The Bash Shell

Tour, Variables, Control Flow, Functions, Builtins, History, Error Handling, Aliases, Best Practices

Bash is a powerful shell environment that provides excellent interactive and scripting control of the underlying OS, installed software platforms and can be used to build and manage your own custom tools too. Bash is good for repetitive chores, setting up an environment, automating non-trivial workflows, software builds and product installations.

The goal of well-written shell scripts is to allow repeatable / configurable / reliable task execution. The overall aim of this course is to equip attendees with a clear understanding of how to achieve that using Bash.

Professionals skilled in the art of Bash scripting are significantly more productive. Knowledge of Bash is a "must-have" tool in the skillset of every heterogeneous system administrator or developer. Though initially popular on Unix and Linux platforms, Bash is now very widely available, including on macOS and Microsoft itself has added it to modern Windows (10 & Server 2019, as an optional install). With a little effort you can even get Bash to work on mobile OSes. This wide availability is important because it means the effort you invest to learn Bash well now will pay repeated dividends in future regardless of which OS you use.

Target Audience

Administrators, developers and power users who wish to use the command line to interact with the system and to create shell scripts to automate such activities

Prerequisites

Some previous experience of shell usage required.

General knowledge of the operating system command layout is useful.

Contents of One-Day Training Course

Overview

What is Bash good for?
Compliance with POSIX Shell Standard
Feature tour
Setting up Bash – for Linux/macOS, Windows Subsystem For Linux and Git Bash (limited)
How Bash compares to PowerShell

Control Flow

if
select / case
for
break / continue

Variables

set and unset for local variables
env
Export to child processes

Processes and Command Line

Command pipeline
Command line arguments
Exit code and results
Process architecture of executing Bash
exec command
$$ - process id of process executing shell

Functions

Parameters to functions
Viewing declared functions
Nested functions
Job control

Configuring the Environment

.bashrc
Start up scripts - profiles
Login shell

Scripts

Creating/calling/debugging Bash scripts
Sourcing – external library of functions
Job control
History
Arranging larger blocks of script

Builtin Functions

Wide range of builtins
Regular expressions
readline

Alias

Listing aliases
Role of aliases
alias and unalias functions

Error handling

Error handling in scripts
Signal handling
trap

File Handling

How files are represented in Bash
general file I/O
umask
read

Scripting Best Practices

Small chunks of script
Pay attention to debugging
Resilience to error situations
Similar to/different from regular programming

Project

Explores the scripting needed for automated workflow for a non-trivial enterprise scenario