Study Guide

Compia Linux + Exam XK0-004

Michael McClaren
Sept 11, 2019

Contents

Linux Boot Process Concepts

The boot process consists of the follwing stages - BIOS/UEFI POST - Boot Loader - Kernel initialization - Start the system initilization processes (sysv or systemd)

BIOS / UEFI

This is the Basic Input Output System, a firmware that is on the mainboard that is used to initialize the hardware on the system. The process is called the Power On Self Test (POST) and it ensures that all of the connected devices are initialized and responding before passing over control to the bootloader. Most recently, the BIOS system has been replaced by the Unified Extensible Firmware Interface that performs the same tasks as the BIOS, but it has additional capabilities.

Once the post is complete, the BIOS/UEFI system locates the bootloader and loads it into memory.

Boot Loader

This is the program that loads the operating system for the computer. It takes over once the POST has completed. On Linux, the most common bootloaders are GRUB (the GRand Unified Bootloader, also called legacy GRUB), and its updated replacement GRUB2, now simply referred to as GRUB. The bootloader is configured with the location of the desired operating system kernel which it loads into memory.

Kernel Initlization

The loaded Kernel will extract itself from a compressed image that is located in the `/boot` directory and then load the system initialization daemon. On older systems this was `sysv-init`, but on modern systems this is `systemd`. Once the kernel and the system initialization daemon are running, the system can start.

System Initialization

The system initialization daemon mounts the devices located in `fstab`, and proceeds to boot the system into the default run level. Once the process has completed the system is considered booted.

Boot Options

There are several ways in which the Kernel image can be loaded and each of these is a separate process. - Boot from ISO: This process uses a ISO image, that is mounted as a drive, to load the kernel. - PXE: The Pre-EXacutable environment is a client environment that searches for an appropriate server on the network from which to acquire a boot image. Once this image has been located, it is downloaded using the trivial file transfer protocol (tftp). - Boot from HTTP/HTPS: This process allows the image to be loaded using standard networking protocols.

Boot File Locations

Most configuration files for the boot process are in the `/boot` directory. Exceptions to this can be the GRUB files that can be located in different places, depending on the BIOS / UEFI specifications of the system. Most often they are found in one of the following:

- `/etc/grub/`
- `/etc/grub2.cfg`
- `/boot/grub` **or** `/boot/grub2`
- `/boot/efi`

Boot Modules and Files

In order to create the necessary files to enable us to boot, there are some commands that we need to used:

- `mkinitrd`**: This command will create the initial ramdisk is used by the kernel to preload block devices that are needed to access the** `root` **filesystem.**

- **`dracut`:Tthis command is similar to `mkinitrd` and results in an initial ramdisk the kernel can use to load block devices that are needed to access the `root` filesystem.**
- **`grub2-install`: This command is used to install the GRUB2 boot loader onto a device, which includes necessary images as well as creating the boot sector.**
- **`grub2-mkconfig`: This command is used for creating a configuratuion file for use by GRUB2**

In addition to the commands that are used to create the required resources to boot the system, there are some additional items you need to be familiar with:

- **`initramfs`: This is a complete set of `root` file system directories, bundled into a cpio archive and compressed.**
- **`efi` files: These files are used by the UEFI bootloader and comprise the efi partition. They are normally located at `/boot/efi/`.**
- **`vmlinuz`: This is the name of the Linux kernel executable. It is a compressed kernel that is capable of loading the operating system into memory (Virtual Memory LINUX gZip).**
- **`vmlinux`: This is a statically-linked executable file that contains the Linux kernel. It can be used in debugging. In contrast to vmlinu(z), this file is not compressed (Virtual Memory LINUX).**

Kernel Panic

This is a situation where the kernel discovers an unrecoverable error and it is not able to recover from that error without risk of data loss. This normally results in a bug check error being printed to the screen, followed by a memory dump prior to either waiting for a manual reboot or automatically rebooting once the memory dump is completed.

Installing, Managing, Configuring, and Monitoring Kernel Modules

The following commands are used to accomplish the tasks in this section:

- `lsmod`: **Lists the currently loaded modules.**
- `insmod`: **Inserts a module into the kernel**
- `modprobe`: **Loads or removes a loadable kernel module to or from the kernel**
- `modinfo`: **Extracts information about a kernel module that is provided to the command on the command line**
- `dmesg`: **Prints the message buffer of the kernel:**
  1. This output usually contains messages from device drivers and kernel modules.
- `rmmod`: **Removes modules from the kernel, but not modules that are in use**
- `depmod`: **Creates a list of module dependencies**

Modules are located in the following places on the file system:

- `/usr/lib/modules/[kernelversion]`: **This is location of the kernel modules specific to a release of the kernel**
- `/usr/lib/modules`: **This is the parent location for module storage.**
- `/etc/modprobe.conf`: **This is the file that contains the options that can be configured for modprobe. It is scheduled to be deprecated.**
- `/etc/modprobe.d/`: **This is the location of the modprobe configuration files. It is the replacement for** `/etc/modprobe.conf`**.**

Configuring and Verifying Network Connections Parameters

The following commands are used to accomplish the tasks in this section:

- `ping`: **Leverages the ICMP protocol to get an echo response from a host that is passed to it on the command line, either as a hostname or an IP address**
- `netstat`: **Used to print network connections, routing tables, and many other pieces of information about the network status**
- `nslookup`: **Queries the DNS system to resolve a domain name to an IP address**
- `dig`: **Short for Domain Information Groper, use for interrogating DNS name servers**
- `host`: **Simple tool for DNS lookups**
- `route`: **Manipulates the kernel's routing table, and can be used to set static routes**
- `ethtool`: **Used to configure and manipulate network device drivers and connections**
  1. It is mostly used for wired connections.
- `ss`: **Dumps socket statistics**
- `iwconfig`: **Configures and displays information about wireless network interfaces**
- `nmcli`: **Command line tool used for controlling NetworkManager and getting its status**
- `brctl`: **Manages ethernet bridges**
- `nmtui`: **Similar to mncli, and provides a text interface that shows options for the tool**

The following locations in the file system contain relevant configuration files:

- `/etc/sysconfig/network-scripts/`: **This is the location of the network interface configuration scripts get used to manipulate the network interfaces.**
- `/etc/sysconfig/network/`: **This is the location of more general network configurations that are not specific to an interface.**

- `/etc/hosts`: **This is the file that is the store for local DNS resolution.**
- `/etc/network`: **This is the location of the network configuration files.**
- `/etc/nsswitch.conf`: **This is the file that is used to determine the sources from which to obtain name-service information.**
- `/etc/resolv.conf`: **This is the file that contains the list of external DNS servers.**
- `/etc/netplan`: **This is the location of files that can be used to configure networking on newer systems.**
- `/etc/sysctl.conf`: **This file is used to override default kernel parameter values.**
- `/etc/dhcp/dhclient.conf`: **This file is used to configure the DHCP client.**

In addition, be familiar with the topic of Bonding, and these terms:

- `Aggregation`: **This is the process of combining several network connections in parallel, to increase network throughput.**
- `Active/Passive`: **This is a type of load balancing in which one side is the active side. The other side is passive and is held in reserve, in the event of the active side experiencing a failure.**
- `Load balancing`: **This is the process of distributing network traffic across multiple resources, to ensure that no single resource experiences resource exhaustion.**

Managing Storage in Linux

Basic Partitions

There are several types of partitions in Linux, These are different than file systems, and partitioning a device is simply dividing it. A partition can span the entire device, or be smaller pieces of it.

A raw device is a special type of logical device. It is associated with a character device file that allows it to be accessed directly, without using the operating systems buffers.

GPT: Short for GUID Partition Table, this type of partition table allows a device to be divided into a nearly unlimited number of partitions, depending on the operating system. GPT also allows for much larger drives, again limited by the operating system and the file system that will be used in the partition. GPT also stores multiple copies of its partition table in several locations on the device.

MBR - The Master Boot Record is a partition table that stores its partition data in a special boot sector, and has a 2TB limit in total drive size. MBR only supports 4 primary partitions.

File System Hierarchy

A real file system is a type of file system that exists on a device. It is physically mounted to the machine. A virtual file system exists in memory and does not actually physically exist on a device. Once the computer is powered off, the virtual file system no longer exists. One example of this would be `/proc`.

A relative path is one that is dependent on the directory that you are currently in, such as `./example`, where this means the object `example` in the current directory.

An absolute path is one that is not dependent on the current directory, such as `/home/user/example`. This is the absolute location of the file `example`.

Device Mapper

The Device Mapper is a framework in Linux that allows the mapping of physical devices to logical file volumes. It provides the following services(note that this is not an exhaustive list):

- **A logical volume is a volume that can be made up of multiple physical partitions. The Logical Volume Manager (LVM) is used to mange these Volumes.**
- **`mdadm` is a command line tool that allows the management of Redundant Array of Independant Disks (RAID) that are created using the device mapper.**
- **Multipath is a storage technology that lets us use more than one method of accessing storage devices. This allows for load balancing and high availability of the storage devices.**

File System Types

- **`ext3`: The 3rd extended file system was introduced in 2001 and it included journaling. It is limited by individual file size of 2TB and an overall system size of 32TB**
- **`ext4`: The 4th extended file system was introduced in 2008. It includes journaling too, but also allows for huge file sizes. Individual files can be up to 16TB and system size can be up to 1EB (exabyte).**
- **`xfs`: This file system was ported to Linux in 2014, it is a 64 bit, journaling file system that has excellent support for parallel I/O loads.**
- **`nfs`: The Network File System is a client/server file system that allows file access across networks as if they were local files.**
- **`smb`: Server Message Block is a network protocol allowing network access to files and other network resources.**
- **`cifs`: This is a version of `smb` and it stands for Common Internet File System.**
- **`ntfs`: This is a proprietary journaling file system developed by Microsoft, and is the default file system on modern versions of the Windows operating system.**

Important File System Locations

- **`/etc/fstab`: This is the location of the File System TABle, and it contains the information necessary to allow automatic mounting of devices.**
- **`/etc/crypttab`: This is the location of the information for encrypted devices that are set up during system boot.**
- **`/dev/`: This contains the special device files for all the devices on the system.**
- **`/dev/mapper`: This contains a listing of the Logical Volumes managed by LVM.**
- **`/dev/disk/by-`**
  1. `id`: This contains a mapping of the devices based on the serial number.
  2. `uuid`: This contains a mapping of the devices based on the UUID. This is how the devices are listed in `fstab` by default.
  3. `path`: This is a mapping of the devices based on the shortest physical path according to `sysfs` and contains the bus name (pci,ata, etc.).
  4. `multipath`: If this exists, it contains the path mappings for the device.
- **`/etc/mtab`: This contains a list of the currently mounted file systems according to the `mount` command.**
- **`/sys/block`: This contains symlinks to each of the block devices on the system.**
- **`/proc/partitions`: This contains the major and minor numbers of the partitioned devices.**
- **`/proc/mounts`: This is similar to `mtab` but it is maintained by the kernel.**

File system Commands

- **`mdadm`: Device mapper administration**
- **`fdisk`: CLI program for managing device partitions does not work with partitions larger than 2TB**
- **`parted`: CLI program for managing device partitions does not have the partition size limit of fdisk**
- **`mkfs`: MaKe File System, used to build a file system on a partitioned device (normally a disk drive)**
- **`iostat`: Reports CPU and device I/O stats**
- **`df`: Disk Free, reports the free space on the file system that is passed to it:**
  1. Using no arguments lists the system

- `du`: **Disk Usage, reports the size of the file that is passed to it:**
  1. With no arguments, it lists the file size of all files on the system each on one line.
- `mount`: **Attaches a file system to a mount point**
- `umount`: **Removes a file system attachment from a mount point**
- `lsblk`: **LiSt BLocK devices**
- `blkid`: **Locate and print block device attributes**
- `dumpe2fs`: **Prints the superblock block group info for a filesystem on a device**
- `resize2fs`: **Used for resizing a filesystem**
- `fsck`: **File System CheckK, used to detect errors on a filesystem and can be instructed to attempt to correct issues**
- `tune2fs`: **Allows adjustment of tunable file system parameters**
- `e2label`: **Displays or changes filesystem labels**

Cloud and Virtualization Concepts and Technologies

Templates

A VM template is what describes a virtual machine, and it is used to create the machine resources. There are several different types of templates, depending on how the virtual machine was created.

An OVA template is an archive if a machine that is a single file, made up of several other files that describe the virtual machine. This is in contrast to an OVF, which is made up of several files that comprise the virtual machine template. The files are contained inside of an OVA archive. OVA and OVF templates are seen in VMware virtualization. Virtual machines can also be described in Java Script Object Notation (JSON). This type of template is used in places such as AWS cloud formation.

Yet Another Markup Language (YAML) can also be used to describe a virtual machine as well as many other deployment types. A container image is different than a VM template. It contains the files that are used by the container, while a VM template describes the configuration.

Bootstrapping

The bootstrapping process refers to the methods that are used to instantiate a Virtual Machine. Machines that are created on cloud providers can use technologies such as **Cloud-Init**. This technique uses identical starting images, and then configures those images for their specific role.

Anaconda is the system installer used in RedHat Linux. The Anaconda configuration can be used to create identical machines via a process called **Kickstart**, which allows unattended installation.

Storage

In virtual machines storage, **Thick provisioning** is when the entire amount of storage is allocated when the storage is instantiated. **Thin provisioning** is where the limit of the disk is created but the actual storage is not allocated until it is used.

A **Persistent Volume** is one that is independent of a virtual machine, and exists even if the machine does not. These types of volumes are attached to machines, as opposed to ephemeral storage that only exists if the machine exists. **Blob** storage is Binary Large Object storage and differs from **Block** storage. It stores objects, and is not a block device like a traditional hard disk.

Block storage is tied to a virtual machine instance, and can contain a file system. **Blob** storage is normally remote storage and is accessed to retrieve objects but does not contain addressable blocks, or a file system that can be used for an operating system.

Networking

Virtual machines use the network of the host system. This can be accomplished in several ways.

Bridging is a method in which the host network is simply *bridged* to the virtual machines. The virtual machines then exist on the host network. An **overlay network** is an entire network segment that is created for the virtual machines independent of the host network. Ingress/egress methods must be created for network traffic to and from this overlay. **Network Address Translation (NAT)** is a technique in which internal IP addresses are mapped to an external IP address, making it appear as if all of the network traffic is coming from the same single IP on the host. A **local network** is one in which the virtual machines are only able to communicate with one another and their host system. **Dual-homed networks** provide high availability by incorporating two or more network interfaces. One is live and the others are hot standbys in case the primary fails.

Hypervisors

The hypervisor is the layer that exists between a virtual machine and its underlying host's system. This can be any one of several that are used, such as **KVM**, **Virtualbox**, **VMware**, etc. In all cases, a management tool us used to start and

stop the virtual machine,s as well as change settings and configurations. Some of the more common tools are **libvirt**, **virsh**, and **vmm**

Localization

Time is one of the most important components, as it affects every system operation. In addition, the language that the system used and the way that time is displayed should be considered. This is referred to as localization. The files used for this can be located in:

- `/etc/timezone`
- `/usr/share/zoneinfo`

Some commands that can be used to administer time settings are:

- `localectl`: **Used to change the keyboard layout and location settings**
- `timedatectl`: **Sets the system clock**
- `date`: **Displays the current time in a specific format**

There are also several environment variables related to localization, such as:

- `LC_*`: **Refers to all of the categories that exist for the Local such as time, messages etc.**
- `LC_ALL`: **Overrides all settings, normally used by applications to output in a known format**
- `LANG`: **Refers to the language that the system is using**
- `TZ`: **The time zone that is used for clock correction from UTC**

In addition to these settings there are character sets to consider, as all languages do not use the same character sets. Special characters, such as those used in Japanese, need to be displayed. Some available character sets include **UTF-8** **ASCII** and **Unicode**.

Operations and Maintenance

Software installation

Software can be provided for installation in packages. Packages come in several types:

- **.rpm : Used in RedHat based distributions**
- **.deb : Debian-based system package type**
- **.tar : Archive file that contains the software files**
- **.tgz : Compressed archive of the software files**
- **.gz : Another type of compression used to package the software files**

Each of the software packages that is used on a distribution has a related installation tool:

- `rpm`**: Installs .rpm packages**
- `dpkg`**: Installs .deb packages**
- `apt`**: Package manager for Debian-based distributions**
- `yum`**: Package manager for RedHat-based distributions**
- `dnf`**: Package manager for Fedora, and is a derivative of the YUM package manager**
- `zypper`**: Package manager for OpenSUSE**

Building software

Creating software requires some special commands such as the `make` and `make install` commands, which are used to build software from source. Before building software, it may be necessary to ensure that dependencies are met. Dependencies can be checked and listed using the `ldd` command.

In order to compile software from source a **compiler** is required. This converts the readable code into machine code. **Shared Libraries** are pieces of code that are used more than once, and the library is included to prevent repetition in the code.

A **repository** is a place where the code can be stored and accessed by development teams. The repositories can be created locally and then configured for remote access. As files are modified they can be synced to the repository. Commands such as **wget** or **curl** can download code to the local machine.

Managing Users and Groups

Users on the system are part of at least one group. Adding users is done with the `useradd` command.
Users can be added to groups using the `groupadd` command.
Once a user exists, it may be necessary to change properties of the user. The `usermod` command will do this. The equivalent group modification command is `groupmod`.
User passwords can be created using the `passwd` command, and password aging can be manged using the `chage` command.
User and group removal is done with the `userdel` or `groupdel` commands.
Users and groups can be assigned quotas for disk space and Inodes. Users can have custom bash profiles, and these are located in hidden files located in the user's home directory.
Global bash profile settings are located in the `/etc` directory.
User and group management files (`/etc/passwd` and `/etc/group`) list and configure the users and groups, respectively, in combination with the `/etc/shadow` file that contains encrypted users passwords.

Some commands that are used with users:

- **`id`: Show the users' IDs**
- **`whoami`: Show the current user**
- **`who`: Shows logged in users**
- **`w`: Shows detailed information about logged in users**
- **`last`: Shows historical user logins**

Working with Files

vi and **nano** are common text editors. But for just displaying and searching the contents of files, there are these commands:

- `grep`: **Prints lines matching a pattern**
- `cat`: **Prints the contents of the file**
- `tail`: **Prints the last lines of the file**
- `head`: **Prints the first lines of the file**
- `less`: **Reads the whole file and paginates the output**
- `more`: **Similar to** `less`

Output from commands can be redirected using meta characters. Text can also be processed using one of several programs available in most distributions, such as the stream editor (`sed`) or `awk`.
Manipulate files on the system by copying (`cp`) or moving (`mv`) the file.
Delete or remove files using the `rm` command.
Search for files in the system's file database using the `locate` command, once the database has been updated with `updatedb`.

Working with Services

The two service management systems that are the most common on Linux systems are the older `sysVinit` and the newer `systemd`.
The newer `systemd` uses *unit files* to describe services and manage them. If a service is **enabled** it will start automatically when the system starts. A **disabled** service is the opposite, and must be started manually. If a *unit file* is modified, it may be necessary to issue a `daemon-reload` so that `systemd` recognizes the changes.
`sysVinit` enabled and disabled services using the `chkconfig` directive either **on** (enabled) or **off** (disabled). `systemd` commands normally start with the `systemctl` directive.

Server Roles

Servers can be configured to provide a specific service or role. Some of the more common roles are:

- **NTP: Using the Network Time Protocol, these servers provide time services so that clocks can be synced across the network.**
- **WEB: These servers serve web pages.**
- **CA: A Certificate Authority, these servers provide certificate validation.**
- **Name Server: These provide DNS services on the network.**
- **DHCP: Using Dynamic Host Configuration Protocol, These servers issue IP addresses.**
- **File Servers: These provide network storage**
- **Monitoring: These servers are usually part of an agent server pair and provide real time performance information about monitored servers**
- **Authentication server: These provide centralized authentication and can be part of a single sign on (SSO) infrastructure.**
- **Database: These servers provide data services.**
- **Load balancer: These servers route traffic, based on load numbers, to servers that are behind them in the network.**

Devices

Some of the commands that can be used to manage devices:

- `lsdev`**: List installed hardware.**
- `lsusb`**: List used devices.**
- `lspci`**: List PCI connected devices.**
- `lsblk`**: List block devices (hard disks).**
- `dmesg`**: Examine the kernel ring buffer, which is normally the location of messages generated by device drivers.**
- `lpr`**: Used to print a file.**
- `lpq`**: Shows the print queue.**

Devices are mostly listed in the system's virtual file systems:

- `/proc`
- `/sys`
- `/dev`

Security

Permissions

User permissions come in three types **Read write** and **execute**.
In standard notation these are listed as **User group** and **other**.

Linux permissions listed as numbers, such as 755 or 644, are known as **octal notation** . Where read = 4 write = 2 and execute = 1 and the permissions are added together, 6 = read and write and 7 = read write execute.
The highest permission is 0777, which means everybody (all users and groups) can read, write, and execute. The default permissions are set using a **umask** which us subtracted from the highest permission. A umask of 0022 would result in a default permission of 0755 (0777 - 0022 = 0755).

If the **sticky** bit is set, only the owner of the file or directory, and `root`, can delete or rename the file. **Inheritance** is the process by which files and directories obtain their permission settings from their parent.

Commands used to manage permissions include:

- `chmod`: **Changes the permissions on a file or directory**
- `chown`: **Changes the owner of a file or directory**
- `chgrp`: **Changes the group ownership**
- `getfacl`: **Displays the File Access Control List (FACL)**
- `setfacl`: **Modifies the FACL**
- `ls`: **Lists files and directories**

Some operations cannot be executed by **standard** users. In these cases it may be necessary to use **privilege escalation** to complete these tasks as `root`. **Service** users are user accounts that are used to run services, and normally do not have login shells.
Users that can escalate privileges are listed in the `sudoers` file, this file is normally edited using the `visudo` command,

which checks the file syntax so permissions are not broken.
The command that is used to elevate permissions is either `su` (switch user) or `sudo` (super user do). Members of the `wheel` group are normally system administrators, and are allowed to elevate privileges to `root` by default.

Context-Based Permissions

Context-based permissions are based on extended attributes of objects.

SELinux

SELinux, or Security Enhanced Linux, is the default context-based permissions kernel module on RedHat-based distributions.

SELinux can be in one of three states:

- **Disabled: Permissions are not applied and are not logged.**
- **Permissive: Permissions are not applied, but permission violations are logged. This is useful for troubleshooting.**
- **Enforcing: Permissions are applied and violations are logged.**

The policy that is applied is one of two by default:

- **Targeted: Only objects that are listed in the targeted policy are evaluated. Unconfined objects are not evaluated.**
- **Strict: All objects are evaluated and confined objects are restricted.**

SELinux has specific tools that are used:

- `getenforce`**: Gets the current state of SELinux**
- `setenforce`**: Sets the state of SELinux**
- `sestatus`**: Lists the status of SELinux including the state and policy**

- `chcon`: **Changes the context for an object**
- `restorecon`: **Restores the context for an object to its default**
- `ls -Z`: **Lists the context for the objects in the directory passed**
- `ps -Z`: **Lists the context for the processes**

AppArmor

AppArmor is the default context-based permissions kernel module on Debian-based distributions.
These permissions are based on file paths of objects, some commands used here:

- `aa-disable`: **Disables an AppArmor profile**
- `aa-complain`: **Used for setting enforcement mode on a profile**
- `aa-unconfined`: **Lists network processes that do not have an AppArmor profile loaded**

Access and Authentication

On Linux, authentication is handled by Pluggable Authentication Modules (PAM). This provides an interface that applications can leverage, and creates a standardized method for authenticating users.
PAM is responsible for:

- **Password Policies: Complex passwords as well as password histories**
- **LDAP integration: Lightweight Directory Access Protocol, used for centralized authentication**
- **User lockouts: Failed logins resulting in lockout after a configured number of failures**

Modules in PAM can be **required** (the module must have a positive return), **optional** (the module is not required to have a positive result), or **sufficient** (the module is the only one necessary to have a positive result).

SSH, or secure shell, is a method of accessing a server and receiving a shell prompt. This access can be limited to only certain users (**User-specific access**) or limited to connections from certain hosts using **TCP Wrappers**.

Some files used with SSH:

`~/.ssh/`:

- **`known_hosts`: A list of trusted connections with the fingerprint for the server**
- **`authorized_keys`: A key store for the keys of users that are allowed to access the server with no password**
- **`config`: The local configuration for the SSH users**
- **`id_rsa`: The private SSH key**
- **`id_rsa.pub`: The public SSH key**

PKI is an infrastructure providing services that can be used to validate hosts. This consists of **private keys** and **public keys**, as well as **certificates** issued by **certificate authorities**. In addition, these services can be used to generate **Digital signatures**.

Security Best Practices

When considering system security, consider the following minimums:

- **If a password can be set, it should be, and it should be a complex password. This includes setting bootloaders and UEFI/BIOS passwords.**
- **If multi-factor authentication can be used, it should be.**
- **Disable `root` logins, and use the `sudo` method instead.**
- **Use keys, as opposed to passwords, where applicable.**
- **No users should share an ID.**
- **Services and users where applicable should use chroot jails.**
- **Maximize use of `deny.hosts`.**
- **Keep application data separate from operating system files.**
- **Change default service ports.**
- **Disable or Uninstall unused services.**
- **Enable and use TLS/SSH.**
- **Make sure that auditing is enabled.**
- **Make sure that necessary security patches are installed.**

- **Encrypt disks where applicable.**
- **Make sure that logging is enabled, and that logs are being rotated.**
- **Ensure that the firewall is properly configured, to provide the least access necessary for the operation of the system.**

Backups

Everything on Linux is a file, and as such it is possible to leverage file backup technology to create system backups. This includes `tar`, `cpio` and `dd`. Once a backup is created, to save space and transfer time to offsite storage, it makes sense to compress the archive using tools such as `gzip`, `xz`,`bzip2` or `zip`.

An **incremental** backup is one that contains the changes since the last incremental backup. A **full** backup contains the entire file set that was intended to be backed up. A **differential** backup contains the changes since the last full backup.

File hashing is the process of generating a hash value based on the file's contents, and can be used to validate that the backup file has not changed on disk.

Scripting

- **Environment is the workspace, and the shell is an instance.**
- **The children inherit from the parent Environment.**
- **Shell variables are set on each child independently.**
- **Shell variables are not passed to children.**

Variables

- **Are used to store data**
- **Can be changed**
- **Can be local or global**

Scripts

- **Used for automating repetitive tasks**
- **Used for configuring a task**
- **Can be on the command line (CLI) or in files (.sh).**
- **Need to be executable.**
- **Need to start with** `#!/bin/bash`

Comments should indicate what the script is intended to accomplish.

Variable declaration: The process of creating and assigning a value to a variable
Shell expansion: The process of extracting the value of a variable
File globbing: Pattern matching file names
Meta characters: Used for redirection of output, or passing output to the next command
Sourcing scripts: Including one script file's resources in another script

Positional parameters: Arguments passed to a script on the command line when the file is called. Escaping characters: The process of formatting lines so that bash can interpret it special charaters correctly

Conditional statements
These use Boolean logic to determine if the statements contained inside should be executed. Conditionals include `if`, `else`, and `then`.

Looping statements
These iterate over a set of statements until a condition is met. Looping statements include `while`, `until`, and `foreach`.

Source Code Management
In order to properly maintain it, source code needs to be checked into a source code management system. The native system for Linux is Git, which was created by Linux Torvalds.
Git repositories are created using the `git init` command, and can be copied using the `git clone` command. Once changes are set using `git commit`, the code can be pushed to a remote repository using the `git push` command. Git repositories can be checked out into **branches** so that code can be worked on, reviewed, and committed without disrupting the main body of code.

Central Configuration

Central Configuration is the process of managing servers via code. It is a system of automation that allows for provisioning, monitoring, and patching resources within an infrastructure. This could be virtualized infrastructure, or bare-metal.
An **Agent** system is one that uses software installed on target machines. The software typically reports back to a management server.

Conversely, a management system that does not require software agents to be installed on the managed systems is referred to as **Agentless**.

Puppet is a system that uses an agent, and Ansible is an example of an agentless system.

Inventory management is the process of recording the specifications of systems in the infrastructure. This is important as it can affect the commands that are used to manage the systems, as well as assisting with life cycles and patch management.

Infrastructure as Code is a way to describe the deployed resources so that they can be reproduced from that code. an example of this is a Kickstart file or a cloud-init file. This leverages **build automation** as well as **automated configuraiton management** to ensure that instantiated resources are tagged and configured correctly.