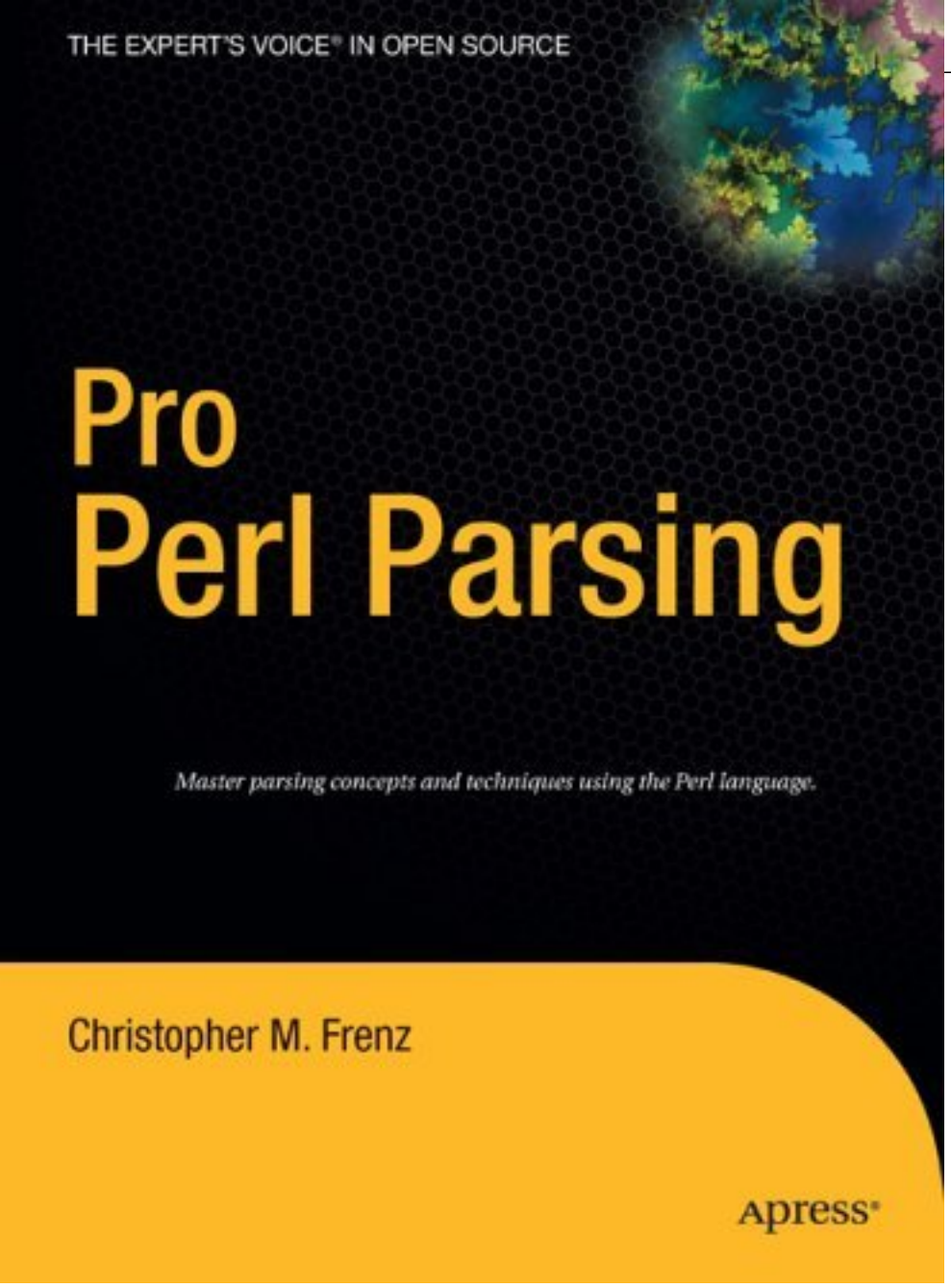
THE EXPERT'S VOICE® IN OPEN SOURCE
Pro
Perl Parsing
Master parsing concepts and techniques using the Perl language.
Christopher M. Frenz
Apress®

Pro Perl Parsing

Christopher M. Frenz

Apress®

Pro Perl Parsing

Lead Editors: Jason Gilmore and Matthew Moodie
Technical Reviewer: Teodor Zlatanov
Editorial Board: Steve Anglin, Dan Appleman, Ewan Buckingham, Gary Cornell, Tony Davis, Jason Gilmore, Jonathan Hassell, Chris Mills, Dominic Shakeshaft, Jim Sumser
Associate Publisher: Grace Wong
Project Manager: Beth Christmas
Copy Edit Manager: Nicole LeClerc
Copy Editor: Kim Wimpsett
Assistant Production Director: Kari Brooks-Copony
Production Editor: Laura Cheu
Compositor: Linda Weidemann, Wolf Creek Press
Proofreader: Nancy Sixsmith
Indexer: Tim Tate
Artist: Wordstop Technologies Pvt. Ltd., Chennai
Cover Designer: Kurt Krames
Manufacturing Manager: Tom Debolski

Library of Congress Cataloging-in-Publication Data

Frenz, Christopher.
Pro Perl parsing / Christopher M. Frenz.
p. cm.
Includes index.
ISBN 1-59059-504-1 (hardcover : alk. paper)
1. Perl (Computer program language) 2. Natural language processing (Computer science) I. Title.
QA76.73.P22F72 2005
005.13'3--dc22

2005017530

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit http://www.springeronline.com.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit http://www.apress.com.

The source code for this book is available to readers at http://www.apress.com in the Downloads section.

For Jonathan!
You are the greatest son
any father could ask for.

Contents at a Glance

About the Author xiii
About the Technical Reviewer xv
Acknowledgments xvii
Introduction xix

CHAPTER 1 Parsing and Regular Expression Basics 1
CHAPTER 2 Grammars 37
CHAPTER 3 Parsing Basics 63
CHAPTER 4 Using Parse::Yapp 85
CHAPTER 5 Performing Recursive-Descent Parsing with Parse::RecDescent 109
CHAPTER 6 Accessing Web Data with HTML::TreeBuilder 137
CHAPTER 7 Parsing XML Documents with XML::LibXML and XML::SAX 161
CHAPTER 8 Introducing Miscellaneous Parsing Modules 185
CHAPTER 9 Finding Solutions to Miscellaneous Parsing Problems 201
CHAPTER 10 Performing Text and Data Mining 217

INDEX 243

Contents

About the Author xiii
About the Technical Reviewer xv
Acknowledgments xvii
Introduction xix

CHAPTER 1 Parsing and Regular Expression Basics 1

Parsing and Lexing 2
Parse::Lex 4
Using Regular Expressions 6
A State Machine 7
Pattern Matching 12
Quantifiers 14
Predefined Subpatterns 15
Posix Character Classes 16
Modifiers 17
Assertions 20
Capturing Substrings 24
Substitution 26
Troubleshooting Regexes 26
GraphViz::Regex 27
Using Regexp::Common 28
Regexp::Common::Balanced 29
Regexp::Common::Comments 30
Regexp::Common::Delimited 30
Regexp::Common::List 30
Regexp::Common::Net 31
Regexp::Common::Number 31
Universal Flags 32
Standard Usage 32
Subroutine-Based Usage 33
In-Line Matching and Substitution 34
Creating Your Own Expressions 35
Summary 36

CHAPTER 2 Grammars 37

Introducing Generative Grammars 38
Grammar Recipes 39
Sentence Construction 41
Introducing the Chomsky Method 42
Type 1 Grammars (Context-Sensitive Grammars) 44
Type 2 Grammars (Context-Free Grammars) 48
Type 3 Grammars (Regular Grammars) 54
Using Perl to Generate Sentences 55
Perl-Based Sentence Generation 56
Avoiding Common Grammar Errors 59
Generation vs. Parsing 60
Summary 61

CHAPTER 3 Parsing Basics 63

Exploring Common Parser Characteristics 64
Introducing Bottom-Up Parsers 65
Coding a Bottom-Up Parser in Perl 68
Introducing Top-Down Parsers 73
Coding a Top-Down Parser in Perl 74
Using Parser Applications 78
Programming a Math Parser 80
Summary 83

CHAPTER 4 Using Parse::Yapp 85

Creating the Grammar File 85
The Header Section 86
The Rule Section 87
The Footer Section 88
Using yapp 94
The -v Flag 99
The -m Flag 103
The -s Flag 103
Using the Generated Parser Module 104
Evaluating Dynamic Content 105
Summary 108

CHAPTER 5 Performing Recursive-Descent Parsing with Parse::RecDescent 109

Examining the Module's Basic Functionality 109
Constructing Rules 111
    Subrules 112
Introducing Actions 115
    @item and %item 116
    @arg and %arg 117
    $return 118
    $text 120
    $thisline and $prevline 120
    $thiscolumn and $prevcolumn 121
    $thisoffset and $prevoffset 121
    $thisparser 121
    $thisrule and $thisprod 122
    $score 122
Introducing Startup Actions 122
Introducing Autoactions 124
Introducing Autotrees 125
Introducing Autostubbing 127
Introducing Directives 128
    <commit> and <uncommit> 129
    <reject> 130
    <skip> 131
    <resync> 132
    <error> 132
    <defer> 132
    <perl ....> 133
    <score> and <autoscore> 134
Precompiling the Parser 135
Summary 135

CHAPTER 6 Accessing Web Data with HTML::TreeBuilder 137

Introducing HTML Basics 137
Specifying Titles 138
Specifying Headings 139
Specifying Paragraphs 140
Specifying Lists 141
Embedding Links 142
Understanding the Nested Nature of HTML 143
Accessing Web Content with LWP 145
Using LWP::Simple 146
Using LWP 146
Using HTML::TreeBuilder 150
Controlling TreeBuilder Parser Attributes 152
Searching Through the Parse Tree 154
Understanding the Fair Use of Information Extraction Scripts 158
Summary 159

CHAPTER 7 Parsing XML Documents with XML::LibXML and XML::SAX 161

Understanding the Nature and Structure of XML Documents 163
The Document Prolog 164
Elements and the Document Body 166
Introducing Web Services 172
XML-RPC 173
RPC::XML 173
Simple Object Access Protocol (SOAP) 174
SOAP::Lite 175
Parsing with XML::LibXML 177
Using DOM to Parse XML 177
Parsing with XML::SAX::ParserFactory 179
Summary 182

CHAPTER 8 Introducing Miscellaneous Parsing Modules 185

Using Text::Balanced 185
Using extract_delimited 186
Using extract_bracketed 188
Using extract_codeblock 189

Using extract_quotelike 190
Using extract_variable 191
Using extract_multiple 192
Using Date::Parse 193
Using XML::RSS::Parser 194
Using Math::Expression 197
Summary 199

CHAPTER 9 Finding Solutions to Miscellaneous Parsing Problems 201

Parsing Command-Line Arguments 201
Parsing Configuration Files 204
Refining Searches 205
Formatting Output 212
Summary 214

CHAPTER 10 Performing Text and Data Mining 217

Introducing Data Mining Basics 218
Introducing Descriptive Modeling 219
Clustering 219
Summarization 220
Association Rules 221
Sequence Discovery 224
Introducing Predictive Modeling 224
Classification 225
Regression 225
Time Series Analysis 228
Prediction 229
Summary 241

INDEX 243

About the Author

CHRISTOPHER M. FRENZ is currently a bioinformaticist at New York Medical College. His research interests include applying artificial neural networks to protein engineering as well using molecular modeling techniques to determine the role that protein structures have on protein function. Frenz uses the Perl programming language to conduct much of his research. Additionally, he is the author of *Visual Basic and Visual Basic .NET for Scientists and Engineers* (Apress, 2002) as well as numerous scientific and computer articles. Frenz has more than ten years of programming experience and, in addition to Perl and VB, is also proficient in the Fortran and C++ languages. Frenz can be contacted at `cfrenz@gmail.com`.

About the Technical Reviewer

TEODOR ZLATANOV earned his master's degree in computer engineering from Boston University in 1999 and has been happily hacking ever since. He always wonders how it is possible to get paid for something as fun as programming, but tries not to make too much noise about it.

Zlatanov lives with his wife, 15-month-old daughter, and two dogs, Thor and Maple, in lovely Braintree, Massachusetts. He wants to thank his family for their support and for the inspiration they always provide.

Acknowledgments

Bringing this book from a set of ideas to the finished product that you see before you today would not have been possible without the help of others. Jason Gilmore was a great source of ideas for refining the content of the early chapters in this book, and Matthew Moodie provided equally insightful commentary for the later chapters and assisted in ensuring that the final page layouts of the book looked just right. I am also appreciative of Teodor Zlatanov's work as a technical reviewer, since he went beyond the role of simply finding technical inaccuracies and made many valuable suggestions that helped improve the clarity of the points made in the book. Beth Christmas also played a key role as the project manager for the entire process; without her friendly prompting, this book would probably still be in draft form. I would also like to express my appreciation of the work done by Kim Wimpsett and Laura Cheu, who did an excellent job preparing the manuscript and the page layouts, respectively, for publication. Last, but not least, I would like to thank my family for their support on this project, especially my wife, Thao, and son, Jonathan.

Introduction

Over the course of the past decade, we have all been witnesses to an explosion of information, in terms of both the amounts of knowledge that exists within the world and the availability of such information, with the proliferation of the World Wide Web being a prime example. Although these advancements of knowledge have undoubtedly been beneficial, they have also created new challenges in information retrieval, in information processing, and in the extraction of relevant information. This is in part due to a diversity of file formats as well as the proliferation of loosely structured formats, such as HTML. The solution to such information retrieval and extraction problems has been to develop specialized parsers to conduct these tasks. This book will address these tasks, starting with the most basic principles of data parsing.

The book will begin with an introduction to parsing basics using Perl's regular expression engine. Once these regex basic are mastered, the book will introduce the concept of generative grammars and the Chomsky hierarchy of grammars. Such grammars form the base set of rules that parsers will use to try to successfully parse content of interest, such as text or XML files. Once grammars are covered, the book proceeds to explain the two basic types of parsers—those that use a top-down approach and those that use a bottom-up approach to parsing. Coverage of these parser types is designed to facilitate the understanding of more powerful parsing modules such as Yapp (bottom-up) and `RecDescent` (top-down).

Once these powerful and flexible generalized parsing modules are covered, the book begins to delve into more specialized parsing modules such as parsing modules designed to work with HTML. Within Chapter 6, the book also provides an overview of the LWP modules, which facilitate access to documents posted on the Web. The parsing examples within this chapter will use the LWP modules to parse data that is directly accessed from the Web. Next the book examines the parsing of XML data, which is a markup language that is increasingly growing in popularity. The XML coverage also discusses SOAP and XML-RPC, which are two of the most popular methods for accessing remote XML-formatted data. The book then covers several smaller parsing modules, such as an RSS parser and a date/time parser, as well as some useful parsing tasks, such as the parsing of configuration files. Lastly, the book introduces data mining. *Data mining* provides a means for individuals to work with extracted data (as well as other types of data) so that the data can be used to learn more about a given area or to make predictions about future directions that area of interest may take. This content aims to demonstrate that although parsing is often a critical data extraction and retrieval task, it may just be a component of a larger data mining system.

This book examines all these problems from the perspective of the Perl programming language, which, since its inception in 1987, has always been heralded for its parsing and text processing capabilities. The book takes a practical approach to parsing and is rich in examples that are relevant to real-world parsing tasks. While covering all the basics of parser design to instill understanding in readers, the book highlights numerous CPAN modules that will allow programmers to produce working parser code in an efficient manner.

CHAPTER 1

Parsing and Regular Expression Basics

The dawn of a new age is upon us, an information age, in which an ever-increasing and seemingly endless stream of new information is continuously generated. Information discovery and knowledge advancements occur at such rates that an ever-growing number of specialties is appearing, and in many fields it is impossible even for experts to master everything there is to know. Anyone who has ever typed a query into an Internet search engine has been a firsthand witness to this information explosion. Even the most mundane terms will likely return hundreds, if not thousands, of hits. The sciences, especially in the areas of genomics and proteomics, are generating seemingly insurmountable mounds of data.

Yet, one must also consider that this generated data, while not easily accessible to all, is often put to use, resulting in the creation of new ideas to generate even more knowledge or in the creation of more efficient means of data generation. Although the old adage "knowledge is power" holds true, and almost no one will deny that the knowledge gained has been beneficial, the sheer volume of information has created quite a quandary. Finding information that is exactly relevant to your specific needs is often not a simple task. Take a minute to think about how many searches you performed in which all the hits returned were both useful and easily accessible (for example, were among the top matches, were valid links, and so on). More than likely, your search attempts did not run this smoothly, and you needed to either modify your query or buckle down and begin to dig for the resources of interest.

Thus, one of the pressing questions of our time has been how do we deal with all of this data so we can efficiently find the information that is currently of interest to us? The most obvious answer to this question has been to use the power of computers to store these giant catalogs of information (for example, databases) and to facilitate searches through this data. This line of reasoning has led to the birth of various fields of informatics (for example, bioinformatics, health informatics, business informatics, and so on). These fields are geared around the purpose of developing powerful methods for storing and retrieving data as well as analyzing it.

In this book, I will explain one of the most fundamental techniques required to perform this type of data extraction and analysis, the technique of *parsing*. To do this, I will show how to utilize the Perl programming language, which has a rich history as a powerful text processing language. Furthermore, Perl is already widely used in many fields of informatics, and many robust parsing tools are readily available for Perl programmers in the form of CPAN modules. In addition to examining the actual parsing methods themselves, I will also cover many of these modules.

Parsing and Lexing

Before I begin covering how you can use Perl to accomplish your parsing tasks, it is essential to have a clear understanding of exactly what parsing is and how you can utilize it. Therefore, I will define *parsing* as the action of splitting up a data set into smaller, more meaningful units and uncovering some form of meaningful structure from the sequence of these units. To understand this point, consider the structure of a tab-delimited data file. In this type of file, data is stored in columns, and a tab separates consecutive columns (see Figure 1-1).

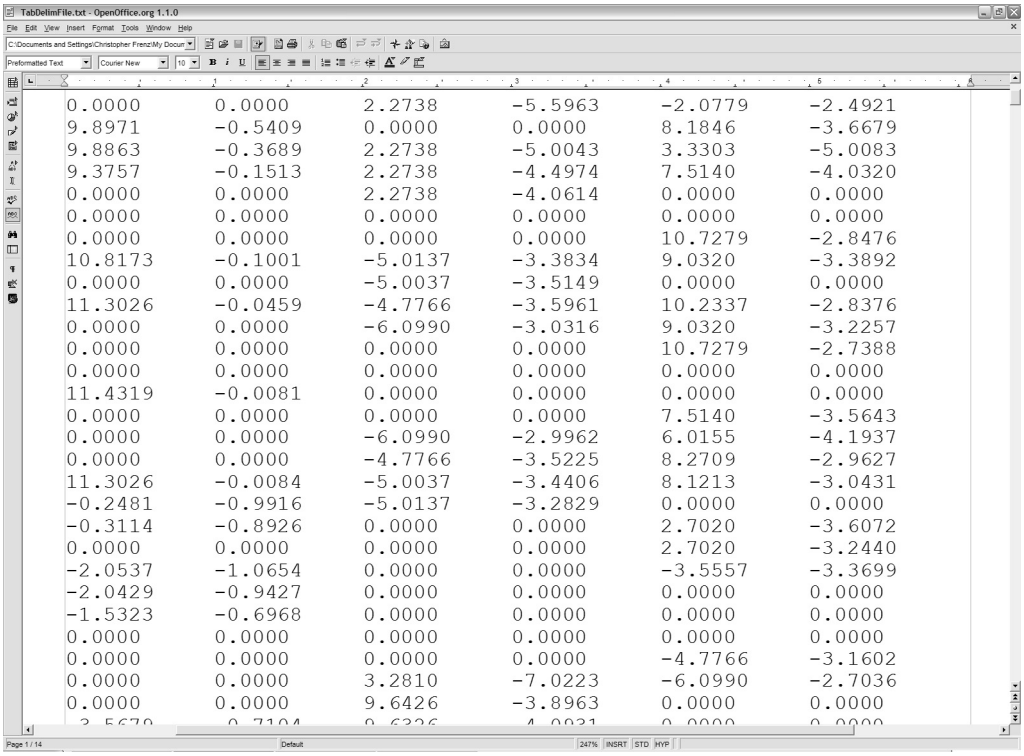

Figure 1-1. *A tab-delimited file*

Reviewing this file, your eyes most likely focus on the numbers in each column and ignore the whitespace found between the columns. In other words, your eyes perform a parsing task by allowing you to visualize distinct columns of data. Rather than just taking the whole data set as a unit, you are able to break up the data set into columns of numbers that are much more meaningful than a giant string of numbers and tabs. While this example is simplistic, we carry out parsing actions such as this every day. Whenever we see, read, or hear anything, our brains must parse the input in order to make some kind of logical sense out of it. This is why parsing is such a crucial technique for a computer programmer—there will often be a need to parse data sets and other forms of input so that applications can work with the information presented to them.

The following are common types of parsed data:

- Data TypeText files
- CSV files
- HTML
- XML
- RSS files
- Command-line arguments
- E-mail/Web page headers
- HTTP headers
- POP3 headers
- SMTP headers
- IMAP headers

To get a better idea of just how parsing works, you first need to consider that in order to parse data you must classify the data you are examining into units. These units are referred to as *tokens*, and their identification is called *lexing*. In Figure 1-1, the units are numbers, and a tab separates each unit; for many lexing tasks, such whitespace identification is adequate. However, for certain sophisticated parsing tasks, this breakdown may not be as straightforward. A recursive approach may also be warranted, since in more nested structures it becomes possible to find units within units. Math equations such as 4*(3+2) provide an ideal example of this. Within the parentheses, 3 and 2 behave as their own distinct units; however, when it comes time to multiply by 4, (3+2) can be considered as a single unit. In fact, it is in dealing with nested structures such as this example