

UNIX[®] and Linux[®] Tools and Utilities: Hands-On - 4 Days

Course 396 Overview

- You Will Learn How To**
- Become an expert builder and user of UNIX/Linux tools and utilities
 - Perform complex search strings using regular expressions
 - Employ standard, programmable text filters to manipulate text and data
 - Build shell scripts to automate routine tasks
 - Achieve significant productivity gains by matching the mix of tools to the task at hand
 - Process structured data with **awk**
- Course Benefits** UNIX and Linux provide a rich set of utilities to enable developers to streamline tasks and maximize productivity. To realize the full power of your UNIX/Linux system, you need to choose the right tools and use them in the right combination. Through extensive in-class hands-on exercises, you gain the knowledge and skills to adapt the UNIX environment to your particular needs.
- Who Should Attend** Those who want to maximize the power of their UNIX/Linux system. Knowledge of UNIX or Linux at the level of Course 428, "UNIX Comprehensive Introduction," or Course 143, "Linux Comprehensive Introduction," is assumed.
- Hands-On Training** Extensive exercises applying UNIX tools, utilities and scripting applications using Red Hat[®] Enterprise Linux[®] are performed, including:
- Forming powerful regular expressions for searching text
 - Combining filters for sophisticated text processing
 - Performing complex text selection and manipulation with **awk**
 - Automating simple, repetitive tasks using shell scripts
 - Writing shell scripts to customize the behavior of standard UNIX tools

UNIX[®] and Linux[®] Tools and Utilities: Hands-On - 4 Days

Course 396 Outline

Basic UNIX and Linux Concepts

The evolution of UNIX

- How UNIX developed
- The current state of UNIX/Linux standards

Review of UNIX commands

- File and directory manipulation
- I/O redirection and pipes
- Writing shell start-up files
- Using the shell command history

Finding UNIX documentation

- The **man** command
- Other manual page browsers

Searching Text with Regular Expressions

UNIX regular expressions

- Specifying string patterns for filtering operations
- The meta character set
- Building search patterns
- Developing extended regular expressions

Using the **grep** command

- Processing files
- Processing command output

UNIX Text Filters

The characteristics of a UNIX filter

- Reading from standard input
- Writing to standard output and standard error
- Combining filters into pipelines to perform complex tasks
- Redirecting output of a pipeline

Common UNIX filters

- Editing the output of commands with the stream editor **sed**
- Translating characters with **tr**
- Sorting files and command output
- Comparing different versions of files with **diff**
- Using other common filters: **cut** and **uniq**
- Combining filters for complex text processing
- Executing filter commands with **find**
- Finding, comparing and searching files

Shell Programming

Shell basics

- Writing simple shell scripts

- Storing data in shell variables
- Exporting variables to the environment
- Preventing the creation of a subshell environment

Controlling logic flow

- Making decisions with **if** and **case**
- Quoting shell commands to control substitutions
- Testing file attributes, strings and numbers
- Reading and testing standard input
- Looping with **for** and **while**
- Accessing the shell's built-in variables

Other shell features

- Accepting command line arguments
- Redirecting standard output
- Substituting command output
- Performing arithmetic in shell scripts
- Scanning for command line options

Working with tools creatively

- Combining UNIX filters with pipelines and command substitution
- Developing scripts incrementally

Restructuring Data with **awk**

awk as a flexible search tool

- Testing and extracting fields from structured input
- Performing arithmetic calculations
- Writing useful **awk** one-liners

Creating long **awk** scripts

- Matching patterns with extended regular expressions
- Modifying **awk**'s default behavior with special patterns and built-in variables
- Calling **awk** built-in functions

Advanced **awk** capabilities

- Using **awk**'s control constructs for testing and looping
- Storing data in arrays
- Formatting output using **printf**
- Searching files with multiline records