a similar fashion to other enterprise database services (MySQL, Oracle, DB2, SQL Server). It waits for requests to come from a client program and then responds by interacting with databases on the server that the service is running on.

To test that your database service is running, you can use the command-line tool psql with the list parameter to give you a list of available databases:

```
> psql -l
           List of databases
      Name      |  Owner   | Encoding
----------------+----------+-----------
 template0      | postgres | SQL_ASCII
 template1      | postgres | SQL_ASCII
(2 rows)
```

This shows two available databases that are part of every PostgreSQL system. The template1 database is the default used as a template for creating a new database. It is copied, and all settings in it are made available to any new database you create. To enable PostGIS functionality for a database, you load PostGIS settings and functions into that database. If you load them into template1, all the PostGIS functionality will be available in every subsequent

13.6. Creating a Spatial Database

The template1 database is the default system template used when creating new databases. Don't use it as an operational database! Instead, create a new database for your projects. New databases are created using the command-line program createdb, followed by the name that identifies the new database:

```
> createdb project1

CREATE DATABASE
```

In this case, the new database is called project1. The text that is printed out (CREATE DATABASE) confirms that the command ran and finished. Now when existing databases are listed, project1 should be included:

```
> psql -l

       List of databases

    Name    |   Owner   | Encoding

------------+----------+-----------

 project1   | tyler     | SQL_ASCII

 template0  | postgres | SQL_ASCII

 template1  | postgres | SQL_ASCII

(3 rows)
```

The project1 database is now ready for data loading. To quit psql, use the \q command.

13.7. Load Data into the Database

There are many ways to put data into a database. One method is to manually type SQL commands with psql to insert data into a table. You can also use programs that convert data into an SQL script that can then be loaded, or you can use a tool that exports data directly to the database.

13.7.1. Using shp2pgsql

PostGIS comes with command-line tools called pgsql2shp and shp2pgsql. These can be used to convert from a PostGIS table to a shapefile and back.

The tool shp2pgsql converts the shapes to a text stream of SQL commands. Therefore, you need to pipe it to a text file for later loading or to the psql command to use it immediately. For example, use one of these methods:

```
> shp2pgsql countyp020.shp countyp020 >  mycounties.sql

> psql -d project1 -f mycounties.sql
```

or use this one to load the data immediately to the database:

```
> shp2pgsql mycounties.shp mycounties | psql -d project1
```

By default, the shp2pgsql command puts the geometry data into a field called the_geom, whereas the default for the ogr2ogr command (shown next) puts the geometry data into a field called wkb_geometry. These defaults can be overridden and changed to something more meaningful. To acquire the countyp020 data, see the next section.

13.7.2. Using ogr2ogr

ogr2ogr is an excellent program for putting spatial data into a database. It's part of the GDAL/OGR toolkit included in FWTools and introduced in Chapters 3 and 7. This command-line tool takes any OGR supported data layer and exports it into the database.

This example uses some data taken from the U.S. National Atlas site at http://nationalatlas.gov/atlasftp.html. Specifically, it uses the County Boundaries data at http://edcftp.cr.usgs.gov/pub/data/nationalatlas/countyp020.tar.gz.

The file is a gzip'd tar file. On Linux, you can expand this file using the command-line program tar:

```
> tar -xzvf countyp020.tar.gz
```

This creates a shapefile named countyp020. On Windows, most Zip programs (e.g., WinZip) can decompress this file for you.

The ogrinfo command then provides a summary of how many features are in the file:

```
> ogrinfo countyp020.shp -al -so

...

Feature Count: 6138

...
```

It is a good idea to have a feel for how much data will be converted before running the next step. The more features there are, the more time it will take to load the data to the database.

13.8. Spatial Data Queries

One of the strengths of PostGIS is its ability to store spatial data in a mature, enterprise-level, relational database. Data is made available through standard SQL queries, making it readily available to do analysis with another program without having to import and export the data into proprietary data formats. Another key strength is its ability to use spatially aware PostGIS functions to perform the analysis within the database, still using SQL. The spatial data query examples in this section are the most basic possible. PostGIS boasts of a variety of operators and functions for managing, manipulating, and creating spatial data, including the ability to compute geometric buffers, unions, intersections, reprojecting between coordinate systems, and more. The basic examples shown here focus on loading and accessing the spatial data.

To see what data is in the countyp020 table, you can use any basic SQL query to request a listing. For example, this typical SQL query lists all data in the table, including the very lengthy geometry definitions. Running this query isn't recommended:

```
# SELECT * FROM countyp020;
```

Getting a list of over 6,000 counties and their geometries would hardly be useful. A more useful query might be something like Example 13-3 which limits and groups the results.

Example 13-3. A basic SELECT DISTINCT query to list the counties in New Mexico state

```
# SELECT DISTINCT county FROM countyp020 WHERE state = 'NM';
             county
---------------------------------
 Bernalillo County
 Catron County
 Chaves County
 Cibola County
 Colfax County
 Curry County
 DeBaca County
 Dona Ana County
 Eddy County
 Grant County
 Guadalupe County
 Harding County
 Hidalgo County
 Lea County
 Lincoln County
...
 (33 rows)
```

13.9. Accessing Spatial Data from PostGIS in Other Applications

Visualizing results isn't always the end goal of a spatial query. Often, someone just needs to write a report or provide a statistic. However, figures often help explain the data in a way that a report can't, which is why mapping is such a useful tool for communication.

If you want to visualize the results of PostGIS queries, you have a few options. Depending on what software you use, the simplest method may be to export the data from PostGIS to another format, such as a GML file or an ESRI shapefile. Or you may want to view the data directly in a desktop viewer such as OpenEV or put it in a MapServer web map application.

13.9.1. Exporting PostGIS Data into a Shapefile or GML

You can use the ogr2ogr utility to convert from PostGIS into many other formats. As discussed in Chapter 7, it is run from the command line. Here is an example of how to convert the mycounties view from PostGIS into a shapefile, though any OGR-supported output format can be used:

```
> ogr2ogr -f "ESRI Shapefile" mycounties.shp "PG:dbname=project1" mycounties
```

The -f parameter specifies what the output data format will be. The output dataset name will be mycounties.shp. The source dataset is the project1 database. The final word, mycounties, is the name of the layer to request from PostGIS. In this case it is a PostgreSQL view, but it can also be a table.

The shapefile can then be loaded into any GIS or mapping product that supports shapefiles. This format is fairly universal. To create a GML file, it looks almost the same:

```
> ogr2ogr -f "GML" mycounties.gml "PG:dbname=project1" mycounties
```

As noted earlier in the "Load Data into the Database" section, the shp2pgsql and pgsql2shp command-line tools may also be used. Both shapefiles and GML can be used as a data source for a layer in MapServer as discussed in Chapters 10 and 11.

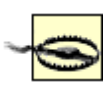
13.9.2. Viewing PostGIS Data in OpenEV

The real power of PostGIS is that the data it holds can be accessed directly by a number of applications. When you export data from PostGIS into another format, it is possible for your data to become out of sync. If you can access the data directly from PostGIS, you won't need to do the exporting step. This has the added benefit of always being able to access the most current data. OpenEV and MapServer are two examples of programs that access PostGIS data directly.

OpenEV can load spatial data from a PostGIS database. This is only possible if you launch OpenEV from the command line and specify the database connection you want to access. For example, to access the project1 database, the command line to start OpenEV looks like this:

```
> openev "PG:dbname=project1 user=tyler host=localhost"
```

You will see a long list of warning and status messages as OpenEV starts up. If it connects to the database properly, you will see a list of the tables to choose from. Figure 13-5 shows the layer list from my project1 database.



The vertical scrollbar seems to have some problems. You may need to resize the window by stretching it taller, to be able to scroll down to find your layers. More recent layers/tables are listed at the bottom.

Chapter 14. Custom Programming with MapServer's MapScript

If you have programming experience, you may want to do more to customize MapServer, incorporating MapServer capabilities directly into a custom application. Enter MapScript, an application program interface for MapServer.

14.1. Introducing MapScript

MapScript exposes the functionality of MapServer to various scripting languages. This reduces programming time for developers who want to add mapping capabilities to an application. Rather than create a custom method for mapping, the MapScript API provides some powerful tools that are robust and ready to use. It also provides a convenient way to interact with mapping data while still using your favorite programming language.

MapScript allows you to load, manipulate, and create map files. For example, you can change layer settings, manipulate map file classes, produce output images, export spatial data, and much more. Because it uses common scripting environments, MapScript functions can be combined with other functions of that language.

Several languages have MapScript support; PHP, Python, Perl, Java and Ruby are readily available. C# and TCL support are under development as well, and these are also included in the MapServer source code tree.

14.2. Getting MapScript

How do you get MapScript? There are variety of methods, some easier than others. The common requirement for running MapScript is that the scripting language itself must be available. The header files for the language may also be required if building the MapScript modules from source code.

14.2.1. Building MapScript from Source Code

Building MapScript from source for each language isn't detailed here, but there is a common approach.

To build MapScript for a language, support for that language must have been configured and built into MapServer itself when it was compiled. This sets up the various files in the MapScript language folders that are needed for the next step.

The MapServer source code directory structure includes a folder called mapscript. This folder has subsequent folders for each language that has some form of MapScript support. The folder may have a Makefile that is ready to be used by the make command, or it may have some setup scripts. Running these produces MapScript modules or libraries (depending on the language), and the scripts can often be used to help install them too. For example, with Python, you build and then install MapScript by running:

```
> python setup.py build

> python setup.py install
```

There are several other files and directories that are part of the Simplified Wrapper and Interface Generator (SWIG) environment. SWIG is used to port MapScript to certain languages. This is now the standard method for producing a MapScript extension for additional languages. If you are familiar with SWIG and want to write a *mapscript_wrap.c* file for a new language, your contributions would be more than welcome. For more information on SWIG, see http://www.swig.org.

Compilation instructions are included in README files in the specific MapScript language folder. More information can be found in the MapScript and Reference Guides sections of the MapServer documentation page at http://mapserver.gis.umn.edu/doc.html.

14.2.2. Obtaining Binary Versions of MapScript

Binary versions of MapScript are also available, depending on the programming language. The best place to get some personal direction is from the MapServer mailing list. These sites distribute various forms of MapScript:

- DM Solutions has PHP MapScript binaries available at http://maptools.org/php_mapscript/index.phtml.
- It is also available as part of the MapServer for Windows (MS4W) environment at http://maptools.org/ms4w/index.phtml.
- Frank Warmerdam's FWTools package includes Python MapScript. Windows and Linux platforms are supported. See http://fwtools.maptools.org.
- The FGS project at http://www.maptools.org/fgs/ provides an installer for MapServer and PHP MapScript under Linux.
-

14.3. MapScript Objects

For a complete MapScript object, variable, and method guide see the latest on the MapServer documentation web site page or Sean Gillies' reference guide (http://zcologia.com/mapserver).

Class diagrams are also available from these web sites:

Classes within the map file can be accessed as objects in MapScript. In fact, there are more objects to handle than you normally see, because MapServer does a lot of the work with them behind the scenes. The main part of a MapScript application is the map object (mapObj). The mapObj can have many layer objects (layerObj) as well as legend (legendObj), scale bar (scalebarObj), and reference map (referenceMapObj) objects. Figure 14-1 shows a diagram of how these objects are hierarchically structured.

The main object is the mapObj. You can add a layerObj, classObj, and styleObj to the mapObj. Other objects should never be explicitly created, such as webObj, scalebarObj, legendObj, and colorObj. These are already part of the mapObj, and constructing new objects for them will fail. If you get segmentation faults from MapScript, this may be the cause.

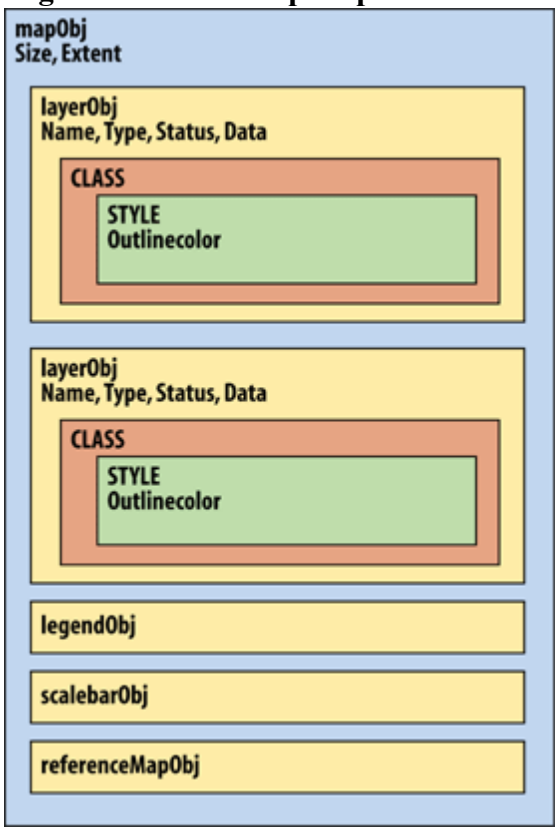
All the attributes of objects that you normally see in the map file are available for manipulation through MapScript.

Using the MapScript API is best understood by walking through some simple examples.

14.4. MapScript Examples

MapServer map files are still a key component to writing a MapScript-based application. If you have a map file ready to use, you can start accessing and manipulating it right away. However, a powerful feature of MapScript is the ability to create maps on the fly without having a map file. In this case, you can build your map from scratch.

Figure 14-1. The MapScript API's hierarchical object structure



In the first two examples, a simple map file is used, as shown in Example 14-1. In the third example, a custom map file is created using MapScript. To keep the example short and simple, I've removed optional components like the scale bar, legend, and reference map.

Example 14-1. A simple map file used for the examples in this chapter

```
MAP

  SIZE 600 300

  EXTENT -180 -90 180 90

  IMAGECOLOR 180 180 250

  IMAGETYPE PNG


  UNITS DD

  WEB

    IMAGEPATH "/srv/www/htdocs/tmp/"

    IMAGEURL "/tmp/"

  END
```

14.5. Other Resources

For more examples of MapScript, see some of these links. They point to a few different packages and tutorials.

PHP MapScript By Example HOWTO

http://mapserver.gis.umn.edu/doc/phpmapscript-byexample-howto.html

Chameleon, MapLab and Gmap demo

Comprehensive mapping systems built with PHP MapScript (http://maptools.org/)

Mobile Geographics:

Simple recipes using PHP MapScript (http://www.mobilegeographics.com/mapserver/)

Tom Kralidis

Example of downloading point coordinates and creating a new shapefile (http://www.kralidis.ca/gis/eqmapping/index.html)

Python Cartographic Libary (PCL)

Mapping tools for Python, not necessarily using MapServer, but similar in features; also includes links to Cartographic Objects for Zope. (http://zcologia.org/)

14.6. Parallel MapScript Translations

A simple example is shown in this section using multiple scripting programming languages. Examples 14-5 through 14-9 provide a reference point for programmers who want to see how MapScript can be used in various languages. All these examples do the exact same thing—open a map file, draw the map, and save it to an image file. They are meant to be run in the same folder as the *global.map* map file, and they all produce an image file called *worldmap.png*.

Example 14-5. Python MapScript basic example

```
# map1.py

# Python MapScript Example 1


import mapscript


# Set the map file to use

mapfile = "global.map"


# Create a mapObj, initialized with the mapfile above

mapobject = mapscript.mapObj(mapfile)


# Create an imgObj that has an image of the map

mapimage = mapobject.draw(  )


# Save the mapimage to a file

mapimage.save("worldmap.png")
```

Example 14-6. Perl MapScript basic example

```
# map1.pl

# Perl MapScript Example 1


use mapscript;


# Set the map file to use

$mapfile = "global.map";
```

Appendix A. A Brief Introduction to Map Projections

Map projections are a critical component of any mapping application, whether for a hardcopy printout or an interactive web map. If you are new to mapmaking, you may see projections as a confusing and intimidating topic. If so, keep in mind that even some of the more advanced mapmakers have just enough map projection knowledge to do what they need. The theory runs as deep as math itself. With a minimal amount of orientation, you can get started with map projections.

This guide will prepare you for using map projections with MapServer. For more information about MapServer, see the other chapters in this book. In this section, general concepts are discussed, and a few popular projections are introduced. The syntax for setting projections in a MapServer map file is also covered.

A.1. The Third Spheroid from the Sun

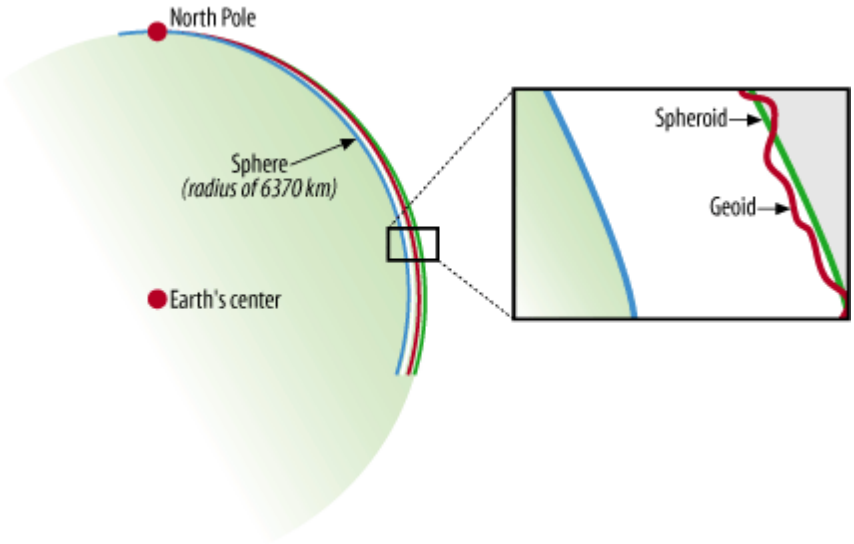
The Earth is round, or so they say. Video games, globes, and graphic art may depict the Earth as a perfect ball shape or sphere, but in reality the Earth is a bit squished. Therefore, we call the Earth a spheroid, rather than a sphere. It is sphere-like, but somewhat elliptical.

To take it one level further, we all know that the surface of the Earth isn't perfectly uniform. There are mountains and valleys, bumps and dips. Geoid is the term used for a more detailed model of the Earth's shape. At any point on the globe, the geoid may be higher or lower than the spheroid. Figure A-1 shows an example of the relationships among the sphere, spheroid, and geoid.

Figure A-1 is based on a graphic courtesy of Dylan Prentiss, Department of Geography, University of California, Santa Barbara. Further descriptions are available at the Museum's Teaching Planet Earth web site, http://earth.rice.edu/mtpe/geo/geosphere/topics/remotesensing/60_geoid.html.

As you can see, when talking about the shape of the Earth it is very important to know what shape you are referring to. The shape of the Earth is a critical factor when

Figure A-1. Illustration of methods for describing the Earth's shape



producing maps because you (usually) want to refer to the most exact position possible. Many maps are intended for some sort of navigational purpose, therefore mapmakers need a consistent way of helping viewers find a location.

A.1.1. Geographic Coordinate System

How do you refer someone to a particular location on the Earth? You might say a city name, or give a reference to a landmark such as a mountain. This subjective way of referring to a location is helpful only if you already have an idea of where nearby locations are. Driving directions are a good example of a subjective description for navigating to a particular location. You may get directions like "Take the highway north, turn right onto Johnson Road and go for about 20 miles to get to the farm." Depending on where you start from, this may help you get to the farm, or it may not. There has to be a better way to tell someone where to go and how to get there. There is a better way; it is called the Geographic Coordinate System.

The increasing use of hand-held Global Positioning Systemreceivers is helping the general public think about the Geographic Coordinate System. People who own a GPS receiver can get navigation directions to a particular location using a simple pair of numbers called coordinates. Sometimes an elevation can be provided too, giving a precise 3D description of a location.

A Geographic Coordinate System, like that shown in Figure A-2, is based on a method of describing locations using

A.2. Using Map Projections with MapServer

The projection settings in a MapServer map file are at two levels: the output map projection and the projection of the input layers of features to be drawn. Example A-1 shows a sample of the projection settings for an output map.

Example A-1. Example settings for the map to be rendered using an Albers Equal Area projection

```
PROJECTION

  "proj=aea"

  "ellps=WGS84"

  "lat_0=10"

  "lon_0=-90"

END
```

The settings may look confusing to anyone unfamiliar with projections, but they have a straightforward purpose. If you can get these four settings clear in your mind, you'll be able to handle most projections with ease.

PROJ.4: Projection Library and Utilities

The terms used in Example A-1, proj, ellps, lat_0, lon_0, are all keywords for the projection library used behind MapServer, called PROJ.4. There are other options (e.g., datum, units, etc.), but these are the most common. PROJ.4 isn't just a set of libraries that MapServer uses. It also comes with some command-line utilities. One of these is a program called proj, which lists the available projections, ellipsoids, etc.

Running the command:

```
> proj -l
```

returns a list of the available map projections, including the abbreviation used in the proj= setting for MapServer.

This utility can also project coordinates interactively by keyboard or from a text file; it's perfect for bulk projection of coordinates.

For more information on PROJ.4 see http://proj.maptools.org/.

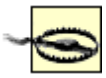
The first line in Example A-1, "proj=aea", is the most important. It specifies the name of the projection. The abbreviation aea is short for Albers Equal Area projection. Every PROJECTION...END object must have a projection name specified. You should also specify an ellipsoid, e.g., ellps=WGS84. Some functions will not require an ellps parameter to be used, but some will fail without it. If in doubt, use the World Geodetic System WGS84. All the other parameters can be optional. There are 121 projections available through PROJ.4.

The second line, "ellps=WGS84", specifies the ellipsoid to use for the projection. WGS 84 is a specific

A.3. Map Projection Examples

The examples in the following section are based on some global map data that is stored in the Geographic Coordinate System. Each example is one projection at a suitable scale. This section is intended to introduce a few of the more common types of map projections and provide a picture of what they look like. Some brief description is provided, but other books and documents are recommended for learning more about these projections.

The specific MapServer projection settings are included for reference. As long as all your layers have a valid projection set for them, these example projection settings can be used to change the output map projection.



A global map grid shapefile was used as a layer for the map figures in this section. This shapefile was used instead of the internal MapServer graticule generator (the GRID object). The GRID object produces very simple lines that can't be properly transformed by many projections.

A Python script called mkgraticule.py is included in the pymod folder of the FWTools package. This created the map grid for these maps.

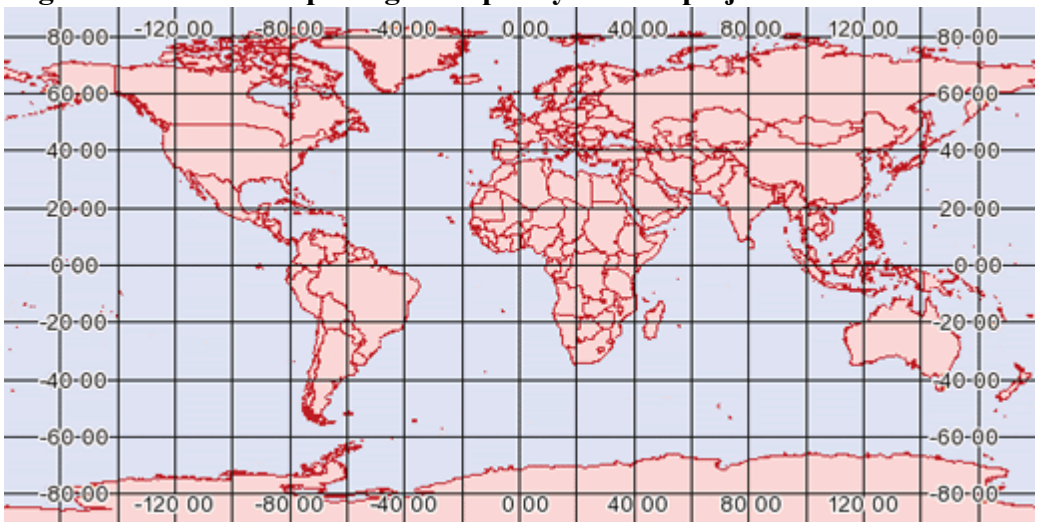
For more on FWTools, see http://fwtools.maptools.org/.

A.3.1. Simple Cylindrical Projection

Figure A-6 shows the most basic world map. The global coordinates, latitudes, and longitudes are displayed on a rectangular grid. Each rectangle is the same size on both x and y planes. This is the default projection used with MapServer when you specify the latlong projection. It is commonly known as a Simple Cylindrical projection, more technically known as Plate Carré. It is from the family of Equidistant Cylindrical projections. See the other references listed at the end of this chapter for more information about these types of projections.

This projection is most useful near the equator because the features are distorted as you move away from the equator. The features of Antarctica in Figure A-6 are a good example of how distorted it can be.

Figure A-6. Global map using a Simple Cylindrical projection



A.3.1.1 MapServer syntax

Specifying this projection is easy. All you need to set is the projection name and ellipsoid:

A.4. Using Projections with Other Applications

MapServer is just one application of many that use PROJ.4 to help project coordinates. Two other utilities discussed elsewhere in this book can use the power of PROJ.4 to project data/features into new projections. This is often accomplished using a few command-line options. The gdalwarp and ogr2ogr command-line tools (part of the GDAL/OGR package hosted at http://gdal.org) allow you to convert raster and vector data, respectively, between formats. You can also specify a target spatial reference system using the -t_srs option. For example, to convert an image that is referenced to latitude/longitude coordinates, into a Mercator projection, the following command might be used:

```
> gdalwarp -t_srs "+proj=merc +ellps=WGS84" in.tif out.tif
```

There is also an option, -s_srs, that allows you to specify the source SRS. This is essential if the data you have isn't already encoded with projection information.

Here is another example; it uses the ogr2ogr vector conversion utility to convert a shapefile from one SRS to another. The source SRS is also supplied because there was no projection information (or *.prj* file) supplied with the data.

```
> ogr2ogr -s_srs "+proj=latlong" -t_srs "+proj=merc +ellps=WGS84" city_merc.shp
city_in.shp
```

A.5. References

This appendix is only an introduction to projections. For more information about map projections, technical specifics, or cartographic principles, please refer to the following online resources:

Museums Teaching Planet Earth web site

http://www.geog.ucsb.edu/~dylan/mtpe/geosphere/topics/rs/howis.html

PROJ.4: Cartographic Projections Library

The main PROJ.4 web site includes links to more technical documents that describe the math and parameters involved with specific projections. See the references section at the bottom of the main web page for more links at http://proj.maptools.org.

Atlas of Canada's Map Making learning resources

This resource includes significant information on map projections (http://atlas.gc.ca/site/english/learningresources/carto_corner/index.html).

United States Geologic Survey (USGS) projection document

Check here for an overview of specific projections and theory: http://erg.usgs.gov/isb/pubs/MapProjections/projections.html.

GDAL/OGR libraries and utilities

Command-line utilities and libraries allow comprehensive access to raster (GDAL) and vector (OGR) data, including projection support; check out http://gdal.org.

Spherical trigonometry

This site has a simple tutorial focused on understanding the trigonometric concepts behind spherical measurements and navigation; find it at http://www.dynagen.co.za/eugene/where/sphertrg.html.

Coordinates, datums and transformations

Here you find many links to articles, papers and country-specific map projection information http://www.ferris.edu/htmls/academics/course.offerings/burtchr/geodesy/datums.html.

Appendix B. MapServer Reference Guide for Vector Data Access

This document, created by Jeff McKenna and Tyler Mitchell, was presented as part of a workshop given at the Open Source GIS/MapServer Users Meeting in Ottawa, Canada in 2004.

This is a comprehensive reference guide to using different vector data formats with MapServer. This guide is also available on the MapServer web site:
http://mapserver.gis.umn.edu/doc.html

If you have comments, additions or corrections, please contact one of the authors at:
Tyler Mitchell: tylermitchell@shaw.caJeff McKenna: jmckenna@dmsolutions.ca

B.1. Vector Data

What is vector data? This quote is a good description of what vector data is:

Vector: An abstraction of the real world where positional data is represented in the form of coordinates. In vector data, the basic units of spatial information are points, lines, and polygons. Each of these units is composed simply as a series of one or more coordinate points. For example, a line is a collection of related points, and a polygon is a collection of related lines. Vector images are defined mathematically as a series of points joined by lines. Vector-based drawings are resolution independent. This means that they appear at the maximum resolution of the output device, such as a printer or monitor. Each object is self-contained, with properties such as color, shape, outline, size, and position on the screen.

From: http://coris.noaa.gov/glossary/glossary_l_z.html#v

MapServer can access vector file formats and database connections. It can also access raster or image data. This is a summary of the vector and database formats with particular focus on how to use them in a MapServer map file and access them using command-line tools such as ogrinfo.

B.1.1. MapServer and Vector Data Access

MapServer offers two methods for accessing data:

Using built-in, format-specific, data access capabilities

The most basic form of data access uses the built-in capabilities that were linked into MapServer when it was compiled. These capabilities are limited to only a few types of vector data, such as ESRI shapefile, PostGIS, Oracle Spatial, and ArcSDE. The default, built-in format for MapServer is the ESRI shapefile format.

Using the capabilities of third-party data access libraries

One of the most powerful features of MapServer is the ability to use data formats through a pseudo plug-in environment. The most significant third-party library being used is GDAL/OGR. This includes raster (GDAL) and vector (OGR) data.

B.1.1.1 Using OGR

OGR is used behind the scenes by MapServer when requested. MapServer doesn't require OGR in order to run. Some users may never need the additional capabilities OGR offers, but many users find them absolutely necessary. Because MapServer can access data via OGR, you don't have to program specific types of data format support directly into MapServer. Instead, you can make further additions to OGR and then use MapServer. In essence, the background libraries allow MapServer to bring the data into an internal, memory-based format that MapServer can use. For the most part, using OGR-related formats is seamless and intuitive.

B.1.2. Data Format Types

There are three types of data mapping and GIS data formats. Each type is handled differently. Here are the types and some example formats:

File-based

Shapefiles, Microstation Design Files (DGN), GeoTIFF images

B.2. Data Format Guide

The rest of this document is the data format guide. This guide is structured to show the fundamentals of each MapServer-supported data format. Each section discusses one format, ranging from one to several pages in length. The sections typically start with a summary of the most important information about the format, followed by examples of file listings, connection methods, ogrinfo usage, and MapServer map file syntax examples.

Each section has been designed to stand alone, so you may notice that certain warnings and comments are repeated or redundant. This is intentional. Each format is presented in rough order of popular use, based on a survey of the MapServer community.

The following formats are included:

- ESRI shapefiles (SHP)
- PostGIS/PostgreSQL database
- MapInfo files (TAB/MID/MIF)
- Oracle Spatial Database
- Web Feature Service (WFS)
- Geography Markup Language (GML)
- VirtualSpatialData (ODBC/OVF)
- TIGER/Line files
- ESRI Arc Info coverage files (ADF)
- ESRI ArcSDE database (SDE)
- Microstation design files (DGN)
- IHO S-57 files
- Spatial Data Transfer Standard files (SDTS)
- Inline MapServer features
-

Also known as ESRI ArcView. ESRI is the company that introduced this format; ArcView was the first product to use shapefiles.

File listing

Shapefiles are made up of a minimum of three similarly named files, with different suffixes: Countries_area.dbfCountries_area.shpCountries_area.shx

Data access/connection method

- Shapefile access is built directly into MapServer. It is also available through OGR, but direct access without OGR is recommended and discussed here.
- The path to the shapefile is required. No file extension should be specified.
- Shapefiles hold only one layer of data, therefore no distinction needs to be made.

ogrinfo examples

- The directory can serve as a data source.
- Each shapefile in a directory serves as a layer.
- A shapefile can also be a data source. In this case the layer has the same prefix as the shapefile.

Here's an example that uses ogrinfo on a directory with multiple shapefiles:

```
> ogrinfo /data/shapefiles/

INFO: Open of '/data/shapefiles/'

using driver 'ESRI Shapefile' successful.

1: wpg_h2o (Line String)

2: wpg_roads (Line String)

3: wpg_roads_dis (Line String)

4: wpgrestaurants (Point)
```

Here's an example that uses ogrinfo on a single shapefile:

```
> ogrinfo /data/shapefiles/Countries_area.shp

Had to open data source read-only
```

PostGIS is Refraction Research's spatial extension to the PostgreSQL enterprise database.

PostGIS support

PostGIS is supported directly by MapServer and must be compiled into MapServer to work.

In most cases, PostgreSQL and PostGIS libraries (*.dll* or *.so*) must be present in the system's path environment for functionality to be present. This includes the libpq and libpostgis libraries.

Map file example

- Specify CONNECTIONTYPE POSTGIS.
- Define CONNECTION as:

```
"host=yourhostname dbname=yourdatabasename user=yourdbusername
password=yourdbpassword port=yourpgport"
```

- CONNECTION parameters can be in any order. Most are optional. dbname is required. host defaults to localhost, port defaults to 5432--the standard port for PostgreSQL.
- Define DATA as: "geometrycolumn from yourtablename". MapServer had a bug related to the keyword from. Specify it in lowercase to avoid problems. geometrycolumn can be the_geom if the shp2pgsql utility is used to load data, or wkb_geometry if ogr2ogr is used.

For example:

```
LAYER
  NAME pg_test
  TYPE POLYGON
  CONNECTIONTYPE POSTGIS
  CONNECTION "host=mapserver.com dbname=gmap user=julio"
  DATA "wkb_geometry FROM province"
  CLASS
     ...
  END
END
```

MapInfo files are also known as TAB or MID/MIF files.

File listing

The *.DAT*, *.ID*, *.MAP* files are also associated with .TAB files. Here's an example:

```
border.DAT border.ID border.MAP border.TAB
```

The term MID/MIF refers to files with .MID and *.MIF* extension.

Data access/connection method

- TAB and MID/MIF access is available in MapServer through OGR.
- The CONNECTIONTYPE OGR parameter must be used.
- The path to the (**.tab* or **.mif*) file is required, and the file extension is needed.
- The path may be relative to the SHAPEPATH

MapInfo files already contain styling information. This styling information can be used optionally by specifying the STYLEITEM "AUTO" parameter in the LAYER object of the map file.

If you use STYLEITEM "AUTO", you must have an empty class in the layer.

ogrinfo examples

Here's an example that uses ogrinfo on a single TAB file:

```
> ogrinfo elev5_poly.TAB
Had to open data source read-only.
INFO: Open of 'elev5_poly.TAB'
using driver 'MapInfo File' successful.
1: elev5_poly (Polygon)
```

Here's an example that uses ogrinfo to examine the structure of the file/layer:

```
> ogrinfo elev5_poly.TAB elev5_poly
Had to open data source read-only.
```

- MapServer can support Oracle Spatial through OGR.
- OGR must be compiled with Oracle Spatial support and MapServer must be compiled to use OGR.
- MapServer also supports Oracle Spatial natively.

For more information about Oracle Spatial and MapServer see the MapServer documentation and reference pages at http://mapserver.gis.umn.edu/doc.html.

Map file example using OGR support

```
LAYER
  ...
  CONNECTION "OCI:user/pwd@service"
  CONNECTIONTYPE OGR
  DATA "Tablename"
  ...
END
```

Example:

```
LAYER
   ...
   NAME "Ottawa"
   CONNECTIONTYPE OGR
   CONNECTION "OCI:jeff/blah@ora_cities"
   DATA "CITIES"
   TYPE POINT
   ...
END
```

Map file example using native support

```
LAYER
```

WFS is an Open Geospatial Consortium specification. For more information about the format itself, see http://www.opengeospatial.org/.

WFS allows a client to retrieve geospatial data encoded in Geography Markup Language (GML) from multiple Web Feature Services. GML is built on the standard web language XML.

WFS differs from the popular Web Map Service specification in that WFS returns a subset of the data in valid GML format, not just a graphic image of data.

Capabilities

Requesting capabilities using the GetCapabilities request to a WFS server returns an XML document showing what layers and projections are available, etc.

Example of a WFS GetCapabilities URL

```
http://www2.dmsolutions.ca/cgi-bin/mswfs_gmap
?VERSION=1.0.0
&SERVICE=wfs
&REQUEST=GetCapabilities
```

Example of the resulting XML from GetCapabilties

Example B-1. Resulting XML from GetCapabilities

```
...
<FeatureTypeList>
 <Operations>
  <Query/>
 </Operations>
   <FeatureType>
    <Name>park</Name>
    <Title>Parks</Title>
    <SRS>EPSG:42304</SRS>
    <LatLongBoundingBox minx="-173.433" miny="41.4271" maxx="-13.0481" maxy="83.7466" />
  </FeatureType>
  <FeatureType>
    <Name>road</Name>
```

- Also known as Geographic Markup Language and GML/XML.
- GML is a text-based, XML format that can represent vector and attribute data.
- This is an Open Geospatial Consortium specification for data interchange.

File listing

GML files are usually a single text file with a GML filename extension (*coal_dep.gml*). Some may use XML as the filename extension.

XML schema documents often accompany GML files that have been translated from some other format (e.g., using the ogr2ogr utility).

Example of text in a GML file

GML uses sets of nested tags to define attributes and geometry coordinates:

```
<gml:featureMember>
  <Coal_Deposits f>
    <UNKNOWN>0.000</UNKNOWN>
    <NA>0.000</NA>
    <ID>2</ID>
    <ID2>2</ID2>
    <MARK>7</MARK>
    <COALKEY>110</COALKEY>
    <COALKEY2>110</COALKEY2>
    <ogr:geometryProperty>
      <gml:Point>
        <gml:coordinates>78.531,50.694</gml:coordinates>
      </gml:Point>
    </ogr:geometryProperty>
  </Coal_Deposits>
</gml:featureMember>
```

Data access/connection method

This is an OGR extension to MapServer. It allows you to connect to databases that don't explicitly hold spatial data, as well as flat text files. Your data must have an x and a y column, and the data may be accessed through an ODBC connection or a direct pointer to a text file.

Types of databases

The VirtualSpatialData OGR extension has been tested with the following databases and should, in theory, support all ODBC data sources.

- Oracle
- MySQL
- SQL Server
- Access
- PostgreSQL

Types of flat files

Comma, tab, or custom delimited text/flat files work with VirtualSpatialData.

Create the data source name (DSN)

Specific notes about creating a DSN on Windows and Linux can be found by searching the MapServer reference documents site at http://mapserver.gis.umn.edu/doc.

On some Windows systems you must create a SYSTEM DSN.

Test your connection

Test your connection with ogrinfo. The syntax for this command is:

```
> ogrinfo ODBC:user/pass@DSN table
```

ogrinfo examples

Here's an example that accesses a comma-separated text file through ODBC; it's a flat text file *coal_dep.txt* containing lat/long points:

```
unknown,na,id,id2,mark,coalkey,coalkey2,long,lat

0.000,0.000,1,1,7,87,87,76.90238,51.07161
```

TIGER/Line files are created by the U.S. Census Bureau and cover the entire United States. They are often referred to simply as TIGER files. For more information, see http://www.census.gov/geo/www/tiger/.

File listing

TIGER/Line files are text files and directory-based data sources. For example, one county folder TGR06059 contains several associated files:

```
TGR06059.RT1 TGR06059.RT2 TGR06059.RT4 TGR06059.RT5

TGR06059.RT6 TGR06059.RT7 TGR06059.RT8 TGR06059.RTA

TGR06059.RTC TGR06059.RTH TGR06059.RTI TGR06059.RTP

TGR06059.RTR TGR06059.RTS TGR06059.RTT TGR06059.RTZ
```

Data access/connection method

- TIGER/Line access occurs through OGR.
- The full path to the directory containing the associated files is required in the CONNECTION string. The layer number is added to the CONNECTION string, after the path, separated by a comma: for example., CONNECTION "/tiger/data,0".
- The layer number in the map file is actually the ogrinfo layer number, minus one.

ogrinfo examples

Here's an example that uses ogrinfo on a TIGER directory to retrieve layer numbers:

```
> ogrinfo TGR06059 (NOTE that this is a directory)

ERROR 4: Tiger Driver doesn't support update.

Had to open data source read-only.

INFO: Open of 'TGR06059'

using driver 'TIGER' successful.

1: CompleteChain (Line String)

2: AltName (None)

3: FeatureIds (None)

4: ZipCodes (None)

5: Landmarks (Point)
```

ESRI ArcInfo coverage files are also known simply as coverages and, less commonly, as ADF files.

File listing

Coverages are made up of a set of files within a folder. The folder itself is the coverage name. The files roughly represent different layers, usually representing different types of topology or feature types.

```
> ls /data/coverage/brazil

aat.adf  arc.adf  arx.adf  bnd.adf  lab.adf  prj.adf  tic.adf  tol.adf
```

A folder with the name INFO is also part of the coverage. It sits at the same hierarchical level as the coverage folder itself. Therefore, to copy a coverage (using regular filesystem tools), the coverage folder and the INFO folder must both be copied. The INFO folder holds some catalog information about the coverage:

```
> ls /data/coverage/info

arc0000.dat  arc0001.dat  arc0002.dat  arc.dir

arc0000.nit  arc0001.nit  arc0002.nit
```

Data access/connection method

- CONNECTIONTYPE OGR must be used. The ability to use coverages isn't built into MapServer.
- The path to the coverage folder name is required.
- The layer number is used to specify what type of features to draw (as per the layer number from ogrinfo, but minus one).

ogrinfo examples

- The directory is the data source.
- Layers are found within the directory.

Here's an example that uses ogrinfo on a coverage directory:

```
> ogrinfo /data/coverage/brazil

INFO: Open of 'brazil'

using driver 'AVCBin' successful.
```

ArcSDE is ESRI's spatial plug-in for SQL Server, Oracle, and DB2 databases. It is also known as Spatial Database Engine, but most commonly as simply SDE.

SDE support

- SDE is supported directly by MapServer.
- MapServer 4.4 supports SDE 9, 8, and SDE for coverages on Linux, Windows, and Solaris platforms.
- Recent developments in MapServer (v4.4) have added support for SDE raster layers. Using raster layers is beyond the scope of this guide.

Connecting to SDE

1. Install the SDE client libraries from the SDE CDs.
2. Compile MapServer with SDE support.
3. Define the layer in the map file.

Map file example

1. Specify CONNECTIONTYPE SDE
2. Define the CONNECTION as: *hostname*,*instancename*,*databasename*,*username*,*password*
3. Define the DATA as: *tablename*, *geometrycolumn*

For example:

```
LAYER
  NAME test
  TYPE POLYGON
  CONNECTION "sde.dms.ca,port:5151,sde,user,password"
  CONNECTIONTYPE SDE
  DATA "NTDB.WATER,SHAPE"
  CLASS
    ...
```

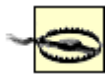
These are also known as DGN files.

File listing

Data is encapsulated in a single file, usually with the suffix .dgn; for example: *0824t.dgn*.

Data access/connection method

- Access is available in MapServer through OGR.
- The CONNECTIONTYPE OGR parameter must be used.
- The path to the dgn file is required; file extension is needed.
- All types of features in a DGN file are held in one "layer" of data. The layer is called elements and is the first and only layer.
- The type of feature to be read from the DGN depends on the TYPE parameter in the map file.
- DGN files typically contain POINT, LINE, POLYGON, and ANNOTATION feature types.
- DGN files contain "styling" information, i.e., how to color and present the data. This is used, optionally, by specifying the STYLEITEM "AUTO" parameter.



DGN files typically use white as a color for their features and therefore aren't visible on maps with white backgrounds.

ogrinfo examples

Using ogrinfo on a single DGN file

Note that no geometry/feature type for the layer is identified because it can be multiple types.

```
> ogrinfo /data/dgn/0824t.dgn

Had to open data source read-only.

INFO: Open of '0842t.dgn'

using driver 'DGN' successful.

1: elements
```

- Also known as S57. The IHO S-57 format is a vector interchange format used for maritime charts.
- Developed by the International Hydrographic Organization (IHO). For more information about the IHO see http://www.iho.shom.fr/.

File listing

Individual S57 data files have an extension of *.000; for example, US1BS02M.000.

Data access/connection method

- S57 access in MapServer occurs through OGR; CONNECTIONTYPE OGR must be used.
- Specify a full path or a relative path from the SHAPEPATH to the .000 file for the CONNECTION.
- The CONNECTION must also include a layer number (as per the layer number from ogrinfo, but minus one).

Special notes

The underlying OGR code requires two files from your GDAL/OGR installation when reading S57 data in MapServer: s57objectclasses.csv and s57attributes.csv. These files can be found in the /GDAL/data/ folder. If you receive an error in MapServer such as:

```
    msDrawMap(): Image handling error. Failed to draw layer named 's57'. msOGRFileOpen(
):

    OGR error. GetLayer( 9) failed for OGR connection
```

you may have to point MapServer to these files using the CONFIG parameter in the main section of your map file:

```
    CONFIG GDAL_DATA "C:\gdal\data"
```

ogrinfo examples

Here's an example that uses ogrinfo on an S57 file to get the OGR index number:

```
    > ogrinfo us1bs02m.000

    ERROR 4: S57 Driver doesn't support update.

    Had to open data source read-only.

    INFO: Open of 'us1bs02m.000'
```

This is a U.S. Geological Survey (USGS) format. SDTS has a raster and a vector format. The raster format isn't supported in MapServer; only the vector formats are supported, including VTP and DLG files.

File listing

SDTS files are often organized into state-sized pieces; for example, all pf the state of Maryland, U.S.A.

Files are also available for multiple types of features including hydrography, transportation, and administrative boundaries.

This example uses transportation data, which consists of 35 separate files, each with the suffix DDF.

```
MDTRAHDR.DDF  MDTRARRF.DDF  MDTRCATS.DDF
MDTRDQCG.DDF  MDTRFF01.DDF  MDTRLE02.DDF
MDTRNA03.DDF  MDTRNO03.DDF  MDTRSPDM.DDF
MDTRAMTF.DDF  MDTRBFPS.DDF  MDTRCATX.DDF
MDTRDQHL.DDF  MDTRIDEN.DDF  MDTRLE03.DDF
MDTRNE03.DDF  MDTRPC01.DDF  MDTRSTAT.DDF
MDTRARDF.DDF  MDTRBMTA.DDF  MDTRDDSH.DDF
MDTRDQLC.DDF  MDTRIREF.DDF  MDTRNA01.DDF
MDTRNO01.DDF  MDTRPC02.DDF  MDTRXREF.DDF
MDTRARDM.DDF  MDTRCATD.DDF  MDTRDQAA.DDF
MDTRDQPA.DDF  MDTRLE01.DDF  MDTRNA02.DDF
MDTRNO02.DDF  MDTRPC03.DDF
```

Data access/connection method

- SDTS access is available in MapServer through OGR.
- The CONNECTIONTYPE OGR parameter must be used.
- The path to the catalog file (????CATD.DDF) is required, including file extension.
- There are multiple layers in the SDTS catalog, some of which are only attributes and have no geometries.

ogrinfo examples

Inline features refer to coordinates entered directly into the map file. They aren't a file or database format and don't require any DATA or CONNECTION parameters. Instead they use a FEATURE section to define the coordinates.

Inline features can be used to define points, lines, and polygons as if taken from an external file; this requires direct entry of coordinate pairs in the map file using a particular syntax.

Data access/connection method

This is a native MapServer option that doesn't use any external libraries to support it.

Map file example

Each FEATURE..END section defines a feature.

Points

Multiple points can be defined in a FEATURE section. If multiple points are defined in the same layer, they have the same CLASS settings; for example, for colors and styles.

Coordinates are entered in the units set in the layer's projection. In this case, it assumes the map file projection is using decimal degrees.

```
LAYER
  NAME inline_stops
  TYPE POINT
  STATUS DEFAULT
  FEATURE
    POINTS
      72.36 33.82
    END
    TEXT "My House"
  END
  FEATURE
    POINTS
      69.43 35.15
      71.21 37.95
      72.02 38.60
    END
    TEXT "My Stores"
  END
```

NTF files are mostly used by the U.K. Ordnance Survey (OS). For more on the Ordnance Survey, see their web site at http://www.ordnancesurvey.co.uk.

File listing

NTF files have an NTF extension.

Data access/connection method

- NTF access requires OGR.
- The path to the NTF file is required in the CONNECTION string. It may be relative to the SHAPEPATH setting in the map file or the full path.
- The CONNECTION must also include a layer number (as per the layer number from ogrinfo, but minus one).

ogrinfo examples

Here's an example that uses ogrinfo on an NTF file to retrieve layer numbers:

```
> ogrinfo llcontours.ntf

ERROR 4: NTF Driver doesn't support update.

Had to open data source read-only.

INFO: Open of 'llcontours.ntf'

using driver 'UK .NTF' successful.

1: LANDLINE_POINT (Point)

2: LANDLINE_LINE (Line String)

3: LANDLINE_NAME (Point)

4: FEATURE_CLASSES (None)
```

For the LANDLINE_LINE layer, the ogrinfo layer number is 2; however, when referring to this layer in a map file connection, the layer number is 1.

Here's an example that uses ogrinfo to examine the structure of an NTF layer:

```
> ogrinfo llcontours.ntf LANDLINE_LINE

ERROR 4: NTF Driver doesn't support update.

Had to open data source read-only.
```

Colophon

About the Author

Colophon

About the Author

Tyler Mitchell is a geographer and open source enthusiast. He works as a Geographic Information Systems (GIS) manager for Timberline Forest Inventory Consultants and lives in beautiful British Columbia, Canada. He is a regular speaker, moderator, and workshop leader at GIS conferences. His foray into the open source world began while looking for alternatives to proprietary mapping tools. He is now a devoted open source GIS advocate.

Colophon

Our look is the result of reader comments, our own experimentation, and feedback from distribution channels. Distinctive covers complement our distinctive approach to technical topics, breathing personality and life into potentially dry subjects.

The animal on the cover of Web Mapping Illustrated is a common snipe (Gallinago gallinago). Snipe are medium-sized (about 10 inches), wading shorebirds with short legs, pointed wings, and long, straight bills. Both sexes have a strongly patterned back with several buff, longitudinal stripes, a white belly, and dark bars on their flanks.

When flushed, the snipe rises away in a zigzag flight pattern. The flight call resembles a short rasping sneeze. In the spring, males produce an aerial drumming display using outstretched outer tail feathers to generate a low-pitched, whirring sound that attracts interested female partners. Once a female shows interest, the male pursues her and dives with wings held above the body in a V-shape, often rolling and turning upside down. Snipe nest on the ground in thick vegetation on wetlands or pasture where there is easy access to soft ground and small shallow pools. Females produce up to four eggs in 18 to 20 days; the babies are ready to fledge in 19 to 20 days.

Mary Anne Weeks Mayo was the production editor and copyeditor for Web Mapping Illustrated. Leanne Soylemez proofread the book. Adam Witwer and Claire Cloutier provided quality control. Peter Ryan provided production assistance. Julie Hawks wrote the index.

Ellie Volckhausen designed the cover of this book, based on a series design by Edie Freedman. The cover image is a loose antique engraving. Karen Montgomery produced the cover layout with Adobe InDesign CS using Adobe's ITC Garamond font.

David Futato designed the interior layout. This book was converted by Keith Fahlgren to FrameMaker 5.5.6 with a format conversion tool created by Erik Ray, Jason McIntosh, Neil Walls, and Mike Sierra that uses Perl and XML technologies. The text font is Linotype Birka; the heading font is Adobe Myriad Condensed; and the code font is LucasFont's TheSans Mono Condensed. The illustrations that appear in the book were produced by Robert Romano, Jessamyn Read, and Lesley Borash using Macromedia FreeHand MX and Adobe Photoshop CS. The tip and warning icons were drawn by Christopher Bing. This colophon was compiled by Mary Anne Weeks Mayo.

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

&& operator and spatial queries
3D perspective
3D view
  combining DEM and drape images
  in OpenEV
  navigating
  vertical exaggeration
< = operator
<= operator
= operator
> = operator

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

accuracy of data
acquiring map data 2nd
  3D perspective
  air photos
  appraising data needs
  data exchange agreement
  data repositories
  derived products
  GPS
  Internet map data sources
  premade maps
  resources
  satellite images
  shaded surface maps
  street maps
  unavailable data
ADF file extension
air photos
airports shapefile
  layer
Albers Equal Area Projection
AND operator
application development frameworks
appraising data needs
  raster data, types of
  scale of map
  vector data
    types of
    versus raster data
ARC.ADF file
ArcExplorer
ArcGIS ArcView
ArcIMS emulator
areas/polygons data
Atlas of Canada's Map Making learning resources web site
awk command (Unix)

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

base maps
  grabbing base map layer from WMS
  identifying requirements for
  photos unsuitable for
BGC Engineering Inc.
British Columbia (Canada)

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

CADD programs
Camptocamp SA
Canada GeoBase Portal
Canada, satellite imagery
Canadian Government
capabilities document 2nd
  checking
Cartesian Plane
Cave-Ayland, Mark
CCGIS
Chameleon
  demo
  web site
CIA World Factbook
CLASS
CLASS objects in an WMS layer
CLASSITEM
classObj object
color depth, increasing
color theming map
colorObj object
colrm command (Unix)
column command (Unix)
command-line tools
  examples
    adding labels to map
    adding scale bar
    color theming map
    creating legend
    creating simple MapServer map
    final map file
    ogrinfo, using to examine contents of world countries shapefile
    shp2img
    shp2img, selecting different EXTENT
  legend
  scalebar
  shp2img
  shp2pdf
  shptree
  sortshp
  tile4ms
commercial GIS and mapping programs
common mapping pitfalls
common mapping tasks
Concurrent Versioning System (CVS) tools
configuration and compilation process
conic projections
converting and viewing
converting map data
  creating JPEG preview of satellite image

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

data
  accuracy
  content, examining
  format guide
    format types
      database connections
      directory-based data
      file-based data
    formats
      proprietary
      understanding
  management tools, downloading
  mapping
  mapping types
  repositories
  transforming between coordinate systems
database connections
database query statements (SQL)
databases
  spatial
  tabular
DC Maintenance Management System (DCMMS) web site
Debian Linux package repositories
DebianGIS
decimal degrees versus degrees minutes seconds
Digital Elevation Model (DEM)
  combining with drape image
  preparing
digital mapping tools
digital maps
  difficulties
  modifying
  personal maps
  technology barriers
  versus conventional maps
digital tools, dependency on
digitizing
directory-based data
DM Solutions Group
  DM Solutions
drape image
  combining with DEM
Draw Area tool (OpenEV)
Draw Line tool (OpenEV)
Draw ROI tool (OpenEV)
  defining region of interest
duplicate lines, removing

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

Earth Observation Portal
elevation datasets
END keyword
EPSG
  code system
  codes
  European Petroleum Survey Group
  European Petroleum Survey Group code number
  projection codes versus project details
eq operator
Equidistant Cylindrical projections
ESRI ArcInfo Coverage files
  data access/connection method
  map file example
  ogrinfo examples
ESRI ArcInfo Coverages 2nd
ESRI ArcMap program
ESRI ArcSDE
ESRI ArcSDE Database (SDE)
  connecting to SDE
  map file example
  SDE support
ESRI ArcView
ESRI Shapefiles (SHP) 2nd 3rd
  data access/connection method
  map file example
  ogrinfo examples
expand/unexpand command (Unix)
Export tool (OpenEV)
EXPRESSION keyword
expressions, creating
EXTENT

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

Feature Manipulation Engine (FME) web site
FGS
  FGS project
  Free Open Source Software GIS Suite
file-based data
FILTER keyword
Flexible Internet Spatial Template (FIST) web site
fonts, TrueType
forest fire, drawing area of
free GIS/mapping products
free viewers
FreeGIS
  FreeGis.org list
Freshmeat web site
FWTools 2nd 3rd
  acquiring latest version
  web site

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

Gaia
GD library
GDAL/OGR
  GDAL 2nd
    support formats page
    utilities
  gdalinfo command
    -mm parameter
    image data
    raster data
  gdaltindex
GDAL/OGR (continued)
  (see also OGR)
  gdalwarp
    requirements of
  gdalwarp tool
  libraries and utilities web site
  package
gdal_translate
  creating JPEG preview of satellite image
  GDAL support formats page
  reducing image size
  selecting portions of data
  translating images to other formats
  translating portion of image
ge operator
Geo-Consortium
geo-referencing
  needed information
GeoConnections Discovery Portal
  web site
Geodan
Geographic Coordinate System
geographic data
  detailed information using ogrinfo
geographic databases
geographic information and maps
geographic information system (GIS), ix
GeoGratis data site
geoid
Geomatica FreeView
GeoMedia Viewer
geometric computations
Geometry Engine Open Source (GEOS)
GEOS web site
GeoServer
Geospatial Data Abstraction Library (GDAL) 2nd
  programming libraries
  raster formats supported
  SWIG and

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

Handyside Web Programming
head/tail command (Unix)
higher resolution boundaries
HostGIS 2nd

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

IBM Spatial Extender
IHO S-57 files
  data access/connection method
  map file example
  ogrinfo examples
  special notes
illustration of methods for describing the Earth's shape
image size, modifying to fit extent of map
IMAGECOLOR
images
  based on map file
  gdalinfo command
  multiple bands
  of legends
  PNG
  setting output image formats
    IMAGETYPE
  WBMP
  which formats supported
Inline MapServer features
  data access/connection method
  map file example
interactive maps 2nd 3rd 4th
Intergraph Microstation design files (DGN)
Intergraph OGC WMS Viewer
Internet map data sources
interoperability of software
Intevation

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

join command (Unix)
JPEG format
JUMP
  Java Unified Mapping Platform

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

Kralidis, Tom

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

LABELITEM setting
labels
  adding to map
  default types
Laboratório de Computação Aplicada - G10
Lambert Conformal Conic (LCC) 2nd
latitudes, from the equator to 90°
layer controls, adding to map web page
layerObj object
Layers window (OpenEV)
le operator
legend
  adding a link
  adding to a web page
  command
  creating
    embedded
      removing
  POSITION setting
  status settings (ON, OFF, or EMBED)
legend (MapServer utility)
legendObj object
legends, adding to an application
Lime, Steve
linear data
LinGIS
Linux Documentation Project site
Liyanaga, Marc
Locative Technologies
logical expressions
longitudes
  divided into hemispheres of 180° each
  running from 0 to 360°
look command (Unix)
lower-resolution boundaries
lt operator
lwpostgis.sql SQL script, loading

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

Mac OS X
Manifold
map data
  analyzing
map examples
  global map using the Simple Cylindrical projection
  map of Europe in Albers Equal Area Projection
  map of South Asia in Transverse Mercator Projection
  map of the Eastern Hemisphere using Orthographic Projection
  map of the North Pole in Stereographic Projection
  showing northern latitudes in Mercator Projection
map file
  comments
  creating map images
  diagram
  filename
  indenting lines
  keywords
  mapping file
  MapServer
  overview
  reference documents
  rules and recommendations
  values needing quotations
map file examples
  ESRI ArcInfo Coverage files
  ESRI ArcSDE Database (SDE)
  ESRI Shapefiles (SHP)
  GML
  IHO S-57 files
  Inline MapServer features
  MapInfo files (TAB/MID/MIF)
  Microstation Design files (DGN)
  National Transfer Format files (NTF)
  PostGIS/PostgreSQL Database
  Spatial Data Transfer Standard files (SDTS)
  TIGER/Line files
  using OGR support
  VirtualSpatialData (ODBC/OVF)
  WFS (Web Feature Service)
map image
  based on map file
Map Maker
map projections
  conic projections
  cylindrical projections
  examples
    Albers Equal Area Projection
    Mercator Projection
    orthographic projection

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

NASA Topography
National Geophysical Data Center Interactive Map Services web site
National Oceanic and Atmospheric Administration (NOAA)
National Transfer Format files (NTF)
  data access/connection method
  map file example
  ogrinfo examples
ne operator
nl command (Unix)

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

oblique air photos
observations
  locating positions of
  quantifying
  visualizing on a map
OGR 2nd
  project web site
  utilities
  utilities web site
  vector formats supported
ogr2ogr 2nd 3rd
  -where option
  as feature extraction tool
  converting PostGIS data
  CSV files and
  selecting portions of data
  transforming coordinates
  using to convert a shapefile into GML format
  using to convert shapefiles to other formats
  using with ogrinfo
ogrinfo
  --help parameter
  -sql option
  -sql parameter
  -summary option
  detailed information about geographic data
    examples
      ESRI ArcInfo Coverage files
      ESRI Shapefiles (SHP)
      GML
      IHO S-57 files
      MapInfo files (TAB/MID/MIF)
      Microstation Design files (DGN)
      National Transfer Format files (NTF)
      Spatial Data Transfer Standard files (SDTS)
  filtering output from
  listing data in shapefile
  on a single TAB file
  TIGER/Line files
      VirtualSpatialData (ODBC/OVF)
  using to examine contents of world countries shapefile
  using to examine structure of file/layer
  using to examine the structure of the file/layer
  using with the ogr2ogr conversion tool
ogrtindex
online street mapping applications
Open Geospatial Consortium (OGC) 2nd
  OGC Transactional Web Feature Server (WFS-T)
  Open Geospatial Consortium web services (OWS)
  specification

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

PAL.ADF file
paste command (Unix)
PDF file output
PeopleGIS
personal knowledge versus general knowledge
personal map data
  planning your map
    base map
    choosing scale and extent
    street maps
  preprocessing data
    clipping out area of interest
    drawing map features
pest-infected trees
PHP MapScript
  By Example HOWTO
pl/pgsql language support, enabling
planar/projected coordinate system
Plate Carré
PNG
  format
  image type
point data and OpenEV
Point Edit tool (OpenEV)
  using to draw location on map
polygon data
Polygon data
POSITION setting (legend)
PostGIS 2nd 3rd
  1.0
  accessing spatial data in other applications
  adding labels to maps
  ArcMap and
  binaries
  bounding box comparison operator (&&)
  components
PostGIS (continued)
  country polygons, querying
  databases 2nd
  Debian Linux environment
  Distance( ) function
  documentation web site
  drawing labels
  find the country for a given point
  for Linux
  for Windows
  functionality testing
  install packages
  layers in MapServer
  Mac OS X environment

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

Quantum GIS (QGIS) 2nd 3rd 4th
query results, visualizing

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

RADARSAT
raster data
  gdalinfo command
  types of
  versus vector
  versus vector data
recenter, adding to map web page
reference map
  adding to application
  files
Refractions Research 2nd
  OGC Services Survey web site
  web site
Refractions WMS Extension for ArcView 3
relational database management systems (RDBMS)
reprojecting source data

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

s57objectclasses.csv and s57attributes.csv files
satellite images
  viewing statistics for
Scalable Vector Graphic (SVG)
scale bar
  adding
  adding to application
  INTERVALS 3
  purpose of
  scale of map
  scalebar (MapServer utility)
  scalebar command-line utility
  SIZE x y
  STATUS EMBED option
  TRANSPARENT
  UNITS KILOMETERS
SCALEBAR object
scalebarObj object
scanned maps
Schema editor (OpenEV)
sed command (Unix)
  finding specific patterns
  reformatting print results
  removing lines and removing front end of lines
shaded surface maps
shapefiles
  converting to GML
  index
  limitations
  sorting
  that points to location of individual files and their extent rectangles
shapes, highlighting subsets
shp prefix
shp2img
  selecting different EXTENT
  syntax for using
shp2img (MapServer utility)
shp2pdf (MapServer utility)
shp2pgsql
shptree (MapServer utility)
Simple Cylindrical projections
Simple Features Specification for SQL (SFSQL)
Simplified Wrapper and Interface Generator (SWIG) 2nd
size of map
sort command (Unix)
  creating list of ordered elevations
sortshp (MapServer utility)
source data
  locating
Sourceforge web site

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

T-MAPY
TAB files
tabular databases
tabular indexes
Tabular Shapes Attribute (OpenEV)
Tabular Shapes Grid (OpenEV)
TatukGIS Viewer
Terraview
Texas Online Map Library
text-processing tools
  for non-GNU platforms
Thuban 2nd
TIFF file
TIGER file format (U.S. Census)
  TIGER/Line files
    data access/connection method
    map file example
    ogrinfo examples
Tikiwiki web site
tile4ms (MapServer utility)
transforming data between coordinate systems
Transverse Mercator (TM) Projection
TrueType fonts
tsort command (Unix)
TYDAC

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

U.S. Geologic Survey
U.S. Geologic Survey (USGS) projection document
U.S. Maps and Data
U.S. National Atlas
U.S. National Geophysical Data Center (NGDC)
  various types of data
U.S. National Geospatial Data Clearinghouse
uniq command (Unix)
  -c option
  -d option
United Nations
UNITS keyword
Universal Transverse Mercator (UTM) 2nd
Unix commands
  awk
  colrm
  column
  cut
  expand/unexpand
  grep
    using to show only names of airports
  head/tail
  join
  look
  nl
  paste
  sed
    finding specific patterns
    reformatting print results
    removing lines and removing front end of lines
  sort
    creating list of ordered elevations
  text-processing
  tsort
  uniq
    -c option
  uniq-d option
  wc
User-friendly Desktop GIS (UDIG)
USGS DEM
  Digital elevation models for the United States
UTM (Universal Transverse Mercator)
  projection 2nd
  zone map showing, roughly, the location of the zones

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

Vasudevan, Venu
vector data
  access
    MapServer methods
    reference guide
  defined
  types of
  versus raster data 2nd
viewers, free
viewing
viewing tools
VirtualSpatialData (ODBC/OVF)
  creating DSN
  flat files
  map file example
  ogrinfo examples
  testing connection
  types of databases supported

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

Warmerdam, Frank
WBMP images
wc command (Unix)
web mapping
  applications
  servers
web maps
  end users
  servers
  service providers
  web sites
web servers
web services
  accessing maps via
  documentation web site
  internal versus external
  overview
  using with MapServer
WebMapIt
webObj object
WFS (Web Feature Service) 2nd 3rd 4th
  adding layer to a MapServer map file
  capabilities document, checking
  data access/connection method
  data, requesting manually
  documentation for adding support to map files
  GetCapabilities request
  map file example
  provider capabilities
  using MapServer as a server
  WFS request directly through browser
WKT (well-known text format)
WMS (Web Map Service) 2nd
  adding layer to a MapServer map file
  capabilities document
  CLASS objects in a layer
  grabbing base map layer from
  interoperability and
  manually requesting a map
  MapServer as a WMS server
  parameters
  projections in layers in MapServer
Workshop (sample MapServer application)

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

XML format for capability documents
XML-based datafiles

Index

[SYMBOL] [A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X] [Z]

zoom, adding to map web page