



Week 1 – Lesson 1: Unix / Linux Operating System Environment



Chapter Objectives

In this chapter, you will:

- Understand the Background history and purpose of Unix and Linux operating systems
- Understand classifications of software
- Understand the user roles (Where does IT person fit in?)
- Creating Simple Shell Scripts



UNIX and Linux Operating Systems

- UNIX operating system developed by AT&T in the 1960s.
- Many software vendors have developed their own version of UNIX.
- Linux operating system developed from a version called MINIX which has many programming languages, compilers, and system administration tools.
- Linux is a “Unix-Like” OS that can be run on many different types of computers including PCs. Linux is portable allowing it to run on Complex Instruction Set Computers (CISC) and Reduced Instruction Set Computer (RISC) processors.



UNIX and Linux Operating Systems

- Linux offers the benefits of multi-user capability, portability, multi-tasking, multiple processor support, and multiple modes of operation.
- Unix / Linux can use network protocols (such as [TCP/IP](#)) to allow clients to be connect to a server in order to share resources / or distribute processing of tasks (efficiency).
- Unix and Linux OS tend to support the practice of “[Open Source](#)” software development that allow for collaboration and more efficient and timely development of applications.



UNIX and Linux Operating Systems

- Linux supports multitasking - the ability of an operating system to handle multiple jobs at what appears to be the same time
- With pre-emptive multitasking, the operating system has the ability to take control of system resources from the application
- With cooperative multitasking, the application takes control of resources
- Linux supports use of Symmetrical Multiple Processors (SMP)



Modes of Operation

There are basically two modes of operation:

- **Graphical**

- Useful for day-to-day operation of tasks (user-friendly).

- **Text-Based**

- Allow system administrators and software developers to operate “behind the scenes” (sometimes quicker and more efficient, although can also edit files / develop software within graphical environment, for example, using a graphical text editor...)



Developing Applications / Programming

- When working in IT, you may be required to work “behind the scenes” such as setting up configuration files, etc...
- Another important skill you should have is the ability to automate routine tasks into some type of **program** or **script**.



Developing Applications / Programming

Software is divided into 2 general categories:

- **Systems Software**

- The Kernel
- Job Management software
- Memory Management software
- [Programming languages](#)
- Device drivers
- Software utilities

- **Application Software**

- Word Processing
- Spreadsheet
- Database



Developing Applications / Programming

Programming Languages are divided into 2 major categories:

- High-level
 - Designed to be easily read by humans
 - Not specific to the processor
 - Examples are Scripting Languages, C, C++, COBOL, JAVA

- Low-level
 - Specific to the processor
 - Example is the Assembler



Developing Applications / Programming

Programming Languages must be assembled,
compiled or interpreted...

- **Compiled Languages** (C, C+, Java, etc...)
 - Original program (source code) which is in human-readable form is translated into machine language which is run more efficiently...
- **Interpreted Languages** (Bash / Perl scripts, etc...)
 - Program consists of commands or instructions. Each line containing command or instruction is read and executed by a program (called in interpreter ...). Interpreted Languages do not run as efficiently as compiled programs, but are easier and usually quicker to create than compiled programs...



Understanding User Roles

Typical user roles in an OS environment are:

- **Standard User or User**
 - Use the operating system daily
- **System Administrator or root user**
 - Responsible for the complete computer operations
 - Upgrades applications, changes user passwords and monitors the system
 - May write shell script programs
- **Shell Programmer**
 - Responsible for writing shell script programs to automate routine tasks for System Administrator or Standard User.
 - Debugs shell script programs



Creating Shell Scripts

- The shell is an interpreted language
- Shell Scripts include operations such as:
 - Decision-making
 - Arithmetic operations
 - Use loops to create menus
 - Create functions to perform specific tasks
 - Manipulate data in an array
- Those above-mentioned operations make scripting just as relevant as any compiled language (just not as efficient if many people execute shell script simultaneously...)



Creating Shell Scripts

Here is an example of a simple shell script:

```
cat askAge.bash
```

```
# Start of Shell Script
```

```
# Prompt user for age and store result in a variable
```

```
echo -n "Please enter your age (in years): "
```

```
read age
```

```
# Print empty line, then print text using value
```

```
# of age stored in that variable...
```

```
echo
```

```
echo "You are $age years old"
```

```
# End of Shell Script
```

```
./askAge.bash
```

```
Please enter your age (in years): 44
```

```
You are 44 years old
```

Contents of Shell Script



Execution of Shell Script





Chapter Summary

- The operating system interacts with users, the applications, and the hardware.
- Linux operating system is a portable multi-user, multitasking, multiprocessing, and portable system. Linux supports X Windows (graphical) or text mode interface.
- Software can be classified into systems software and application software. Your role in this course is to create Bash Shell scripts (an interpreted language) to automate tasks for the System Administrators or Standard Users...