10.1.3 Process Facts

This lesson covers the following topics:

- Processor types
- Processor numbering
- Process states

Processor Types

A process refers to a program that is loaded into memory and is currently running on the system CPU. The following file types can create processes on a Linux system:

Type	Description
Binary executable	A binary executable is a program written in a programming language that is compiled into a binary file that the CPU can execute.
Internal shell commands	An internal shell command is a command built into the shell executable.
Shell scripts	A script is a text file that contains one or more commands. When the shell reads the file, it executes the commands as if they were entered through the keyboard.

Be aware of the following types of processes:

Type	Description
User	User processes start when a user executes a program file. For example, a user starts a user process when she executes the **grep** command.
System	System processes (also known as daemon processes) are started by the operating system, usually when it boots. However, users can also load daemon processes by manually starting a daemon using its init script or by using the **systemctl** command.

Processor Numbering

All processes are assigned unique process ID numbers randomly from a list of available numbers. Linux processes use the following identifiers:

Type	Description
Process ID (PID)	The PID uniquely identifies each process. System-started processes typically have low numbers, while user-started processes have higher numbers.
Parent Process ID (PPID)	The PPID identifies the process that spawned (or started) the current process. The process that spawned the new process is known as the parent process; whereas the spawned process is known as the child process.

Be aware of the following when working with processes:

- Either the init or the systemd process is the first process started by the kernel, depending upon the distribution. These processes:
  - Always have a PID of **1**.
  - Spawn all additional processes the operating system needs at boot time.
- *Forking* occurs when a parent process spawns a child process that is identical to the parent. An example of forking is the subshell that the shell creates when a user runs a command at the shell prompt. The command runs within the subshell. When the command completes, the subshell it ran within is destroyed.
- A *zombie* process is one where the process has finished executing and exited, but the process' parent didn't get notified that the child process was finished and hasn't released the child process' PID number. Zombie processes can linger in the system, consuming resources and PIDs. A zombie process may eventually clear up on its own. If it doesn't, you may need to manually kill the parent process.
- A single-threaded CPU can run only one process at a time.

Process States

As a process executes, it changes states according to its circumstances. These states can be viewed using the **ps aux** command and will be identified by one or more letters.

The most common state are:

Process State	Process	Description
Running	R	The process is either running (it is the current process in the system) or is waiting for the CPU to process it.
Stopped	T	The process has been stopped or suspended.
Waiting - Sleep	S	Interruptible sleep (waiting for an event to complete)
Waiting - Uninterruptible sleep	D	Uninterruptible sleep. This is a processes that cannot be killed or interrupted with a signal.
Zombie	Z	The process has finished executing and exited, but the process' parent didn't get notified that the child process was finished and hasn't released the child process' PID number.