FUNDAMENTALS OF OPEN SOURCE IT AND CLOUD COMPUTING (LFS200)

horizontal line

**First version: FEBRUARY 19, 2023 / 10:37 PM CT**[**https://trainingportal.linuxfoundation.org/learn/course/fundamentals-of-open-source-it-and-cloud-computing-lfs200**](https://trainingportal.linuxfoundation.org/learn/course/fundamentals-of-open-source-it-and-cloud-computing-lfs200) **00 The Operating System**

1. An operating system is "software that controls the operation of a computer and directs the processing of programs".
2. The heart of an operating system is its kernel. This is sometimes a very big program, but still a program. In fact, the word Linux technically only refers to the kernel, and not the whole operating system. The kernel has privileged features such as the ability to directly control devices connected to it. Devices such as a mouse, keyboard, screen, memory, storage, and many, many more are controlled by the kernel.
3. If the outside world would like to access these devices controlled by the Linux kernel, there must be an interface to interact with it; this interface is in fact another program, a program that can communicate with the kernel to cause some action in the hardware to occur (e.g., actions like light a bulb, read a key button being depressed, etc).
4. We need an interface between the kernel and the user program. The system call is the fundamental interface between an application and the Linux kernel. This interface between the user program and the Linux kernel is [the system call (or syscall)](https://man7.org/linux/man-pages/man2/syscalls.2.html), which is privileged code that can make requests to the Linux kernel. The kernel accepts the properly formatted request, performs the requested action and issues a return code indicating access or failure.

**01 Tasks Performed by an Operating System**

1. **Processes** - A process is any actively running instance of a program. On computers with multi-core processors, Linux can manage how to divide the program so that it runs on more than one processor at a time.
2. **Memory Management** - Linux manages the sharing of internal memory among processes. Allocating and deallocating addresses in RAM is an important aspect of memory management and will happen thousands of times per second.
3. **File Management** - Linux ensures there is sufficient available memory space to load files from disk to memory and back, enforcing the permissions stored with the data.
4. **Input/Output** - The Linux kernel handles requests to the hardware like mouse, keyboard, monitor, etc., and ensures the appropriate device driver is loaded for the device.
5. **Interpreting user commands** - When a user presses a keyboard key, touches a touchscreen or uses a mouse, the operating system will make sure that the machine responds as the user expects.
6. **Resource allocation** - The term is used to describe the combination of process and memory management to ensure that applications have enough processor time and can be allocated sufficient space in RAM as necessary.

**02 What is Linux?**

Linux is a free, open source operating system which has now grown to run on a large number of physical computer architectures. While Linux was originally based on the UNIX operating system, it was written from scratch by Linus Torvalds with assistance from a loosely-knit team of developers across the Net. It aims towards POSIX and Single UNIX Specification compliance where possible. Linux is distributed under the [GNU General Public License, version 2 (GPLv2)](https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html).

**03 What are some of the reasons to use Linux?**

1. Free
2. Stable
3. Secure
4. Maintainability
5. Runs on wide range of hardware
6. Easy to use
7. Support
8. Open Source

**04 Devices and Drivers**

Device drivers control many different types of hardware. Standardization has helped define how hardware is assembled into “main boards” or “motherboards”, where additional components can be connected. There are several “bus architectures” used to connect devices in a manner that the CPU can read, write and control the devices. Almost every bus has a selection method, an address mechanism and a data stream.

**05 Simple Bus Architecture**

A typical motherboard will have a collection of connections and buses depending on the motherboard form factor. There are decades of development and change in the bus architecture as improvements for speed and capacity continue to drive innovation. Additional reference information is included in the following short list of some of the common buses.

* 1. **PCI** - Peripheral Component Interconnect (PCI) is a local computer bus for attaching hardware devices to a computer and is part of the PCI Local Bus standard.
  2. **PCI Express** - Peripheral Component Interconnect Express (PCI Express), abbreviated as PCIe or PCI-e, is a high-speed serial computer expansion bus standard, designed to replace the older PCI, PCI-X and AGP bus standards.

**06 The Path from User Application to Hardware Access**

The Kernel is the connection between the computer hardware and end user applications. The levels of abstraction increase, the requirement for absolute control is relinquished to the Kernel and its device drivers to perform the desired actions.

**07 Connecting to Hardware (x86)**

The application program uses the operating system (OS). The OS controls the hardware, so when we use the computer, the application needs to talk to the OS. In order to manage the input and output devices, the OS needs to perform its control functions. As end-users, we interact with input devices such as typing. Typing a letter or moving the mouse and performing other functions is interpreted by the OS, the information is passed on to the application, and the application then performs the specific tasks that we want to conduct.

**08 Hardware Management**

Notice that the hardware is directly connected to the kernel, which provides services to the user. This interface is known as device drivers, rather than hardware control, since there are many devices to control. Some of the device drivers act on software "devices", which programs that react like real hardware, but are made entirely out of software.

**09 Types of Computers**

1. **Servers** - Servers serve content to other computers and client users, and are the backbone of how the modern cloud computing environment works today. Whether you are using email, messaging, streaming videos, or a web browser, your data is being served by a server. Servers are generally "workhorses" low on frills and high on compute power and/or storage.
2. **Desktops** - Desktop computers usually have monitors, keyboards and mice attached to them, as the user is typically using the machine for business applications like email, and custom applications relevant to the business (like "point of sale" or "finance" applications).
3. **Embedded** - Small purpose built computers are everywhere: cars, refrigerators, TVs and almost anything you can imagine. Many of these special purpose built systems run on highly customized versions of Linux. These are known as embedded systems.

**09 Backing Up a Linux System**

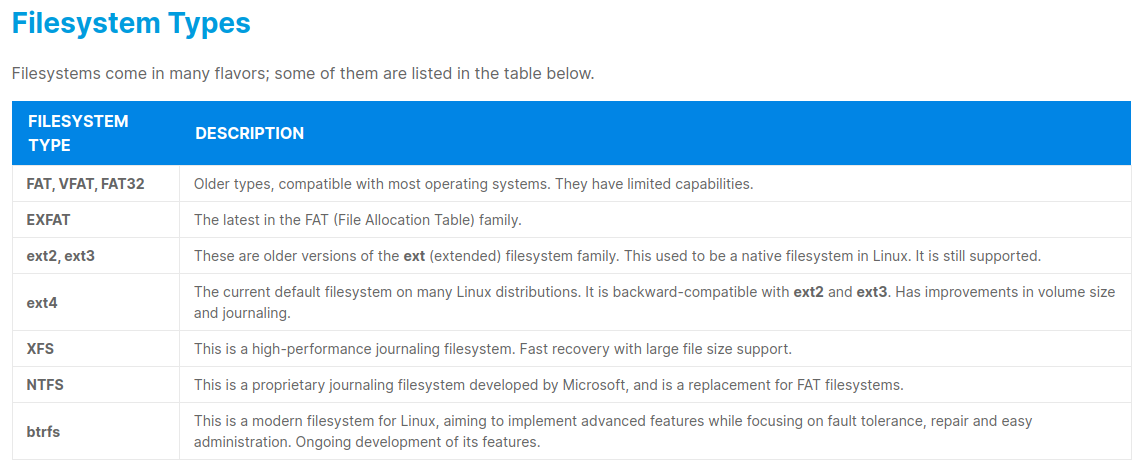
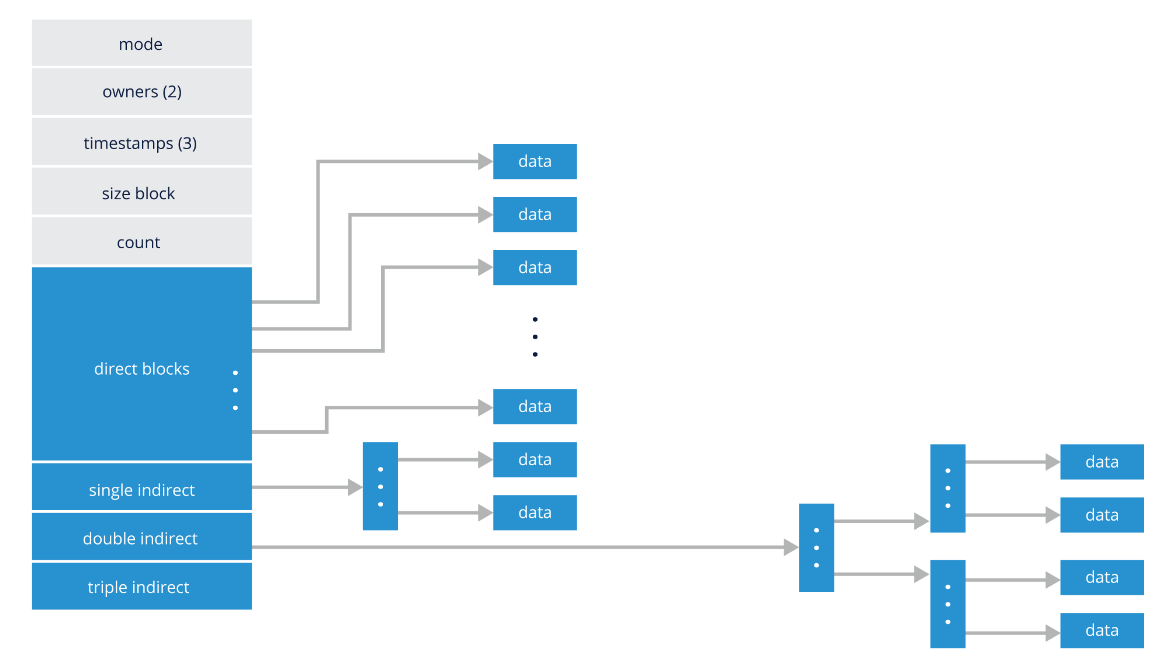
1. **tar** -- It is an easy-to-use backup utility. You can read more about tar in the available documentation.  
   **# tar -cvf <target\_file> <source-files or directory>**

An example of backing up the /home directory:  
**# tar -cvf /opt/backup.tar /home**

1. **rsync** -- rsync backs up differences via a block-by-block compare; this saves time by not continuously backing up the same thing. rsync uses ssh and ssh-formatted command structure, including SSL encryption on the connection. rsync has a vast array of options. An example of backing up the /home directory:  
   **# rsync -rzvh --exclude \*\.cache/ /home/. \ /opt/backups/**
2. **Amanda** -- Amanda (Advanced Maryland Automatic Network Disk Archiver) uses native utilities (including tar and dump) but is far more robust and controllable. Amanda is generally available on Enterprise Linux systems through the usual repositories.
3. **Bacula** -- Bacula is designed for automatic backup on heterogenous networks. It can be rather complicated to use and is recommended (by its authors) only to experienced administrators. Bacula is generally available on Enterprise Linux systems through the usual repositories.
4. **tar** -- It is an easy-to-use backup utility. You can read more about tar in the available documentation.  
   **# tar -cvf <target\_file> <source-files or directory>**
5. An example of backing up the /home directory:  
   **# tar -cvf /opt/backup.tar /home**

**10 Linux File System and Management**

An administrator needs to understand the basic filesystem functions and administration to be able to manage the storage on a system and add new storage.   
  
A filesystem is a software construct to manage the storage in a partition, regardless if the partition is contained in a virtual disk, physical disk, or Logical Volume (according to the Ubuntu wiki, logical volumes correspond to partitions: they hold a filesystem). A Linux system reads and writes files unlike other systems that are "disk sector"-oriented. The read or writes initiated by the user or user program are sent to the VFS (Virtual Filesystem) layer on the way to the device driver. The advantage to the VFS layer is the user environment sees all filesystems the same, a device to read and write, while the filesystem organizes the disk and maintains the appropriate control structures.   
  
In general terms, the filesystems have control blocks that are associated with files. The control blocks keep the file metadata, such as filename, permissions, owner, access time, creation time, and more. These control blocks are commonly known as index nodes or, simply, inodes. The inodes also point to the data blocks used by the file. There is a finite amount of space in an inode, so there may be intermediate pointer blocks to allow for more blocks; this is known as “indirect block pointers”. Large numbers of indirect blocks or levels of indirection may be used.



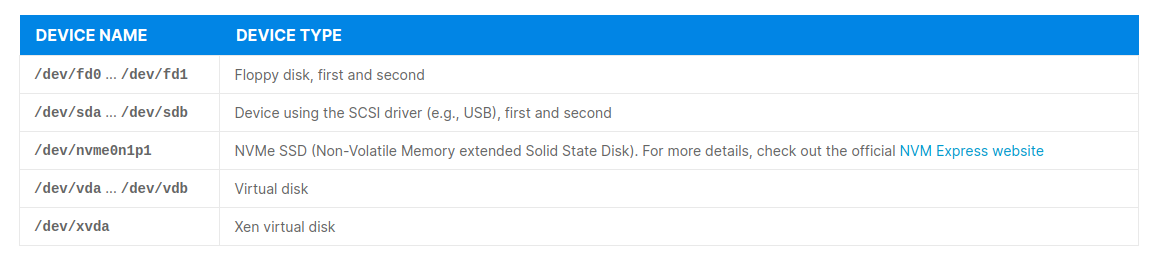
**11 Device Names**

An administrator needs to understand the basic filesystem functions and administration to be able to manage the storage on a system and add new storage. Device names are connected to the operating system through Kernel modules, and have the following general parameters, as seen below.

/dev/xxyn, where:

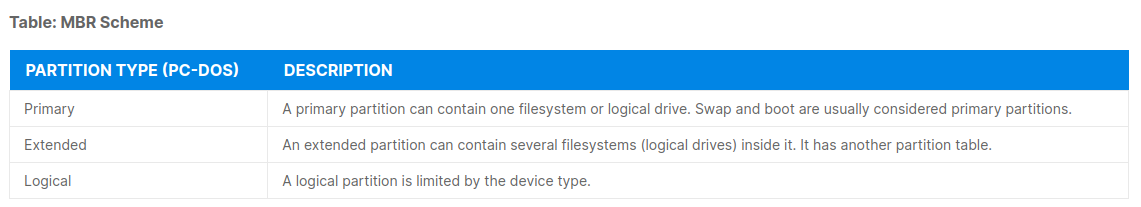
* xx is the type of device (sd = scsi, hd = ide, vd = virtual disk, etc.)
* y is a letter indicating the discovery order (sda = first, sdb = second, etc.)
* n is the partition number ('missing' = whole disk, 1 = first partition, 2 = second partition, etc.)

You can see some examples of filesystem device identifiers in the table below:

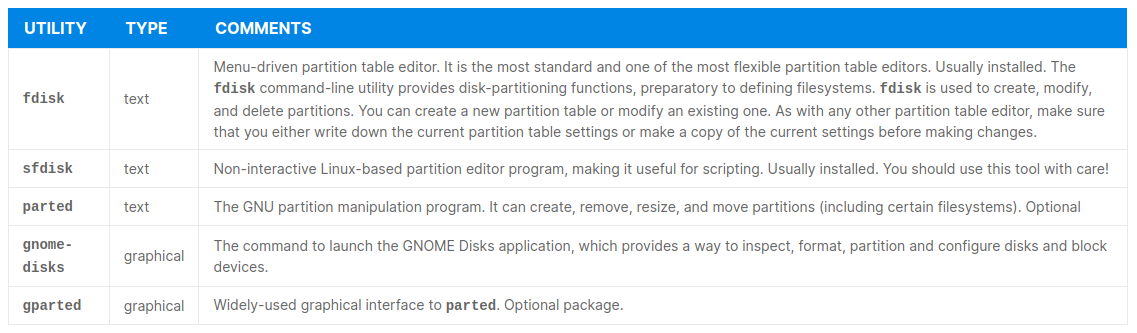
Most devices are described in the [Linux Kernel User's and Administrator's Guide](https://www.kernel.org/doc/html/latest/admin-guide/devices.html).

**12 Storage Management and Configuration**

Disk drives have had partitions on them since the early IBM PC days. A partition is a division of a hard drive disk (HDD) into logically independent sections.The original requirement for a partition was to divide the disk into smaller chunks because the filesystem at the time could not consume the whole disk.

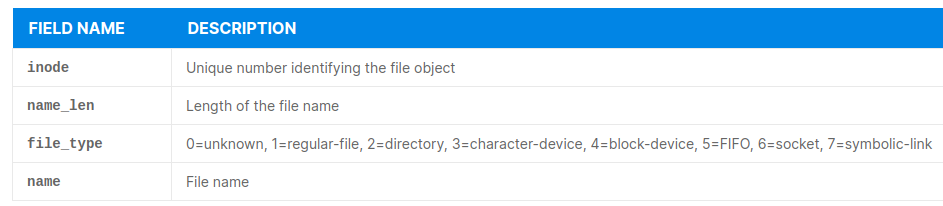
The choices back then either used a 3rd party format tool or partitioned the disk into smaller pieces. It was common to use partitions, and the operating systems at that time supported up to four partitions. This partitioning scheme is known as MBR (Master Boot Record).  
  
This MBR partitioning scheme still exists today. However, there is a new partitioning scheme called GPT (GUID Partition Table), where GUIDs are Globally Unique IDentifiers. The GPT can have up to 128 partitions per disk and is preferred in systems with a UEFI (Unified Extensible Firmware Interface), the replacement for PC-BIOS. Partitions help us section our storage drive into logically separate drives. This enables us to apply special parameters to sub-sections of the disk for security, performance, and other options.  
  
Layout of a disk with GPT. Based on the GUID Partition Table Scheme image by Kbolino, from Wikipedia:  
[https://en.wikipedia.org/wiki/GUID\_Partition\_Table#/media/File:GUID\_Partition\_Table\_Scheme.svg](https://en.wikipedia.org/wiki/GUID_Partition_Table" \l "/media/File:GUID_Partition_Table_Scheme.svg)   
  
**13 Disk Formatting Utilities**

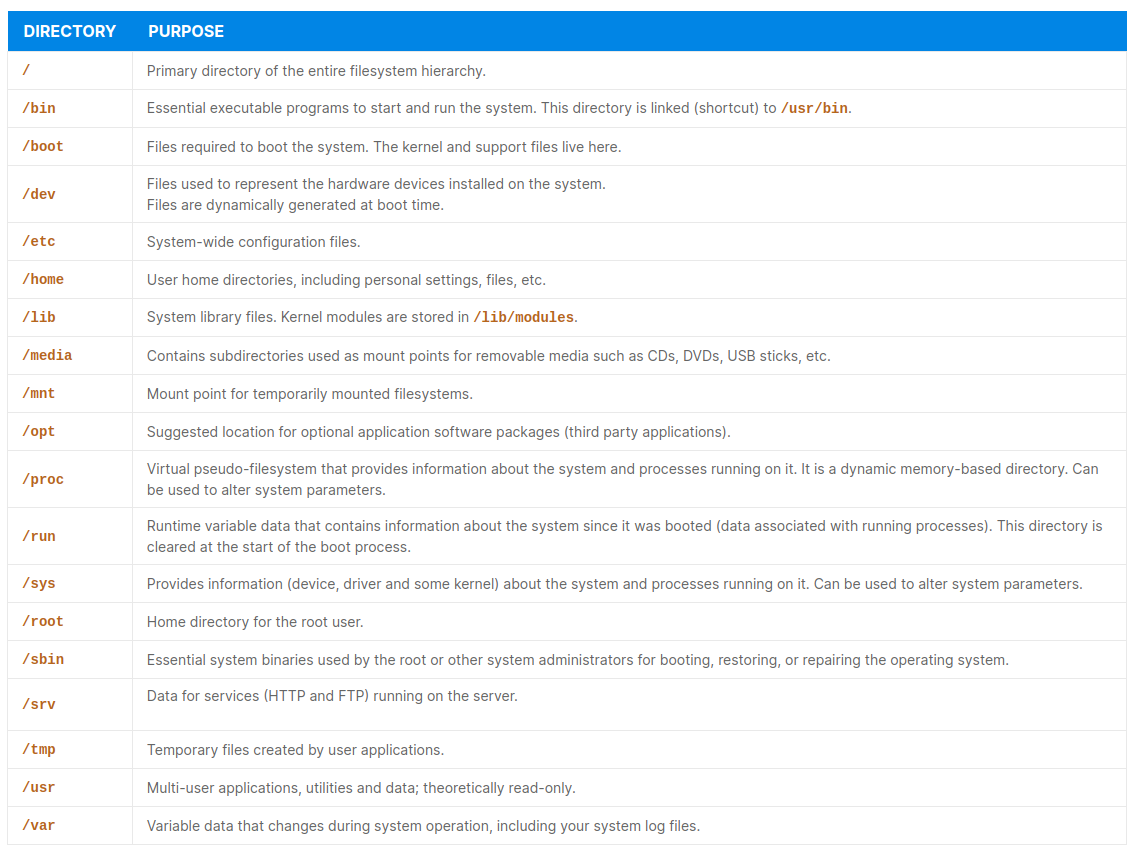
Several tools or utilities can be used to create and/or manipulate device partition tables and partitions. You can see some of these tools in the table below. Note: To partition devices, you should use a partitioning tool that is compatible with the chosen type of partition table. Using incompatible tools may result in the destruction of that table, along with existing partitions or data.

**14 Formatting a Disk and Adding a filesystem**

We will provide an old USB key on a VMware virtual machine. List the existing disk structures as a baseline:  
**# sudo lsblk -f | grep -v loop**  
you can use the following command to unmount old partitions:  
**# sudo umount /media/student/Fedra-WS-Live-25-1-3**  
  
**15 Directories and Files**

We will provide an old USB key on a VMware virtual machine.Unix-like systems store data and programs in files. These are organized in directories. Directories are data constructs (metadata) in an index node (or [inode](https://www.redhat.com/sysadmin/inodes-linux-filesystem)). The directory inode entry contains a list of inode entries for files and other directories.

A directory inode contains a lot of information, including the following fields:  
<https://www.redhat.com/sysadmin/inodes-linux-filesystem>   
By definition, an inode is an index node. It serves as a unique identifier for a specific piece of metadata on a given filesystem. Each piece of metadata describes what we think of as a file. Inodes operate on each filesystem, independent of the others. The filesystem id combines with the inode number to create a unique identification label.  
  
To check the number of inodes on your system, you can use the -i option with the df command, as seen here:  
**# df -i /dev/sda1**  
  
We can also look at the inode number of a specific file or directory, respectively. To do this, we use the ls -i command on the desired file. For example:  
**# ls -i DL-7C.pdf   
**#** df -i Music/   
[ Free online course:** [Red Hat Enterprise Linux technical overview](https://www.redhat.com/en/services/training/rh024-red-hat-linux-technical-overview?intcmp=701f20000012ngPAAQ)**. ]  
  
The** [Filesystem Hierarchy Standard (FHS)](https://refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.pdf) **defines a suggested logical location of directories and files on Linux distributions. These definitions include what should be contained in specific directories and files. The directory structure of the root directory in Linux can be seen executing the following command (more info on the man pages: man hier )  
**#** tree -L 1 -d /******16 Most common Filesystem Directories****

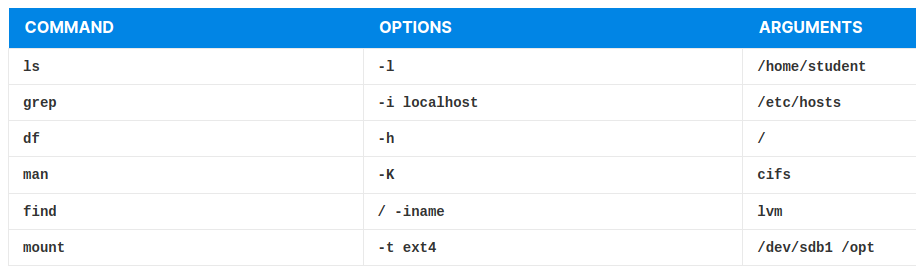
**Let’s now explore some of the most common filesystem directories:******17 Finding Files****

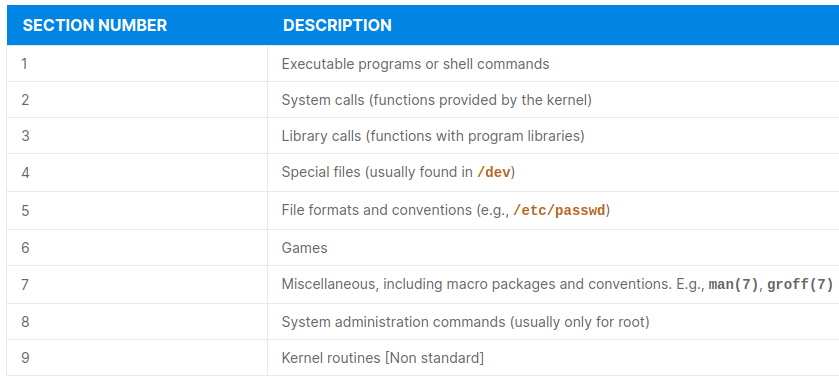
**Using the command line to locate files will become second nature over time; we discuss command line and commands later in the course.   
  
**18 Listing Filesystem Content****

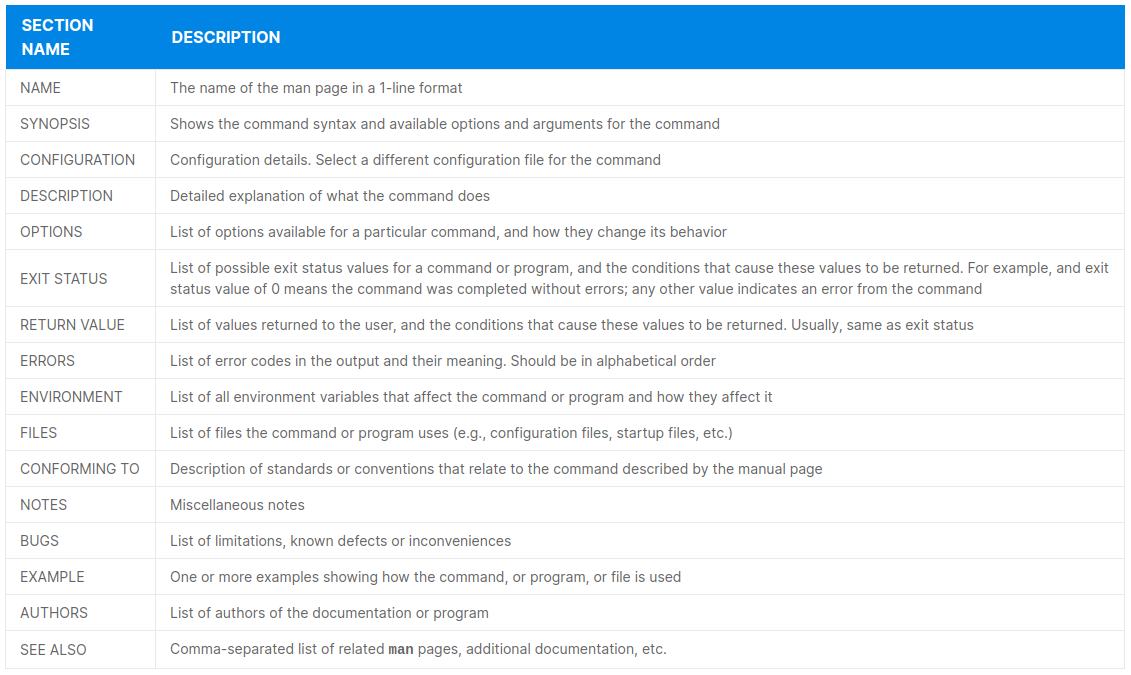
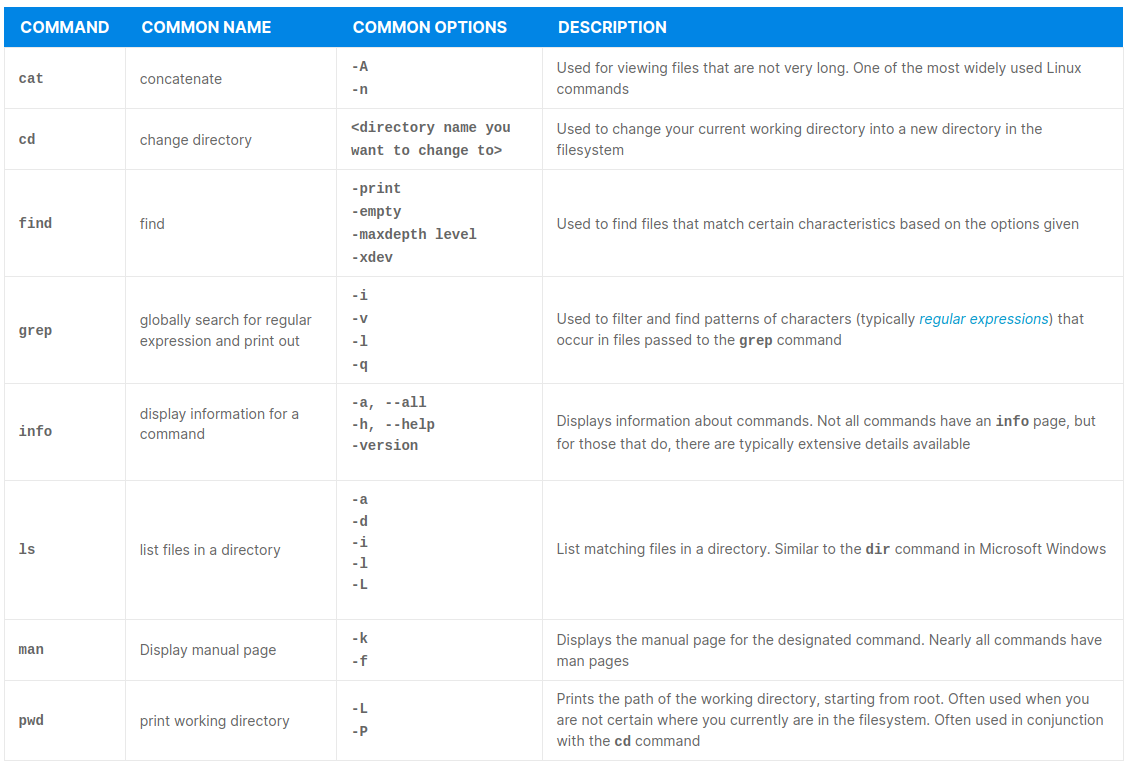
**The **ls** command is often used to list filesystem contents, usually for a directory. To list files in the current directory giving a detailed listing use:  
**# ls -l**  
To sort the list by **access time**, you can use:  
**# ls -t****

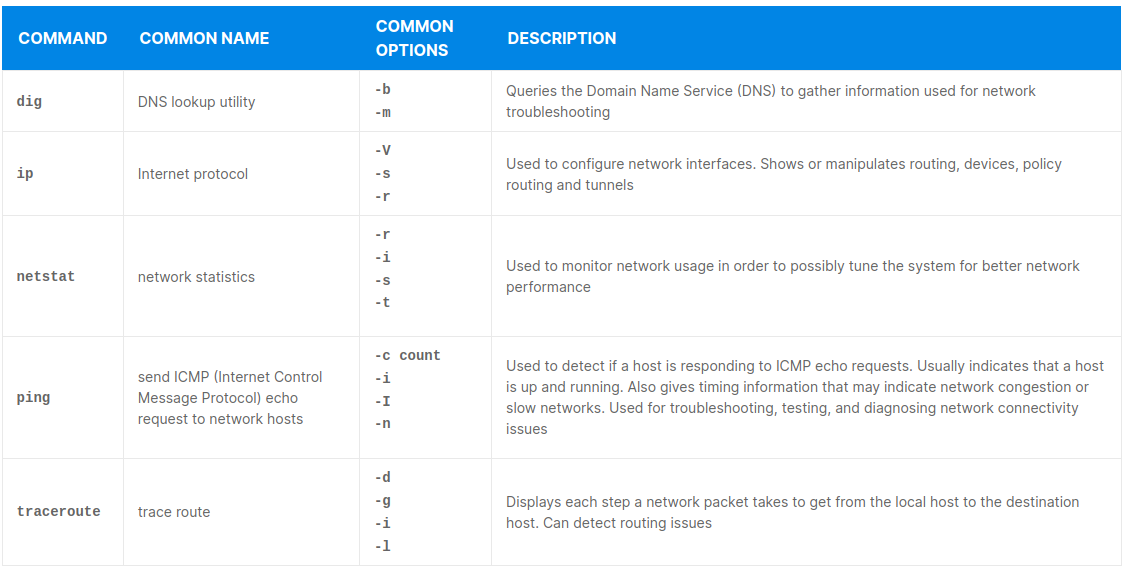
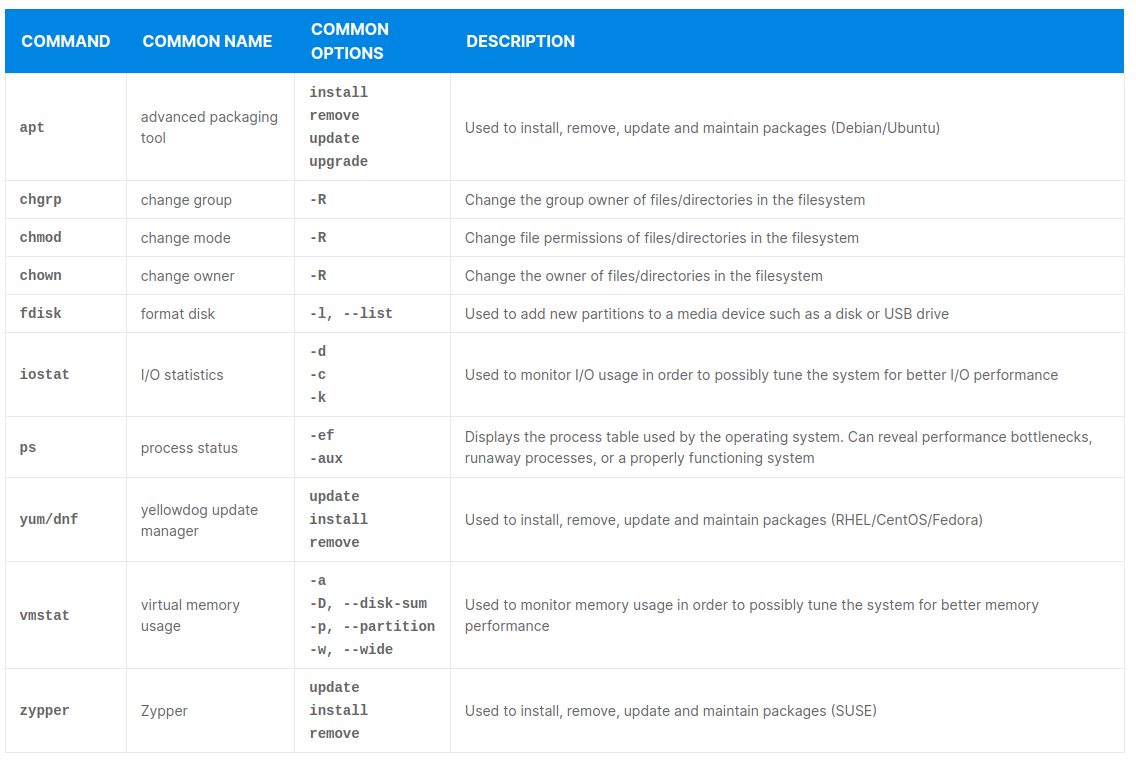
****19 User, Network and System Admin Commands****

**When using or administering Linux systems, commands are used all the time, for functions like changing a directory or folder, and listing the files in a directory. Using the command line provides clean and concise methods to query, display, or change the operation of a system.**

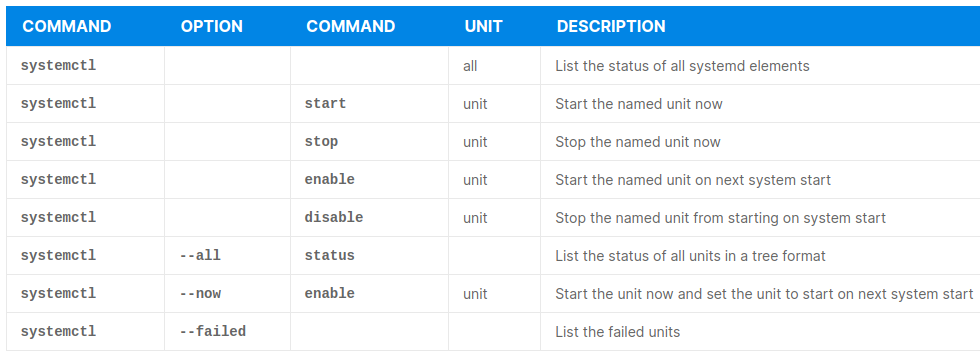
**The Graphical User Interface (GUI) is very useful, providing a bridge to new things, but the GUI is usually created after the commands have been in place for some time. You can see some examples in the table below:**

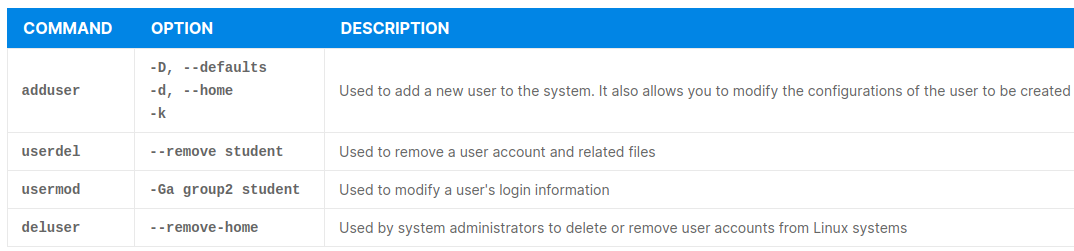
1. **sudo:** allows users to run programs using the security privileges of another user, generally root (superuser).
2. **man**: is the most often used utility for locating and gathering information on the system itself. It is easy to use and gives the most accurate information: **$ man The $ preceding the command in the above example is a convention used to say the command is going to be executed as a "regular user", not "root" as the admin user.   
   The table below shows the section numbers of the manual followed by the types of pages they contain:****Manual pages consist of several sections. The list of section names is not fixed, and not all section names may be available on a manual page. Conventional section names for man pages may include:**

A short command usage, along with a short list of options is typically available with -? or --help options:  
**$ man -?** or **$ man –help** Some general linux commands are:

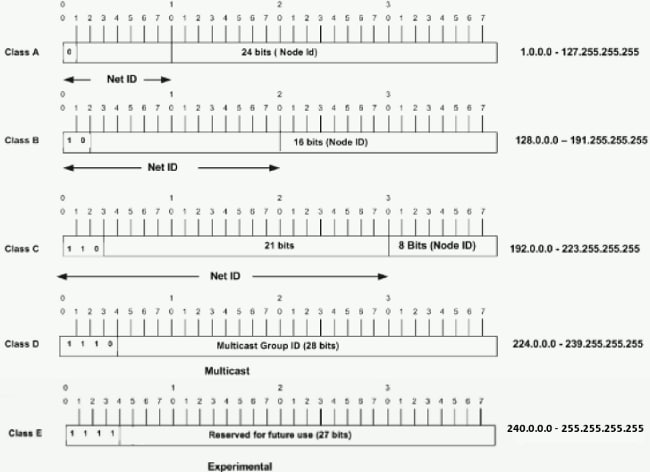
Some SysAdmin linux commands are:  
  
Some NetworkAdmin linux commands are:

****20 systemctl Command****

****systemctl** is a new Linux tool used to control the systemd system and service manager, allowing you to start, stop, or restart a service, get information about system units and about the state of the services in your system. A service or a unit is a resource or an object that systemd is aware of and knows how to address.**  
  
****21 User Admin Commands****

****# id # sudo id # sudo blkid # sudo fdisk -l /dev/sda**Check which port is opened: **# ss -t -a -n**Check available memory: **# free -m  
  
systemctl** is a new Linux tool used to control the systemd system and service manager, allowing you to start, stop, or restart a service, get information about system units and about the state of the services in your system. A service or a unit is a resource or an object that systemd is aware of and knows how to address.**  
  
****22 Networking and Networking Troubleshooting****

**Networking is a critical component for any computer these days, regardless if the computer connects to the Cloud or Internet. Communication between clients and servers plays a role in almost every computer. A detailed overview of network addressing, networks, subnets, and more can be found in the "**[IP Addressing and Subnetting for New Users](https://www.cisco.com/c/en/us/support/docs/ip/routing-information-protocol-rip/13788-3.html)**" documentation. Here you can find some additional information:   
  
**Address** - The unique number ID assigned to one host or interface in a network.  
**Subnet** - A portion of a network that shares a particular subnet address.  
**Subnet mask** - A 32-bit combination used to describe which portion of an address refers to the subnet and which part refers to the host.  
**Interface** - A network connection.  
**IP Address** - An IP address is an address used in order to uniquely identify a device on an IP network. The address is made up of 32 binary bits, which can be divisible into a network portion and host portion with the help of a subnet mask. The 32 binary bits are broken into four octets (1 octet = 8 bits). Each octet is converted to decimal and separated by a period (dot). For this reason, an IP address is said to be expressed in dotted decimal format (for example, 172.16.81.100). The value in each octet ranges from 0 to 255 decimal, or 00000000 - 11111111 binary.**  
These octets are broken down to provide an addressing scheme that can accommodate large and small networks. There are five different classes of networks, A to E. This document focuses on classes A to C, since classes D and E are reserved and discussion of them is beyond the scope of this document. Also note that the terms "Class A, Class B" and so on are used in this document in order to help facilitate the understanding of IP addressing and subnetting. These terms are rarely used in the industry anymore because of the introduction of [classless interdomain routing (CIDR)](https://www.cisco.com/c/en/us/support/docs/ip/routing-information-protocol-rip/13788-3.html" \l "CIDR).   
  
Given an IP address, its class can be determined from the three high-order bits (the three left-most bits in the first octet). Figure 1 shows the significance in the three high order bits and the range of addresses that fall into each class. For informational purposes, Class D and Class E addresses are also shown:

In a Class A address, the first octet is the network portion, so the Class A example in Figure 1 has a major network address of 1.0.0.x - 127.255.255.x (where x can go from 0 to 255). Octets 2, 3, and 4 (the next 24 bits) are for the network manager to divide into subnets and hosts as he/she sees fit. Class A addresses are used for networks that have more than 65,536 hosts (actually, up to 16777214 hosts!).

In a Class B address, the first two octets are the network portion, so the Class B example in Figure 1 has a major network address of 128.0.0.x - 191.255.255.x. Octets 3 and 4 (16 bits) are for local subnets and hosts. Class B addresses are used for networks that have between 256 and 65534 hosts.

In a Class C address, the first three octets are the network portion. The Class C example in Figure 1 has a major network address of 192.0.0.x - 223.255.255.x. Octet 4 (8 bits) is for local subnets and hosts - perfect for networks with less than 254 hosts.

****23 Network Masks****

**A network mask helps you know which portion of the address identifies the network and which portion of the address identifies the node. Class A, B, and C networks have default masks, also known as natural masks, as shown here:  
  
Class A: 255.0.0.0  
Class B: 255.255.0.0  
Class C: 255.255.255.0**  
An IP address on a Class A network that has not been subnetted would have an address/mask pair similar to: 10.20.15.1 255.0.0.0. In order to see how the mask helps you identify the network and node parts of the address, convert the address and mask to binary numbers.  
  
10.20.15.1 = 00001010.00010100.00001111.00000001  
255.0.0.0 = 11111111.00000000.00000000.00000000  
  
Once you have the address and the mask represented in binary, then identification of the network and host ID is easier. Any address bits which have corresponding mask bits set to 1 represent the network ID. Any address bits that have corresponding mask bits set to 0 represent the node ID.  
  
10.20.15.1 = 00001010.00010100.00001111.00000001  
255.0.0.0 = 11111111.00000000.00000000.00000000  
 --------------------------------------------------------  
 net id | host id  
netid = 00001010 = 10  
hostid = 00010100.00001111.00000001 = 20.15.1

****24 Cloud Computing Basics****  
"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction".  
  
**Key Features of Cloud Computing:**

1. **Increased Performance and Agility**

**Cloud resources are provisioned with a few clicks, saving time and providing agility. They also scale up or down with ease, by increasing or decreasing the number of running instances of a particular resource as a response to current demand or based on projected future demand. An additional characteristic is elasticity, allowing some cloud resources instances to increase or decrease in size, which is also demand-driven.**

1. **Cost Savings**

**By reducing the upfront cost to set up the infrastructure, it allows users to focus on applications supporting the business processes. Another beneficial feature of cloud providers is a cost estimator, which assists users in better resource planning. Most cloud providers have a pay-as-you-go model, so users pay only for resources used.**

1. **Easy Access to Resources**

**Users can access cloud services from any place and device, as long as there is connectivity to the cloud service provider.**

1. **Maintenance Offloaded to Provided**

**All the lights-on maintenance work to keep resources up, running, and up-to-date is offloaded onto the provider, and it is no longer the user's concern.**

1. **Multi-tenancy**

**Multiple users share the same pool of resources, which helps to drive down the costs incurred by each user.**

1. **Increased Reliability**

**Resources can be hosted in different data center locations, to provide increased reliability, resiliency and high-availability.**

1. **Increased Security**

**Cloud providers have security embedded into their offered service. Data moving from one cloud node to another is always encrypted. Most cloud providers provide users with a secured login to access the cloud services. The security implementation would be considered as a "mostly closed" configuration to bolster the security of the services running on the cloud environment. Users can customize some of the security settings to allow applications to have access to their data, customize the security parameters, monitor key areas and send alerts as required.  
  
**25 Cloud Computing Delivery Models****

**A cloud delivery model represents a specific combination of IT resources offered by a cloud provider. Four common cloud delivery models have become widely established and formalized.**

1. ****Infrastructure as a Service (IaaS)****

**Infrastructure as a Service (IaaS) provisions processing, storage, networks, and other fundamental computing resources where the client is able to deploy and run system and network software. IT has control of operating systems, storage, and deployment of applications. The IaaS delivery model represents a self-contained IT environment consisting of infrastructure-centric IT resources that can be accessed and managed via cloud service-based interfaces and tools.**

1. ****Platform as a Service (PaaS)****

**Platform as a Service (PaaS) provides the capability to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The developer does not manage or control the underlying cloud infrastructure, including the network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment. The PaaS delivery model represents a predefined "ready-to-use" environment typically composed of already deployed and configured IT resources. Specifically, PaaS relies on (and is primarily defined by) the usage of a ready-made environment that establishes a set of pre-packaged products and tools used to support the entire delivery lifecycle of custom applications.**

**Common reasons a cloud customer would use and invest in a PaaS environment include:**

* **Extend on-premise environments into the cloud for scalability and economic purposes.**
* **Use the ready-made environment to entirely substitute an on-premise environment, and**
* **To become a cloud provider and deploy its own cloud services to be made available to other external cloud consumers.**

1. ****Software as a Service (SaaS)****

**Software as a Service (SaaS) is to use the provider’s applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser, or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.**

**A software program positioned as a shared cloud service and made available as a "product" or generic utility represents the typical profile of a SaaS offering. The SaaS delivery model is typically used to make a reusable cloud service widely available (often commercially) to a range of cloud consumers. An entire marketplace exists around SaaS products that can be leased and used for different purposes and via different terms.**

**A cloud consumer is generally granted very limited administrative control over a SaaS implementation. It is most often provisioned by the cloud provider, but it can be legally owned by whichever entity assumes the cloud service owner role. For example, an organization acting as a cloud consumer while using and working with a PaaS environment can build a cloud service that it decides to deploy in that same environment as a SaaS offering. The same organization then effectively assumes the cloud provider role as the SaaS-based cloud service is made available to other organizations that act as cloud consumers when using that cloud service.**

1. ****Function as a Service (FaaS)****

**Function as a Service (FaaS), also known as "Serverless Cloud", is a cloud model that allows the developer to focus on the application components or microservices that are then managed by the cloud environment.**

**This model is demand-oriented, so costs are only accumulated as the application runs, not the infrastructure.  
  
**26 Types of Cloud Deployments****

* **Public Cloud:** A public cloud is open to anyone, usually billed on a "pay-per-use" model. Some examples of public cloud providers are: Amazon Web Services, Microsoft Azure, Google Cloud Platform and DigitalOcean.
* **Private Cloud**: Private cloud implementations are "in house", usually with limited or controlled access to any outside communication.
* **Hybrid Cloud**: Sometimes, a combination of private and public clouds are used. This allows the application "frontend" or user interface to be on a public cloud, allowing all potential customers the appropriate access and keeping the "backend" or data store restricted from undesired communication.

****27 Cost Savings Using Cloud Environments****

****The first place we look for savings is during the planning and implementation of a cloud environment****

* **Some highly regulated environments mandate that data stay local for reasons beyond the scope of this course, but that does not end the cloud story; by identifying the things that need to stay local, it will highlight the items that can move to the cloud. For example, for one organization, the database servers had to remain in their concrete vault, but the systems that make the database requests did not; this allows for web frontend servers, data verification servers and firewalls that can be moved to the cloud. The servers that have the most flexible traffic flow can live in the cloud, where the cloud environment can respond to the demands of the user community. This could be an excellent hybrid cloud implementation: cost savings on the dynamic components, safety for the fixed components.**
* **The amount of hardware required locally for cloud environments can be next to nothing if all of the processing and storage is moved to the cloud. This could be the start of the completely virtual datacenter. This can provide extremely reliable and flexible networks to connect the business traveller and the work-from-home crowd that simply need web connections to the servers.**
* **If the hardware is virtualized, the requirements for onsite personnel changes; their new role could be in cloud management, performance and security monitoring.**
* **As the real servers age, their maintenance costs could become another reason to pay-as-you-go in the cloud rather than investing in new physical systems.**
* **Planning, implementation and cost analysis services are available from all the major cloud providers.**
* ****The second stream of cost savings is after the initial installation—the ongoing costs for cloud usage. Things that can bring down costs:**  
    
  **Right size**: During the planning implementation stage, there are usually some cases of virtualizing existing servers in the cloud; this may not be the most economical path. Servers are usually configured for peak demand, so it is not unusual for free-standing servers to be running less than 100% all the time, but that is usually what you are paying for. Selecting the right size machine in the cloud is important, as is post installation monitoring to be sure the cloud-based resource utilization is right sized. Having a huge machine in the cloud not doing anything is still being billed as a top shelf server. Monitoring can identify application system size misfits.**
* ****Autoscaling** may be the solution to variable needs systems. Perhaps a little retooling of the application can make it truly scalable from tens to thousands. For example, a local firm recently completed its testing on their new cloud implementation with a dozen testers; the day after the application was approved, 2500 users connected to the system. Autoscaling is the ultimate pay-for-what-you-use model.**
* ****Lower performance resources**: As the cloud providers grow, they often have less performant resources available; sometimes, these can be available at aggressive pricing. This will be different between cloud providers, but it could be a savings on non-critical systems.**
* **Idle resources: It happens. Be on the lookout for unused or low usage resources, they still get billed.**

1. **asgasdf**