

Batky-Howell's Advanced Perl Programming course offers many benefits for organizations that do a significant amount of Perl programming. Several chapters teach advanced productivity features of the Perl language itself, including debugging techniques, sophisticated list usage, code references, tied variables, and effective package use. The course puts programmers on very solid ground for doing object-oriented programming in Perl, going into many details beyond our introductory course.

Programmer productivity increases tremendously by reusing modules written by others, so our course teaches how to find, install, and use libraries of routines from the thousands of freely available Perl modules on the Web. We teach how to use SQL to access relational databases (such as Oracle) with the DBI/DBD Perl modules - most production Perl programs need to do this, especially web-based back-end server programs. We show how to build graphical interfaces in Perl using the Perl/Tk module. The course teaches several other productivity features as well, including extending Perl with C/C++, embedding the Perl interpreter in another language, documentation with POD directives, module development and distribution, and advanced Perl design and implementation considerations.

Course Objectives:

- Expertly manipulate lists, arrays, and hashes.
- Exploit code references and closures.
- Use eval to run dynamically generated Perl code, and to trap exceptions.
- Create and use object-oriented Perl modules.
- Tie Perl variables to subroutines to customize access and assignment.
- Access UNIX DBM files efficiently from Perl.
- Find and effectively use freely available Perl modules.
- Access database management systems from Perl programs using DBI.
- Write GUI application components quickly using Perl/Tk.
- Extend Perl with modules that load C/C++ object code.
- Interface Perl with, and embed Perl in, C/C++ applications.
- Describe the internal representation of Perl's various datatypes and data structures.
- Write reusable Perl modules.
- Avoid coding practices that hurt performance and maintenance.

Audience: Application programmers, system administrators, web-site authors, webmasters, and UNIX/Linux power users.

Prerequisites: *Perl Programming for UNIX* and Perl application development experience. Full comprehension of the extending and embedding material will require some C or C++ programming experience.

Number of Days: 4 days

1. Course Introduction Course Objectives Overview Suggested References	2. Debugging Warnings Diagnostic Messages
--	--

- Carping, Confessing, and Croaking
- Strict Checks
- Compiler Pragmas
- Debugging Flags
- Your Perl Configuration
- The Devel::Peek Module
- The Data::Dumper Module
- 3. Expert List Manipulation**
 - The grep Operator
 - Lists, Arrays, and List Operators
 - Context
 - Context and Subroutines
 - Initializing Arrays and Hashes
 - Reference Syntax
 - Auto-vivification
 - Defined Values
 - Other List Operators
 - Usage of map, grep, and foreach
- 4. Blocks and Code References**
 - Blocks
 - Subroutines
 - Subroutine Prototypes
 - Code Refs and Anonymous Subroutines
 - Typeglobbing for the Non-Squeamish
 - Local (Dynamic) Variables
 - Lexical Variables
 - Persistent Private Subroutine Variables
 - Closures
 - The eval Operator
 - The Block Form of eval
 - The String Form of eval
 - Block Form of eval for Exception Handling
- 5. Packages**
 - Review of Packages
 - BEGIN and END Blocks
 - Symbol Tables
 - Package Variables
 - Calling Package Subroutines
 - Importing Package Symbols
 - Exporting Package Symbols
 - Using the Exporter Package
 - The use Function
 - AUTOLOAD and @ISA
 - AutoLoader and SelfLoader
- 6. Objects and Classes**
 - Object-Oriented Stuff
 - Making Perl Object-Oriented
 - References
 - The bless Function
 - So, What's a Blessed Thing Good For?
 - Calling Class and Object Methods
 - Object Methods
 - Writing Classes
 - Constructors
 - Inheritance
 - What Perl Doesn't Do
- 7. Tied Variables**
 - Why Use tie?
 - Tying a Scalar
 - Inside Tied Variables
 - untie
 - Another Tied Scalar Example
 - Tying an Array
 - A Tied Array Example
 - Tying Hashes
 - Tie::Hash and Tie::Array
 - Tying Filehandles
 - What Are DBM, NDBM, GDBM, SDBM, etc?
 - Using the DBM Modules
- 8. Installing and Using Perl Modules**
 - Laziness, Impatience, and Hubris
 - CPAN
 - Using Modules
 - Installing a Perl Module
 - Unpacking the Module Source
 - The Configuration Step
 - The Build Step
 - The Test Step
 - The Install Step
 - Using CPAN.pm
 - Using Module Documentation
- 9. Introduction to DBI/DBD**
 - The Old Way - DBPerls
 - A Better Way - DBI/DBD
 - Database Programming
 - Handles
 - Connecting to the Database

- Creating a SQL Query
- Getting the Results
- Updating Database Data
- Transaction Management
- Finishing Up
- 10. DBI/DBD SQL Programming**
- Error Checking in DBI
- Getting Connected
- Drivers
- Using Parameterized Statements
- Statement Handle Attributes
- Other Handle Attributes
- Column Binding
- The do Method
- BLOBs and LONGs and Such
- Installing DBI Drivers
- 11. Introduction to Perl/Tk**
- Tcl, Tk, Tcl/Tk, Tkperl, Perl/Tk, etc.
- Perl/Tk
- Creating a Perl/Tk Application
- GUI Programming Overview
- Adding Widgets
- Scrolled Widgets
- Configuring Widgets
- Menus
- More Fun with Menus
- Using FileSelect
- 12. Perl/Tk Programming**
- Tk::Error and Tk::ErrorDialog
- Configuring Widgets
- Geometry Management
- Geometry Management with grid()
- The Frame Widget
- Defining Widget Callbacks
- Bindings
- Nonblocking I/O with fileevent()
- Tags
- Other Widgets
- Other Tk Commands
- Getting Tk
- 13. Extending Perl with C/C++**
- Extending the Perl Interpreter
- Overview of Perl5 XSUBs
- Get Started with h2xs
- Set up the Perl Wrapper Class
- Write the XS Code
- The XS File
- Write Some Test Code
- What Do You Want?
- Returning Values on the Stack
- A Walk Through an XSUB
- Arguments to XSUBs
- Other h2xs Options
- 14. Embedding the Perl Interpreter**
- Why Embed Perl?
- Embedding Perl in a C Program
- Compiling the Program
- perlmain.c
- Perl Data Types
- Macros and Functions
- Manipulating Scalars
- Memory Management
- Script Space
- Evaluating Perl Expressions
- Dynamic Loading
- Multiple Perl Interpreters
- 15. Module Development and Distribution**
- Distributing Modules
- Get Started with h2xs
- Files Created by h2xs
- The Build Library (blib)
 - Directory
- Unit Testing and test.pl
- Versions
- Using blib
- POD
- POD Translators
- Cutting a Distribution
- Other Niceties
- Makefile.PL
- 16. Design and Implementation**
- Think First
- Object-Oriented Design
- Object-Oriented Development
- Library Modules
- Utility Programs
- Filters
- Performance
- Timing with Benchmark