

Shell programs, or scripts, are the means by which the Linux shell is used as a programming language. Linux commands and shell language control constructs are entered into a file by the programmer, then the file is executed as a command and interpreted just as if the commands had been typed on the shell command line. Thus, shell scripts provide a way to automate commonly executed groups of commands – but shell scripts can do much more than this. Although many simple tasks are automated with small scripts, large scripts hundreds of lines long are very common. These larger scripts are written by system administrators, database administrators, testers, utility programmers, and others to create utilities that are largely composed of powerful Linux commands, such as find, sed, awk, and hundreds of others.

In this course, students learn to read, write, and debug bash shell scripts. Back at work they can greatly increase productivity by automating repetitive tasks (for themselves or others), and by creating specifically tailored utilities designed to meet their precise needs. Students will read and write many bash scripts in this class, which will additionally increase their overall Linux knowledge and skills.

Course Objectives:

- Explain the purpose of shell programs.
- Recognize applications for shell programs.
- Design and write shell programs of moderate complexity, using variables, special variables, flow control mechanisms, operators, arithmetic, and functions.
- Debug shell programs using several different debugging techniques.
- Write “real-time” shell scripts that respond to and handle asynchronous events with the trap command.
- Manage multiple concurrent processes to achieve higher utilization of UNIX.

Audience: Linux or UNIX users, programmers, and system administrators.

Prerequisites: *Fundamentals of Linux* or *Fundamentals of UNIX*

Number of Days: 3 days

<p>1. Course Introduction Course Objectives Overview Suggested References</p>	<p>Running Scripts Specifying the Script’s Interpreter The PATH Environment Variable</p>
<p>2. UNIX Processes What is a Process? Process Structure The ps Utility Options to the ps Utility Background Commands (&) Killing Background Processes Redirecting the Standard Error</p>	<p>Sub-shells</p> <p>4. Variables Shell Variables The read Command The export Command The Shell Environment Parameter Expansion Command Substitution</p>
<p>3. Getting Started What is a Shell?</p>	<p>5. The Login Process</p>

- The Login Process
- The System Profile Script
- Your .bash_profile Script
- The . Command
- 6. Conditional Statements**
 - The Exit Status of Commands
 - Command Line Examples
 - The test Command
 - The if-then-else Construct
 - The elif Construct
 - case Statements
- 7. Loops**
 - The for Loop
 - The while Loop
 - break and continue
 - Reading Lines From Files
 - Using Arrays with Loops
- 8. Special Variables**
 - \$\$ - PID of Shell
 - Command-Line Arguments
 - \$# - Number of Arguments
 - * - All Arguments
 - The shift Command
 - The set Command
 - Getting Options
- 9. Quoting Mechanisms**
 - Single vs. Double Quotes
 - What is a Here Document?
 - Using a Here Document
 - Here Document Quoting
 - Ignoring Leading Tabs
- 10. Functions**
 - Shell Functions
 - Passing Arguments to Functions
 - Returning Values from Functions
 - Function Declarations
- 11. Advanced Programming**
 - Shell Arithmetic
 - The select Statement
 - Terminal Independence in Scripts
 - The eval Command
- 12. Debugging Techniques**
 - Using echo
 - Using Standard Error
 - Script Tracing
 - Options for Debugging