

Shell Programming: Hands-On - 3 Days

Course 434 Overview

- You Will Learn How To**
- Write Bash and KornShell scripts for improved productivity
 - Integrate all key language features: arrays, functions, pattern matching, I/O, branches, loops and variables
 - Improve speed by performing multifile handling and string manipulations without external tools
 - Launch and control additional processes
 - Wrap external file and text-handling tools within scripts
 - Customise and extend the user environment login scripts
- Course Benefits** KornShell (**ksh**) and Bash have evolved into full-featured programming languages with efficient built-in modern constructs for superior string handling, decision making, arithmetic and post-processing. This hands-on course provides the skills you need to write reusable, robust shell scripts to extend the user environment and automate complex administrative tasks.
- Who Should Attend** Administrators, developers and other professionals using shell programming for improved productivity. 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** Throughout this course, you write a series of Bash or KornShell scripts that build in complexity as you master each new construct. Instructor-led exercises include:
- Creating loops and making decisions using **case**, **while** and **if**
 - Performing text processing tasks using **IFS** and **read**
 - Breaking a large program into functions
 - Handling errors with default values
 - Handling unexpected events with **trap**
 - Manipulating multiple files

Shell Programming: Hands-On - 3 Days

Course 434 Outline

Introduction and Overview

- Role of shell scripting
- Benefits of KornShell and bash vs. other shells
- Differences and similarities between **bash**, **ksh88** and **ksh93**
- Integrating scripts with external tools: **grep**, **sed**, **awk** and others
- Customising the login environment

Bash and Kornshell Scripting Fundamentals

Shell script elements

- Commands and comments
- Defining exit values

Conditional program execution

- Applying **if** and **case** statements
- Simplifying **if** logic with **elif**

Program loops and iteration

- Conditional looping with **while** and **until**
- Listing **for** loops

Testing files and directories

- Analysing attributes
- Checking file size and contents

Strings and patterns using [[]]

- Comparing strings
- Verifying the existence of a string
- Pattern matching and special characters

Debugging

- Redirecting standard error
- **set** commands for debugging

Storing and Accessing Data

Positional parameters

- Passing and accessing parameters
- Setting and unsetting parameters
- Manipulating parameters as groups

Shell variables

- Defining environment and local variables
- Specifying default values and error conditions

Arrays

- Creating and indexing arrays
- Processing array contents with special variables

Processing Data

Manipulating strings

- Extracting substrings
- Determining string length
- Find and replace

Mathematics

- Arithmetic **for** and **while** loops
- Writing mathematical expressions: (()), \$(()) and **let**

Modular Programming with Functions

Function basics

- Functions vs. scripts
- Parameters and variables

Creating a function library

- Finding your library with **PATH**
- **dot .** commands

Interacting with the Outside World

Manipulating files and redirecting data

- Scripting file and directory management
- Deciphering redirection order
- Unravelling the secrets of **exec**: opening and closing multiple files

Interacting with running processes

- Handling errors
- Defining post-termination actions such as notification, cleanup
- Handling and sending signals: **trap** and **kill**

Accessing network servers

- Connecting to a network server
- Exchanging data with a network server

Creating "Production Quality" Scripts

Ensuring environmental control

- Checking and modifying environment variables
- Using **getopts** to process command line options

Handling user interactions

- Setting up error processing
- Employing **select** to create a menu interface
- Processing keyboard input