Basics

Check out **Launchpad**'s manual: https://documentation.ubuntu.com/launchpad/

nano, Linux's Text Editor: https://www.nano-editor.org/

NeoVim: https://neovim.io/

Vim - the ubiquitous text editor: https://www.vim.org/

GNU Emacs text editor: https://www.gnu.org/software/emacs/tour/

Check out **GIMP**, an open-source graphical editor: https://www.gimp.org/downloads/

script

In the case of script, you can call it like this:

script session.log

This will write the contents of your session to the session.log file. When you want to stop recording, you can write **exit** or press **Ctrl-D**. The generated file will be in ANSI format which includes the colors that were displayed on screen. In order to read them, you can use commands like ansi2txt or ansi2html to convert it to plain text or HTML respectively.

Linux Package Dependencies

The following is a list of terms used in this reading:

- **Debian**: One of many free Linux operating systems (OSes), used as the foundation for other OSes, like Ubuntu.
- **Linux packages**: A compressed software archive file that contains the files needed for a software application. These files can include binary executables, a software libraries, configuration files, package dependencies, command line utilities, and/or application(s) with a graphical user interface (GUI). A Linux package can also be an OS update. Linux OS installations normally come with thousands of packages. Common Linux package types include:
    - **.deb** - Debian packages
    - **.rpm** - Redhat packages
    - **.tgz** - TAR archive file
- **Linux repository**: Storage space on a remote server that hosts thousands of Linux packages. Repositories must be added to a Linux system in order for the system to search and download packages from the repository.
- **Stand alone package**: A package that does not require any dependencies. All files required to install and run the package on a Linux system are contained inside a single package.
- **Package dependency**: A package that other Linux packages depend upon to function properly. Often, packages do not include the dependencies required to install the software they contain. Instead, package manifests list the external dependencies needed by the package.
- **Package manager**: A tool on Linux systems used for installing, managing, and removing Linux packages. Package managers can also read package manifests to determine if any dependencies are needed. The package manager then finds and downloads the dependency packages before installing the packaged software. Several common Linux Package Managers include:
    - For Debian and Debian-based systems, like **Ubuntu**:
        - **dpkg** - Debian Package Manager
        - **APT** - Advanced Package Tool, uses dpkg commands
        - **aptitude** - user-friendly package manager
    - RedHat and RedHat-based systems, like **CentOS**:
        - **rpm** - RedHat Package Manager
        - **yum** - Yellowdog Updater Modified, comes with RedHat
        - **dnf** - Dandified Yum

The dpkg command

The **Linux dpkg** command is used to build, install, manage, and remove packages in Debian or Debian-based systems.

Syntax

The following are a few common dpkg command action parameters, with syntax and uses:

$ sudo dpkg - -install *packagename*	**To install a package**
$ sudo dpkg - -update-avail *packagename*	**To update a package saved locally**
$ sudo dpkg - -remove *packagename*	**To remove a package**
$ sudo dpkg - -purge *packagename*	**To purge a package, which removes the package and all associated files**
$ sudo dpkg - -list	**To get a list of packages installed**
$ sudo dpkg - -listfiles *packagename*	**To get a list of all files belonging to or associated with a package**
$ sudo dpkg -contents *packagename*	**To list the contents of a new package**

When an action parameter is added to the dpkg command, one of the following two commands are run in the background:

- **dpkg-deb**: A back-end tool for manipulating .deb files. The dpkg-deb tool provides information about .deb files, and can pack and unpack their contents.
- **dpkg-query**: A back-end tool for querying .deb files for information.

Additional Debian package managers

There are several alternate methods for managing Debian packages. Some have command-line interfaces (CLI) while others have GUIs. The alternative options to dpkg include:

- **APT (Advanced Package Tool)** - A powerful package manager designed to be a front-end for the dpkg command. APT installs and updates dependencies required for proper .deb package installation.
- **Synaptic Package Manager** – A popular GTK (GNU Image Manipulation Program ToolKit) widget with a GUI. Provides an array of package management features.
- **Ubuntu Software Center** – A GTK GUI developed by Ubuntu and integrated into the Ubuntu OS.
- **aptitude** – A user-friendly front-end for APT, with a menu-driven console and a CLI.
- **KPackage** – A part of KDE (Kool Desktop Environment) used to install and load packages that do not contain binary content. Non-binary content includes graphics and scripted extensions.

Linux Devices and Drivers

In this reading, you will learn how devices and drivers are managed in Linux. Previously, you learned that in Linux, devices attached to the computer are recognized by the operating system as device files. Devices are located in the /dev directory in Linux. A few examples of devices you may find in the /dev directory include:

- **/dev/sda** - First SCSI drive
- **/dev/sr0** - First optical disk drive
- **/dev/usb** - USB device
- **/dev/usbhid** - USB mouse
- **/dev/usb/lp0** - USB printer
- **/dev/null** - discard

Some of the Linux device categories include:

- **Block devices**: Devices that can hold data, such as hard drives, USB drives, and filesystems.
- **Character devices**: Devices that input or output data one character at a time, such as keyboards, monitors, and printers.
- **Pipe devices**: Similar to character devices. However, pipe devices send output to a process running on the Linux machine instead of a monitor or printer.
- **Socket devices**: Similar to pipe devices. However, socket devices help multiple processes communicate with each other.

Installing a device in Linux

There are hundreds of versions of Linux available due to the fact that Linux is an open source operating system. The methods for installing devices on Linux can vary from version to version. The instructions in this section provide various options for installing a printer and its device drivers on a Red Hat 9 Linux system running the GNOME user interface.

Device autodetect with udev

Udev is a device manager that automatically creates and removes device files in Linux when the associated devices are connected and disconnected. Udev has a daemon running in Linux that listens for kernel messages about devices connecting and disconnecting to the machine.

Installation through a user interface - GNOME

There are multiple user interfaces available for Linux. These instructions are specifically for the GNOME user interface.

1. In the GNOME user interface, open the **Settings** menu.
2. On the left-side menu, select **Printers**.
3. Click the **Unlock button** in the top right corner to change the system settings. Note that your user account must have *superuser*, *sudo,* or *printadmin* privileges to unlock the system settings for printers.
4. A dialog box will open showing a list of available printers. If your network has a large number of printers, you can search for the printer by IP address or host name.
5. Select the printer you want to install on the local system and click **Add**.
6. The printer listing will appear in the **Settings** window for the **Printers**.
7. In the top right corner of the printer listing, click the **Printer Settings** icon and select **Printer Details** from the pop-up menu.
8. The details of the printer will open in a new window. You should have three options for installing the printer driver:
    a. **Search for Drivers**: The GNOME Control Center will automatically search for the driver in driver repositories using PackageKit.
    b. **Select from Database**: Manually select a driver from any databases installed on the Linux system.
    c. **Install PPD File**: Manually select from a list of postscript printer description (PPD) files, which may be used as printer drivers.

Installation through the command line

Red Hat Linux uses the Common Unix Printing System (CUPS) to manage printers from the command line. CUPS servers broadcast to clients for automatic printer installation on Linux machines. However, for network environments with multiple printers, it may be preferable to manually install specific printers through the command line.

- From the command-line, enter **$ lpadmin -p printername -m driverfilename.ppd**
    - **Lpadmin** is the printer administrator command.
    - The **-p printername** command adds or modifies the named printer.
    - The **-m driverfilename.ppd** command installs the postscript printer description (PPD) driver filename that you provide. The file should be stored in the **/usr/share/cups/model/** directory.
    - Enter **$ man lpadmin** to open the manual for the lpadmin command to find additional command line options.

How to check if a device is installed
There are a couple of methods for checking if a device is already installed on a Linux machine:

Through a user interface like GNOME

1. In the GNOME user interface, open the **Settings** menu.
2. Browse each device set on the left-side menu.
3. The attached devices of the selected device type will appear in the window pane on the right.

Through the command line

Here are some commands that list specific device types:

- **$ ls /dev** - Lists all devices in the /dev folder
- **$ lcpci** - Lists devices installed on the PCI bus
- **$ lsusb** - Lists devices installed on the USB bus
- **$ lsscsi** - Lists SCSI devices, such as hard drives
- **$ lpstat -p** - Lists all printers and whether they are enabled
- **$ dmesg** - Lists devices recognized by the kernel

Linux kernel

The Linux kernel is the main component of a Linux operating system (OS). The kernel is software located in the memory that tells the central processing unit (CPU) what to do. The Linux kernel is like a personal assistant for the hardware that relays messages and requests from users to the hardware.

The kernel has four main jobs:

1. **Memory management** tracks how much memory is being used by what and where it is stored.
2. **Process management** determines which processes can use the central processing unit (CPU), when, and for how long.
3. **Device drivers** act as an interpreter between the hardware and processes.
4. **System calls and security** receives requests for service from the processes.

To ensure that Linux distribution is running the most current version of the operating system, you will need to update it regularly.

Updating Ubuntu Linux distribution

A Linux distribution is an operating system (OS) that includes the Linux kernel and usually a package management system. There are almost one thousand Linux distributions, and each distribution has a slightly different way of updating.

The Ubuntu distribution is one of the most popular since it is easy to use. There are two ways to update the Ubuntu distribution:

- **Update Manager** is a graphical user interface (GUI) that is nearly 100% automated. When updates are available, it will open on your desktop and prompt you to complete the updates. It checks for security updates daily and nonsecurity updates weekly. You can also choose to check for updates manually.
- **Apt** is the Ubuntu package management system that uses command line tools to update a Ubuntu distribution. Apt does not check for updates automatically, you must manually run it to check for updates. You can use the following commands to check for updates and upgrade:

1. `apt-get update` To update with apt, open the terminal and use the command `apt-get update`. This command prompts you to enter your password, then it updates the list of system packages.
2. `apt-get upgrade` Once the package list is up to date, use the command `apt-get upgrade` to actually download and install all updated versions for the packages in the list.

File system table (fstab)

File System Table (**fstab**) is a Linux configuration table. It helps to simplify mounting and unmounting file systems in Linux. Mounting means to connect a physical storage device (hard drives, CD/DVD drives, network shares) to a location, also called a mount point, in a file system table. In the past, IT Support specialists for Linux systems had to manually mount hard drives and file systems using the **mount** command. The **fstab** configuration file made this administrative task more efficient by offering the option to automate the mounting of partitions or file systems during the boot process. Additionally, **fstab** allows for customized rules for mounting individual file systems.

The fstab configuration table consists of six columns containing the following parameters:

Column 1 - Device:

- The universally unique identifier (UUID) or the name of the device to be mounted (sda1, sda2, … sda#).

Column 2 - Mount point:

- Names the directory location for mounting the device.

Column 3 - File system type:

- Linux file systems, such as ext2, ext3, ext4, JFS, JFS2, VFAT, NTFS, ReiserFS, UDF, swap, and more.

Column 4 - Options:

- List of mounting options in use, delimited by commas. See the next section titled “Fstab options” below for more information.

Column 5 - Backup operation or dump:

- This is an outdated method for making device or partition backups and command dumps. It should not be used. In the past, this column contained a binary code that signified:
  - 0 = turns off backups
  - 1 = turns on backups

Column 6 - File system check (fsck) order or Pass:

- The order in which the mounted device should be checked by the fsck utility:
  - 0 = fsck should not run a check on the file system.
  - 1 = mounted device is the root file system and should be checked by the fsck command first.
  - 2 = mounted device is a disk partition, which should be checked by fsck command after the root file system.

Example of an fstab table:

<File System>	<Mount Point>	<Type>	<Options>	<Dump>	<Pass>
/dev/sda1	/	ext3	nouser	0	1
/dev/sda2	swap	swap	defaults	0	0
/dav/hda1	/mnt/shared	nfs	rw, noexec	0	2

Fstab options

In Column 4 of the fstab table, the available options include:

- **sync or async** - Sets reading and writing to the file system to occur synchronously or asynchronously.
- **auto** - Automatically mounts the file system when booting.
- **noauto** - Prevents the file system from mounting automatically when booting.
- **dev** or **nodev** - Allows or prohibits the use of the device driver to mount the device.
- **exec** or **noexec** - Allows or prevents file system binaries from executing.
- **ro** - Mount file system as read-only.
- **rw** - Mount file system for read-write operations.
- **user** - Allows any user to mount the file system, but restricts which user can unmount the file system.
- **users** - Any user can mount the file system plus any user can unmount file system.
- **nouser** - The root user is the only role that can mount the file system (default setting).
- **defaults** - Use default settings, which include rw, suid, dev, exec, auto, nouser, async.

For more options, consult the man page for the file system in use.

Editing the fstab table

As an IT Support professional, you may need to expand the hard drive space on a server. Imagine that you have installed a new hard drive and the Linux server does not seem to recognize the drive. In the background, Linux has detected the new hardware, but it does not know how to display information about the drive. So, you will need to add an entry in the fstab table so that Linux will know how to mount it and display its entry within the file system. The following steps will guide you through this process:

1. Format the drive using the **fdisk** command. Select a Linux compatible file system, like ext4. If needed, you can also create a partition on the drive with the **fdisk** command.
2. Find which block devices the Linux system has assigned to the new drive. The block device is a storage device (hard drive, DVD drive, etc.) that is registered as a file in the /dev directory. The device file provides an interface between the system and the attached device for read-write processes. Use the **lsblk** command to find the list of block devices that are connected to the system.

Example output from the lsblk command:

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINT
sda	8:0	0	512G	0	disk	
└sda1	8:1	0	1G	0	part	/boot
sdb	8:16	0	1T	0	disk	
└sdb1	8:17	0	128G	0	part	

The seven columns in the output from the lsblk command are as follows:

a. **NAME** - Device names of the blocks. In this example, the device names are the existing sda drive and sda1 partition plus the new sdb hard drive and a newly formatted sdb1 partition.

b. **MAJ:MIN** - Major and minor code numbers for the device:

1. The **major** number is the driver type used for device communication. A few examples include:
   - **1** = RAM
   - **3** = IDE hard drive
   - **8** = SCSI hard drive
   - **9** = RAID metadisk
2. The **minor** number is an ID number used by the device driver for the major number type.
   - The minor numbers for the first hard drive can range from 0 to 15.
     a. The **0** minor number value for **sda** represents the physical drive.
     b. The **1** minor number value for **sda1** represents the first partition on the sda drive.
   - The minor numbers for the second hard drive can range from 16 to 31.
     a. The **16** minor number value for **sdb** represents the physical drive.
     b. The **17** minor number value for **sdb1** represents the first partition on the sdb drive.
   - ***Minor numbers for a third hard drive would range from 32 to 47, and so on.***

c. **RM** - Indicates if the device is:

1. **0** = not removable
2. **1** = removable

d. **SIZE** - The amount of storage available on the device.

e. **RO** - Indicates file permissions:

1. **0** = read-write
2. **1** = read-only

f. **TYPE** - Lists the type of device, such as:

1. **disk** = hard drive
2. **part** = disk partition

g. **MOUNTPOINT** - The location where the device is mounted. A blank entry in this column means it is not mounted.

3. Use an editor, like gedit, to open the fstab file:

Example fstab file:

Device	Mount Point	File System	Options	Dump	Pass
/dev/sda1	/	ext3	nouser	0	1

4. To add the new file system partition:

1. In the first column, add the new file system device name. In this example, the device name would be **/dev/sdb1**.
2. In the second column, indicate the mount point for the new partition. This should be a directory that would be easy to find and identify for users. For the sake of simplicity, the mount point for this example is **/mnt/mystorage**.
3. In the third column, enter the file system used on the new partition. In this example, the file system used for the new partition is **ext4**.
4. In the fourth column, enter any options you would like to use. The most common option is to select **default**.
5. In the fifth column, set the dump file to 0. ***Dump files are no longer configured*** in the fstab file, but the column still exists.
6. In the sixth column, the pass value should be **2** because it is not the root file system and it is a best practice to run a file system check on boot.

   Your fstab table should now include the new partition:

<File System>	<Mount Point>	<Type>	<Options>	<Dump>	<Pass>
/dev/sda1	/	ext3	nouser	0	1
/dev/sdb1	/mnt/mystorage	ext4	default	0	2

7. Reboot the computer and check the **mystorage** directory for the new partition.

Linux File System Repair

In this reading, you will learn how to use the file system consistency check or **fsck** command to repair data corruption in file systems on Linux machines. As an IT Support specialist, you will most likely encounter instances of data corruption in onsite systems. It is critical for you to know how to recover corrupted data, file systems, and hard drives.

A computer file system is software that provides structure for storing the operating system (OS) and all other software installed on system hard drives. A hard drive must be formatted with a file system before the operating system can be installed. Since Linux is an open source OS, innovators have created nearly 100 file systems that support Linux OS installations. Several common file systems that are used for Linux systems include ext, ext2, ext3, ext4, JFS, XFS, ZFS, F2FS, and more.

Like all software, software-based computer file systems can experience corruption. File system corruption can impede the computer's ability to locate files stored on the hard drive, including important OS files. File locations are stored as i-nodes (index nodes) in Linux. Every file in a Linux system has its own i-node identifier. The i-node stores metadata about the storage block and fragment location(s) where each file is stored. The i-node metadata also holds information about the file type, size of the files, file permissions, links to the file, and more.

Symptoms of data corruption

Symptoms of data corruption can include:

- System suddenly shuts down
- Software program will not launch or it crashes when opening a corrupted file. May also give an error message saying:
    - "File format not recognized" *or*
    - "(file name) is not recognized"
- Corrupted files and folders may no longer appear in the file system.
- The operating system (OS) may report bad sectors when failing to execute commands.
- Damaged platter-based hard drives can make clicking sounds or unusual vibrations.

Causes of data corruption

Data corruption on system hard drives and file systems can be caused by:

- **Software errors** -
    - Software errors can be any software event that interferes with normal hard disk read/write operations.
    - Viruses and malware can be designed to intentionally cause corruption to data.
    - Antivirus software can damage files if the software experiences problems while scanning or repairing the files.
- **Hardware malfunctions** -
    - Larger files are more likely to experience corruption than smaller files. Large files occupy more disk space, making them statistically more likely to cross a bad sector on the hard drive.
    - Hard drives that contain platters are at risk of experiencing malfunctioning read/write heads. Damaged heads can corrupt multiple files and directories in a single read/write transaction. Hard drives with moving mechanical parts are more likely to experience failures from moving parts that wear out over time.
- **Electrical damage** - Can happen when a power failure occurs while the system is writing data to a hard drive.

Data corruption repair

The most critical first step, after data corruption has been identified or suspected, is to shut down the affected hard drive(s). The reason for this step is to stop the cause of the corruption from writing to the hard drives. The longer the corruption activity continues, the more difficult recovering the data becomes.

Precautions should be taken before powering up a corrupted hard drive to run repair tools. It is important to minimize any read/write operations on the disk other than those produced by data recovery tools. One method to prevent further damage could be to have a corrupted Linux system boot from an external device or network (PXE boot). An alternative method might be to attach the corrupted hard drive as an external hard drive to a healthy system running Linux. A hard drive adapter or drive docking station can be used to convert an internal drive into an external device.

Before connecting a corrupted drive to a healthy system, the automount service must be disabled. The fsck command will not repair corruption on a mounted file system. In fact, mounting a corrupted file system can cause the healthy Linux system to crash. Although the corrupted file system should not be mounted, the device file for the corrupted hard drive in the /dev directory must be readable for the fsck command to access the drive.

– – Continued – –

The fsck command

Important Warning: The fsck command should NOT be used:

- ***on a hard drive that was a member of a RAID array.***
- ***on a mounted file system (must be unmounted).***

An important command line data recovery tool offered in the Linux operating system is the **fsck** command. It should be run anytime a Linux system malfunctions. The **fsck** command can check the file system and repair some, but not all, inconsistencies found. In some cases, **fsck** may recommend deleting a corrupted file or directory. The default setting for the **fsck** command is to prompt the user to approve or deny the repair of any problems found. The user running the **fsck** command must have write permissions for the corrupted file or directory to be able to approve a repair. If the user does not choose to repair inconsistencies found, the file system will remain in a corrupted state.
The **fsck** command will check for inconsistencies and prompt the user to make decisions on whether or **fsck** should repair for the following problems:

- Block count is not correct
- Blocks and/or fragments that are:
  - allocated to multiple files
  - illegally allocated
- Block and/or fragment numbers listed in i-node metadata that are:
  - overlapping
  - out of range
  - marked free in the disk map
  - corrupted
- Map inconsistencies on the disk or in the i-node.
- Directory:
  - contains references to a file but that number does not equal the number of links listed in the same file's i-node metadata.
  - sizes are not multiples of 512

The following checks are not run on compressed file systems.

- Directory checks:
  - Directories or files that cannot be located or read.
  - The i-node map has an i-node number marked as being free in the directory entry.
  - The i-node number in the metadata is out of range.
  - The . (current directory) or .. (parent directory) link is missing or does not point to itself.
- Fragments found in files larger than 32KB.
- Any fragments that are not listed as the last address of the file in an i-node metadata file.

How to use the fsck command

1. Enter **fsck** as a command line instruction. Syntax:

fsck [-n] [-p] [-y] [-f] [*FileSystem1name - FileSystem2name ...*]

- The -n flag - Sends a "no" response to all fsck questions and does not allow fsck to write to the drive.
- The -p flag - Prevents error messages for minor problems from displaying while automatically fixing those minor errors. Outside of recovering from data corruption, it is a best practice to run the fsck -p command regularly at startup as a preventative measure.
- The -y flag - Sends a "yes" response to all fsck questions to automatically attempt to repair all inconsistencies found. This flag should be reserved for severely corrupt file systems only.
- The -f flag - Runs a fast check that excludes file systems that were successfully unmounted for shutdown before the system crashed.
- *FileSystem#name* - If you do not specify a file system, the fsck command checks all file systems in /etc/filesystems, where the check attribute is set to true.
- To see more advanced flags, use the man fsck command.

a. To have the fsck command check all of the default file systems and prompt the user on how to handle each inconsistency found, simply enter at a command line:

```
$ fsck
```

b. For ext, ext2, ext3, and ext4 file systems, the e2fsck command can be used:

```
$ e2fsck
```

c. To have the fsck command check specific file system(s) and automatically fix any inconsistencies found, enter:

```
$ fsck -p myfilesystem
```

2. The fsck command outputs an exit value, or code, when the tool terminates. The code is the sum of one or more of the following conditions:

- 0 = All scanned file systems have been restored to a functional state.
- 2 = fsck did not finish checks or repairs due to an interruption.
- 4 = File system has changed and the computer needs to be rebooted.
- 8 = fsck could not repair some or all file system damage.

How to run fsck on the next boot or reboot

In many Linux OS distributions, the **fsck** utility will automatically run at boot under certain circumstances, including:

- When a file system has been labeled as “dirty”, meaning that data scheduled to be written to the file system is different from what was actually written or not written to the disk. This could occur if the system was shut down during a write operation.
- When a file system has been mounted multiple times (can be set to a specific value) without a file system check.

Configuring the **fsck** command to run automatically on boot and reboot differs depending on which brand and version of Linux is installed on the system. As a root or sudo user, use **vi** (visual instrument) to add the fsck command to the boot sequence.

1. In Debian and Ubuntu: THESE INSTRUCTIONS ARE INCORRECT
    a. Edit the **rcS** file: **$ sudo vi /etc/default/rcS**
    b. Add the following command to the rcS file: **FSCKFIX=yes**
2. In CentOS:
    a. Create or edit a file named **autofsck**: $ sudo vi /etc/sysconfig/autofsck
    b. Add the following command to the **autofsck** file: AUTOFSCK_DEF_CHECK=yes

Resource Monitoring in Linux

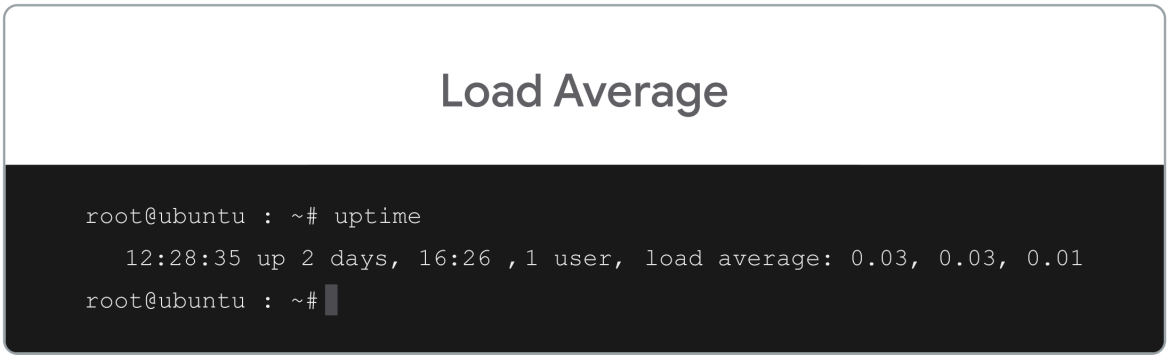
Balancing resources keeps a computer system running smoothly. When processes are using too many resources, operating problems may occur. To avoid problems from the overuse of resources, you should monitor the usage of resources. Monitoring resources and adjusting the balance is important to keep computers running at their best. This reading will cover how to monitor resources in Linux using the load average metric and the common command.

Load in Linux

In Linux, a `load` is the set of processes that a central processing unit (CPU) is currently running or waiting to run. A load for a system that is idle with no processes running or waiting to run is classified as a 0. Every process running or waiting to run adds a value of 1 to the load. This means if you have 3 applications running and 2 on the waitlist, the load is 5. The higher the load, the more resources are being used, and the more the load should be monitored to keep the system running smoothly.

Load average in Linux

The load as a measurement doesn't provide much information as it constantly changes as processes run. To account for this, an average is used to measure the load on the system. The load average is calculated by finding the load over a given period of time. Linux uses three decimal values to show the load over time instead of the percent other systems use. An easy way to check the load average is to run the **`uptime`** command in the terminal. The following image depicts the load values returned from the **`uptime`** command.



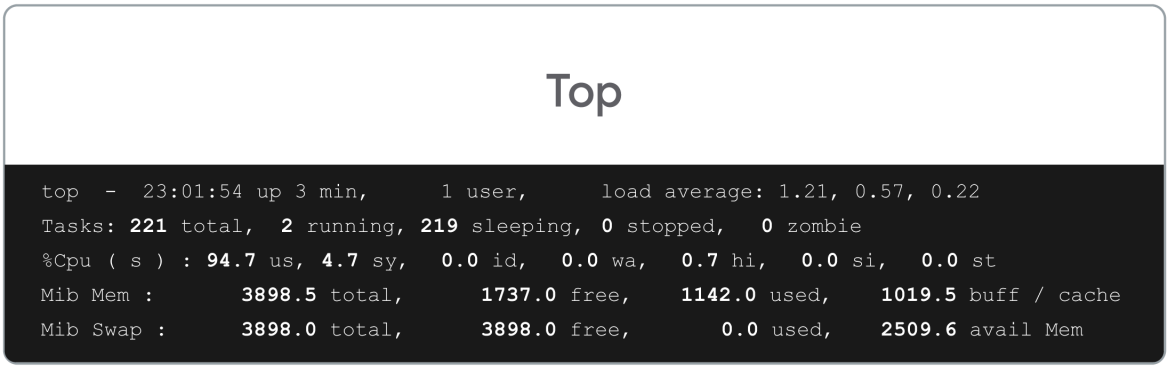
The command returns three load averages:

1. **Average CPU load for last minute**, which corresponds to 0.03. This is a very low value and means an average of 3% of the CPU was used over the last minute.
2. **Average CPU load for last 5 minutes** corresponds to the second value of 0.03. Again, this can be thought of as, on average, 3% of the CPU was being used over the past five minutes.
3. **Average CPU load for last 15 minutes** corresponds to 0.01, meaning on average, 1% of the CPU has been used over the last 15 minutes.

– – Continued – –

Top

Another way you can monitor the load average in Linux is to use the **top** (table of processes) command in the terminal. The result of running the top command is an in-depth view of the resources being used on your system.



The first line displayed is the same as the load average output given using the uptime command It lists what percent of the CPU is running processes or has processes waiting. The second line shows the task output and describes the status of processes in the system. The five states in the task output represent:

1. **Total** shows the sum of the processes from any state.
2. **Running** shows the number of processes currently handling requests, executing normally, and having CPU access.
3. **Sleeping** shows the number of processes awaiting resources in their normal state.
4. **Stopped** shows the number of processes ending and releasing resources. The stopped processes send a termination message to the parent process. The process created by the kernel in Linux is known as the “Parent Process.” All the processes derived from the **parent process** are termed as “Child Processes.”
5. **Zombie** shows the number of processes waiting for its parent process to release resources. Zombie processes usually mean an application or service didn't exit gracefully. Having a few zombie processes is not a problem.

The top command gives detailed insight on usage for an IT individual to gauge the availability of resources on a system.

Key Takeaways

Computers need to balance the resources used with the resources that are free. Ensuring that the CPU is not overused means that a system will run with few issues.

- The load in Linux is calculated by adding 1 for each process that is running or waiting to run.
- Monitoring the average load of Linux allows an IT professional to identify which processes are running to determine what to end in order to balance the system. A balanced system runs with fewer problems than one that is using too high of a percent of resources.
- The load average uses three time lengths to determine the use of the CPU: one minute, five minutes and fifteen minutes.

The `top` command can give detailed information about the resource usage of tasks that are running or waiting to run.

Supplemental Reading for OS Deployment Methods

Disk cloning software

Hard drives can also be cloned using software. This method allows the original and target to be different media from one another. For example, a hard drive can be cloned from an IDE drive to an SSD drive, a CD-ROM/DVD, removable USB drive, cloud-based systems, or other storage media, and vice versa. Software cloning supports full disk copies (including the OS, all settings, software, and data) or copies of selected partitions of the drive (useful for data-only or OS-only copies). Disk cloning software is often used by IT Administrators who need to deploy disk images across a network to target workstations or to cloud-based systems. Cloud platforms normally offer a virtual machine (VM) cloning tool as part of their services. **VM cloning is the most efficient method for cloning servers and workstations. VM cloning takes a few seconds to deploy new systems.**

A few examples of disk cloning software include:

- **NinjaOne Backup** - Cloud-based cloning, backup, and data recovery service designed for managed service providers (MSPs) and remote workplaces.
- **Acronis Cyber Protect Home Office** - Desktop and mobile device cloning software that works with Windows, Apple, and Android systems. Designed for end users. Supports backup, recovery, data migration, and disk replication. Includes an anti-malware service that can overcome ransomware attacks.
- **Barracuda Intronis Backup** - Cloud-based cloning and backup service on a SaaS platform. Designed for MSPs who support small to mid-sized businesses. Can integrate with professional services automation (PSA) and remote monitoring and management (RMM) packages.
- **ManageEngine OS Deployer** - Software for replications, migrations, standardizing system configurations, security, and more. Can create images of Windows, macOS, and Linux operating systems with all drivers, system configurations, and user profiles. These images can be saved to a locally stored library. The library is available to deploy OSs to new, migrated, or recovered systems as needed.
- **EaseUS Todo Backup** - Free Windows-compatible software for differential, incremental, and full backups, as well as disaster recovery. Supports copying from NAS, RAID, and USB drives.

Methods for deploying disk clones:

Flash drive distribution

OSs can be distributed on flash drives. IT professionals can format flash drives to be bootable prior to copying a cloned disk image to the flash drive. Target systems should be set to boot from removable media in the BIOS. After inserting a flash drive containing the OS into an individual computer, restart the system and follow the prompts to install the OS. Microsoft offers this method as an option for Windows installations. Linux systems can also be booted and installed from flash drives.

The Linux dd command

The Linux/Unix dd command is a built-in utility for converting and copying files. On Linux/Unix-based OSs, most items are treated as files, including block (storage) devices. This characteristic makes it possible for the dd command to clone and wipe disks.

For more information on disk cloning and OS deployment techniques, please visit:

- How to clone a hard drive on Windows - Step-by-step guide with screenshots on how to clone a hard drive using the software Macrium Reflect Free.
- Best Hard Drive Duplicator/Cloner Docking Station for 2022 - Comparison guide to popular hard drive duplicator machines.
- OS deployment methods with Configuration Manager - Microsoft's guide to options for deploying Windows in a network environment.
- dd(1) - Linux manual page - The manual for the Linux dd command, which describes how to use the command and lists the available optional flags.

Prompts

Help:
$ - -help } Flag that displays useful information about using a command
$ man ls } Displays the manual pages for a command

Archiving/Un-Archiving Files using 7z and tar command.
$ 7z e *filename.tar* - Extracts the contents of a tar archive.
For more information on tar, check out the link below:
https://www.ibm.com/docs/en/zvm/7.4.0?topic=osc-tar-manipulate-tar-archive-files-copy-back-up-file

Check what drives are mounted:
$ lsblk

Open fdisk in interactive mode:
$ sudo fdisk /dev/[drive]

Check the most recent logs:
$ sudo tail -f /var/log/syslog

Show the top 10 largest files, starting from the **/home** directory:
```
sudo du -a /home | sort -n -r | head -n 10
```

Installing Remote Access on a computer you want to control
$ sudo apt-get install openssh-client

Installing Remote Access on a Server
$ sudo apt-get install openssh-server

Go back to the client to test with this:
$ ssh dave@[IP Address]
dave's password: