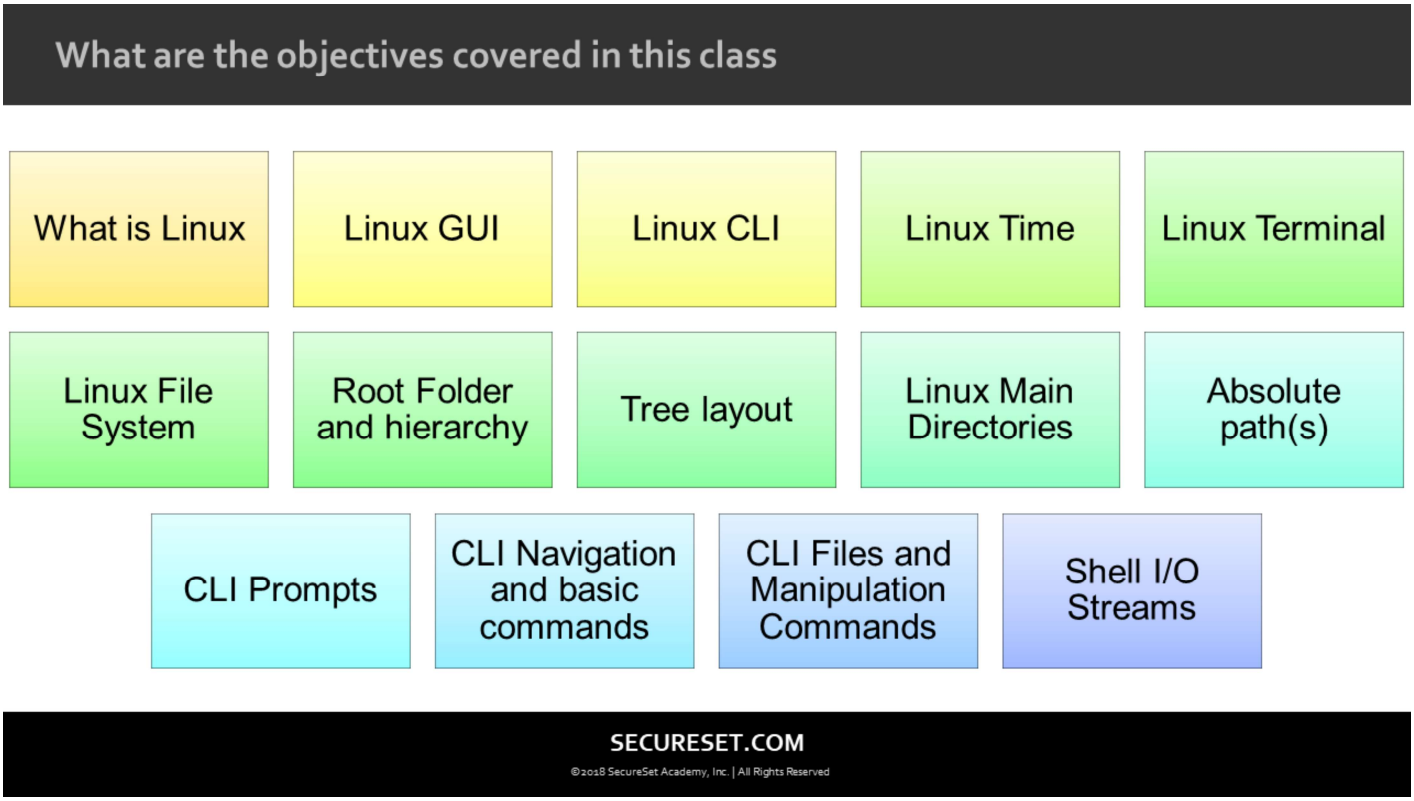
Introduction to Linux

SECURESET ACADEMY

Hacking 101

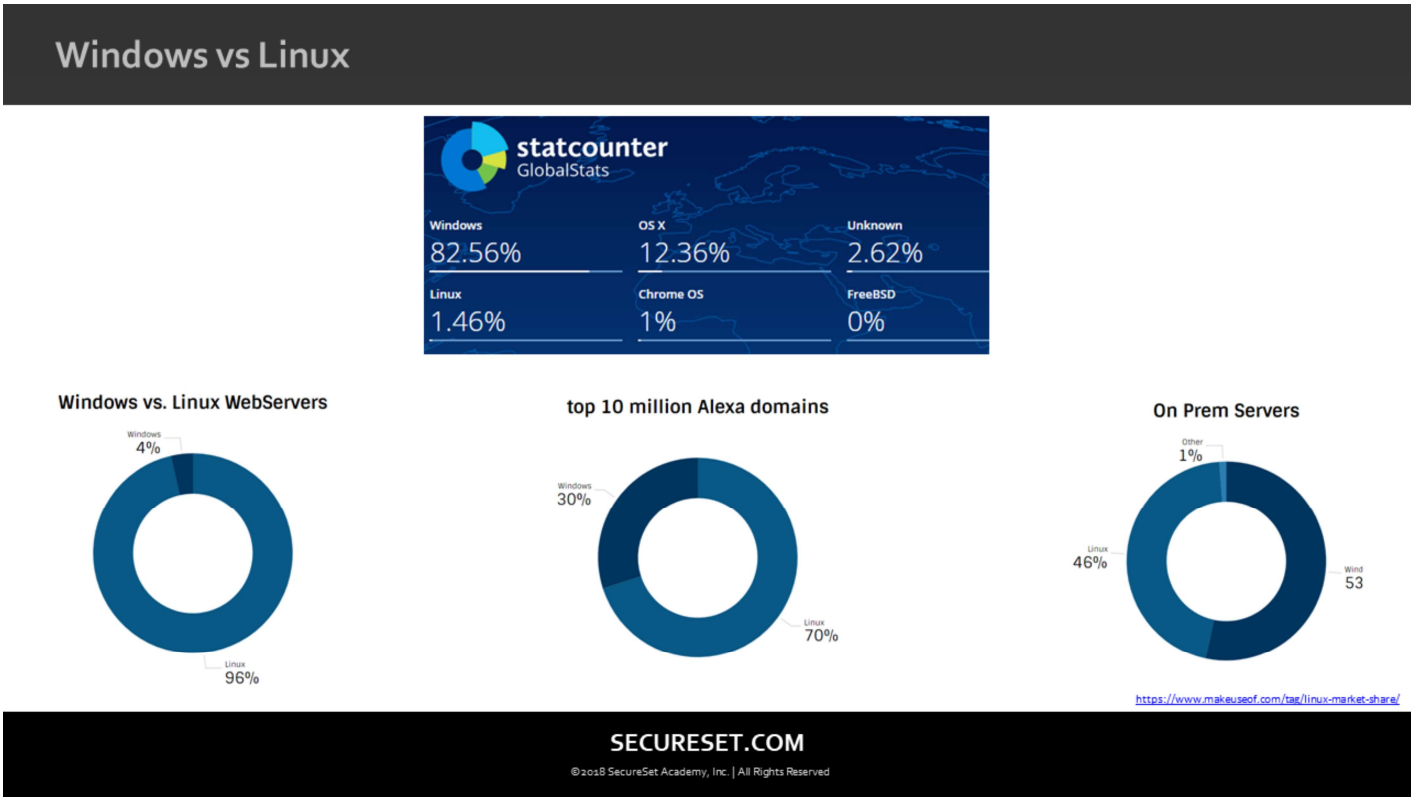
i. How just a few hours?
ii. Curate the information
    1. best and most effective understanding of computers
    2. particular sequence
    3. Enter SecureSet's programs with a solid foundation.
iii. Can't go deep into these subjects
    1. expose you to them
    2. connect dots
    3. Your responsibility to keep going
iv. Just doing the bare minimum isn't enough.
v. Find your passion and follow it.



Here are a list of objectives that we aimed to cover in this SecureSet Systems 2 class.

You should feel comfortable with these topics in a conversational and assessment manner. We encourage you to continue studying on your own.

Amongst these above topics are a very large and numerous amount of jobs. Plus it is the foundation for the rest of your education here at SecureSet.



Top Shows the Distribution of Desktops and Laptops

The bottom donut charts, show you the distribution of Linux vs. Windows SERVERS.

While Windows is still king at the desktop / laptop area, Linux is clearly very prevalent in the server space.

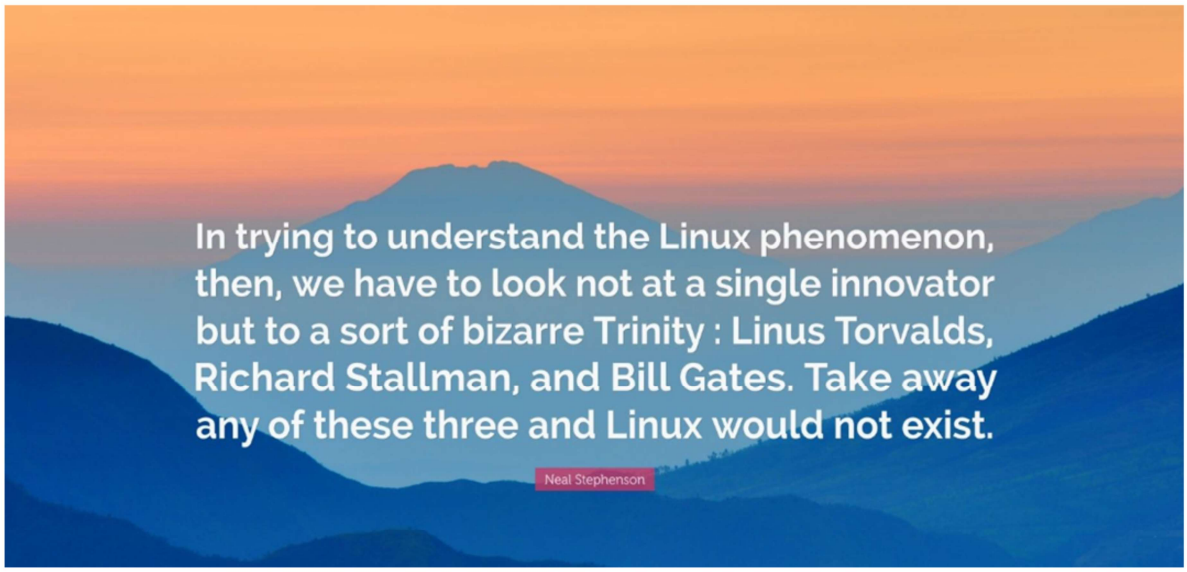
Windows vs Linux

SECURESET.COM

Both Operating systems are very important from a Cybersecurity standpoint and from SecureSets perspective you need to know how to operate in both Operating Systems worlds and HOW they interact with each other.

To be good at cybersecurity you have to know everything.

The Linux Phenomenon



SECURESET.COM

1. Linux is as much a phenomenon as it is an operating system.
2. To understand why Linux has become so popular, it is helpful to know a little bit about its history.
    a. The first version of UNIX was originally developed several decades
        i. primarily as a research operating system in universities.
        ii. High-powered desktop workstations from companies like Sun proliferated in the 1980s, and they were all based on UNIX.

Linus Torvalds, he's kinda a big thing.

SECURESET.COM

a. Linux stepped into this odd landscape and captured a lot of attention.
b. The Linux kernel, created by Linus Torvalds, was made available to the world for free.
c. Torvalds then invited others to add to the kernel provided that they keep their contributions free.

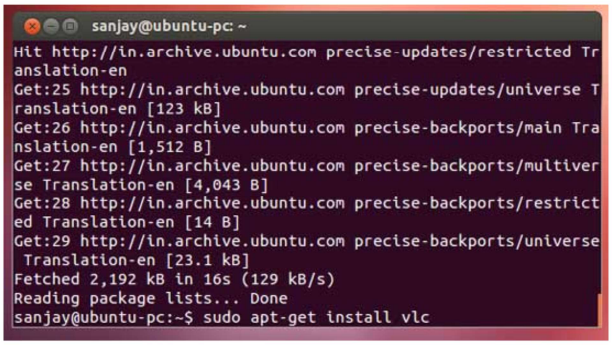
What a GUI?

SECURESET.COM

1. **GUI**
    a. The graphical user interface is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, instead of text-based user interfaces, typed command labels or text navigation.
    b. GUIs were introduced in reaction to the perceived steep learning curve of command-line interfaces which require commands to be typed on a computer keyboard.
    c. A GUI uses a combination of technologies and devices to provide a platform that users can interact with, for the tasks of gathering and producing information.
    d. A series of elements conforming a visual language have evolved to represent information stored in computers. This makes it easier for people with few computer skills to work with and use computer software. The most common combination of such elements in GUIs is the **windows**, **icons**, **menus**, **pointer** (**WIMP**) paradigm, especially in personal

computers.

What's CLI?



SECURESET.COM

1. **CLI**
    a. A command-line interface or command language interpreter (CLI), also known as command-line user interface. A program which handles the interface is called a command language interpreter or shell (computing).
        i. Command-line interfaces to computer operating systems are less widely used by casual computer users, who favor graphical user interfaces or menu-driven interaction.
        ii. Compared with a graphical user interface, a command line requires fewer system resources to implement.
        iii. Since options to commands are given in a few characters in each command line, an experienced user finds the options easier to access.
        iv. Automation of repetitive tasks is simplified - most operating systems using a command line interface support some mechanism for storing frequently used sequences in a disk file, for re-use; this may extend to a scripting language that can take

parameters and variable options.

v. A command-line history can be kept, allowing review or repetition of commands.
vi. A command-line system may require paper or online manuals for the user's reference, although often a "help" option provides a concise review of the options of a command. - Man pages or online resources and wiki's
vii. The command-line environment may not provide the graphical enhancements such as different fonts or extended edit windows found in a GUI.
viii. It may be difficult for a new user to become familiar with all the commands and options available, compared with the drop-down menus of a graphical user interface, without repeated reference to manuals.
ix. The interface is usually implemented with a command line shell, which is a program that accepts commands as text input and converts commands into appropriate operating system functions.
x. Command-line interfaces are often preferred by more advanced computer users, as **they often provide a more concise and powerful means to control a program or operating system**.
xi. Programs with command-line interfaces are generally easier to automate via scripting.
xii. GUI interfaces have to output interpreted commands into the CLI for computers to work.
xiii. If an advanced user understands the commands in the CLI well enough, there is more power and flexibility directly from the Command Line that is ever implemented in a GUI. This is why from SecureSet's perspective true mastery over computers comes from becoming familiar and effective from the command line.

1. **Time Trivia for Unix / Linux**
    a. For Unix like operating systems, the epoch is 00:00:00 Coordinated Universal Time (UTC), Thursday, 1 January 1970.
    b. It is a system for describing a point in time, defined as an approximation of the number of seconds that have elapsed since epoch time.
    c. However, Unix time is not a true representation of UTC, as leap seconds are not accounted for. A leap second in UTC shares the same Unix time as the second which came before it.
    d. It's not SERIOUSLY important for you to know, but it's nice to be aware of it.
    e. To display epoch time the command is simply date +%s

What is a abstraction redux

SECURESET.COM

1. Trivia time again!

2. A abstractions is the process of taking away or removing characteristics from something in order to reduce it to a set of essential characteristics.
    a. In Computer file systems, a file is a singular object.

    b. A folder is a container for those objects.

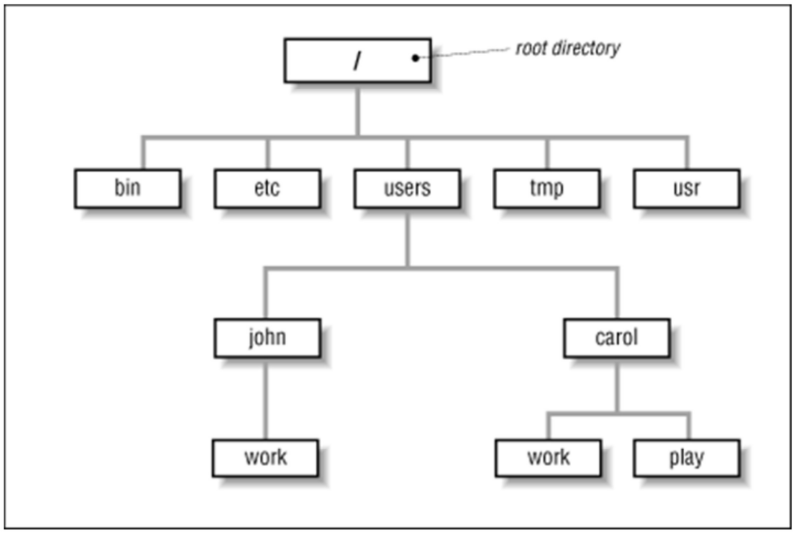
        i. You can have tons, millions of objects in a **container**. Which is a folder.

c. You can have folders *IN* folders creating a hierarchy. They can be folders in folders in folders. Get the idea?

d. It's rather like a group of Russian Babushka Nesting dolls.

1. It's important to understand this relationship because we are about to embark on the tale of the **FILE SYSTEM.**

What is a file system? (Linux Simplified version)



SECURESET.COM

1. **LINUX FILE SYSTEM -- This is a simplified view**
    a. Linux Saying

        i. "**On a UNIX system, everything is a file; if something is not a file, it is a process**."

        ii. This statement is true because there are special files that are more than just files but to keep things simple, saying that everything is a file is an acceptable generalization. A Linux system, just like UNIX, makes no difference between a file and a directory, since a directory is just a file containing names of other files. Programs, services, texts, images, and so forth, are all files. Input and output devices, and generally all devices, are considered to be files, according to the system.

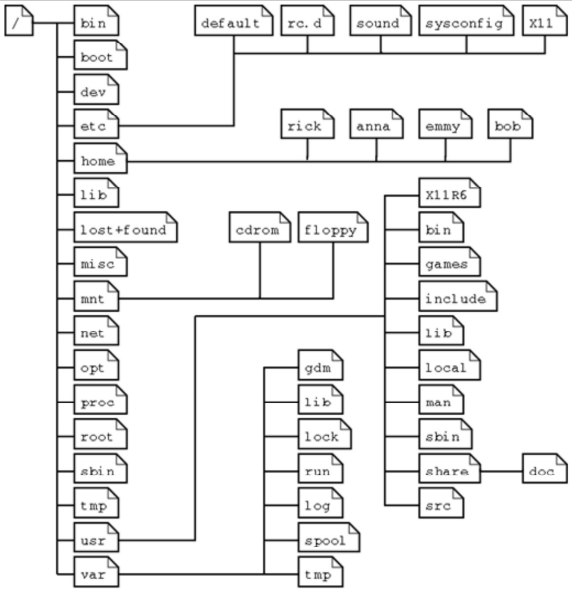
        iii. In order to manage all those files in an orderly fashion, man likes

to think of them in an ordered tree-like structure on the hard disk, as we know from MS-DOS (Disk Operating System) for instance. The large branches contain more branches, and the branches at the end contain the tree's leaves or normal files.

a. For now, we will use an image of the tree, to wrap our heads around this.

b. Depending on the system admin, the operating system and the mission of the UNIX machine, the structure may vary, and directories may be left out or added at will. The names are not even required; they are only a convention.

File System part deux



SECURESET.COM

a. MAIN DIRECTORIES

i. /**bin** is a place for most commonly used terminal commands, like ls, mount, rm, etc.

ii. /**boot** contains files needed to start up the system, including the Linux kernel, a RAM disk image and bootloader configuration files.

iii. /**dev** contains all device files, which are not regular files but instead refer to various hardware devices on the system, including hard drives.

iv. /**etc** contains system-global configuration files, which affect the system's behavior for all users. – How its pronounced.

v. /**home** home sweet home, this is the place for users' home

directories.

vi. **/lib** contains very important dynamic libraries and kernel modules

vii. /**media** is intended as a mount point for external devices, such as hard drives or removable media (floppies, CDs, DVDs).

viii. /**mnt** is also a place for mount points, but dedicated specifically to "temporarily mounted" devices, such as network filesystems.

ix. /**opt** can be used to store additional software for your system, which is not handled by the package manager.

x. /**proc** is a virtual filesystem that provides a mechanism for kernel to send information to processes.

xi. /**root** is the superuser's home directory, not in /home/ to allow for booting the system even if /home/ is not available.

xii. /**run** is a tmpfs (temporary file system) available early in the boot process where ephemeral run-time data is stored. Files under this directory are removed or truncated at the beginning of the boot process.

    1. (It deprecates various legacy locations such as /var/run, /var/lock, /lib/init/rw in otherwise non-ephemeral directory trees as well as /dev/.* and /dev/shm which are not device files.)

xiii. /**sbin** contains important administrative commands that should generally only be employed by the superuser.

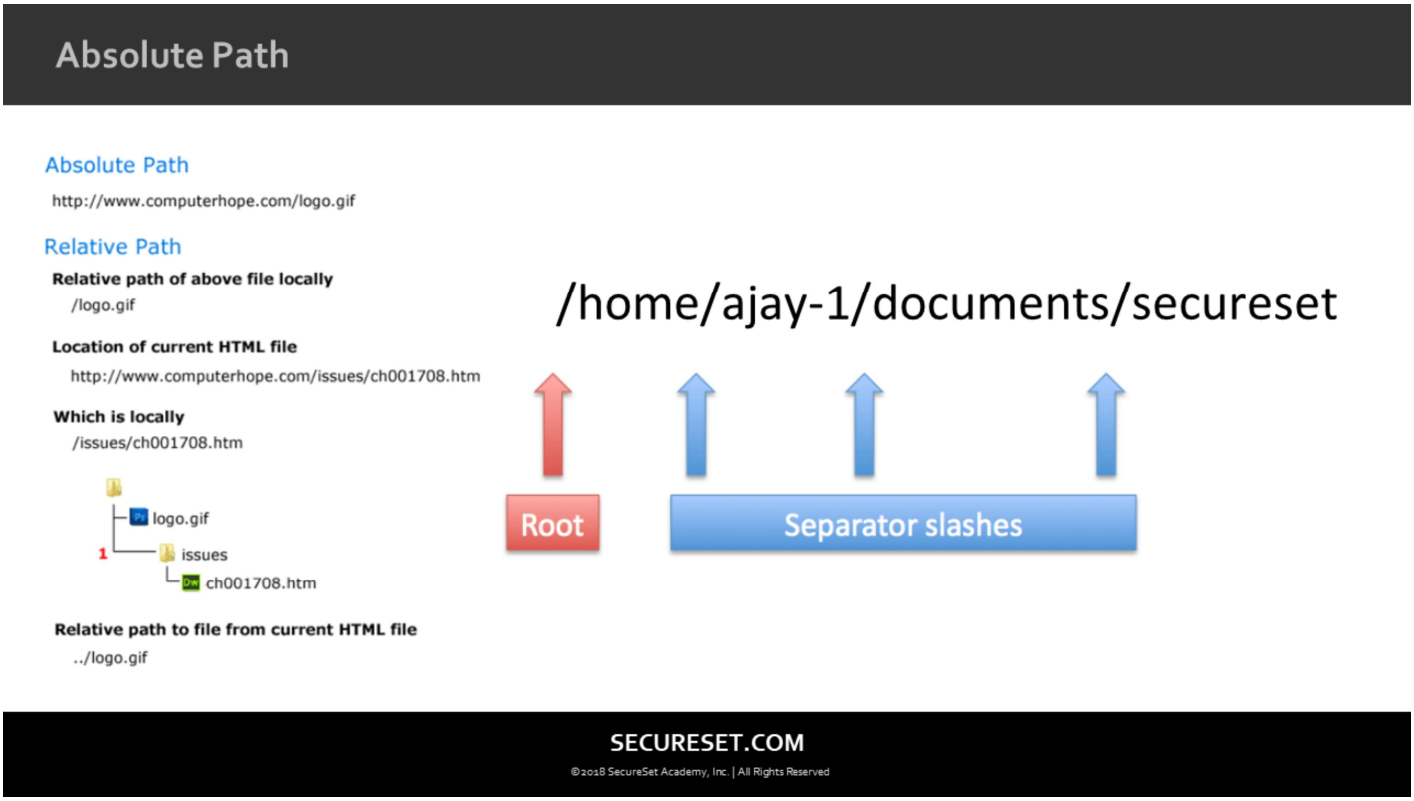
xiv. /**srv** can contain data directories of services such as HTTP (/srv/www/) or FTP.

xv. /**sys** is a virtual filesystem that can be accessed to set or obtain information about the kernel's view of the system.

xvi./**tmp** is a place for temporary files used by applications.

xvii./**usr** contains the majority of user utilities and applications, and partly replicates the root directory structure, containing for instance, among others, /usr/bin/ and /usr/lib.

xviii./**var** is dedicated to variable data, such as logs, databases, websites, and temporary spool (e-mail etc.) files that persist from one boot to the next. A notable directory it contains is /var/log where system log files are kept.



1. ABSOLUTE PATH
    a. An absolute path name, pointing to what is normally an executable file on an Ubuntu system:
        i. /usr/bin/test

    b. An absolute path name, but pointing to a directory instead of a regular file:
        i. /usr/bin/

    c. A relative path name, which will point to /usr/bin/test only if the current directory is /usr/:
        i. bin/test

    d. A relative path name, which will point to /usr/bin/test if the current directory is any directory in /usr/, for instance /usr/share/:
        i. ../bin/test

a. A path name using the special shortcut ~, which refers to the current user's home directory:
    i. ~/Desktop/

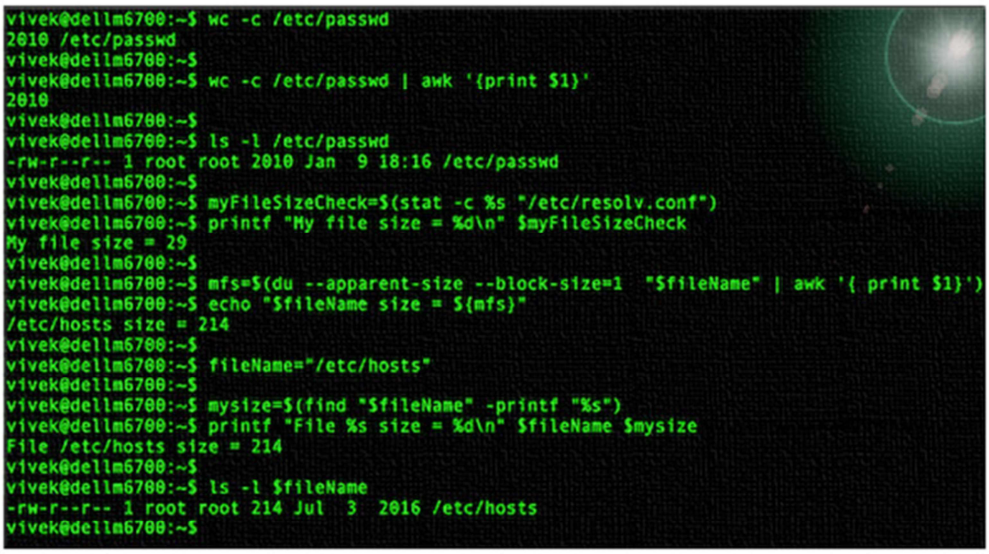
b. Path names can contain almost any character, but some characters, such as space, must be escaped in most software, usually by enclosing the name in quotation marks:
    i. "~/Examples/Experience ubuntu.ogg“

c. or by employing the escape character \:
    i. ~/Examples/Experience\ ubuntu.ogg

d. ./config means you're calling something in the current working directory. In this case config is an executable. You have to specify the path for executables if they're outside your $PATH variable and that's why config isn't enough.

e. ../config would be used if the config executable were in the parent of the current working directory.

CLI Basics



SECURESET.COM

1. Command Line Basics

a. The bash shell is one of several shells available for Linux.

b. It is also called the Bourne-again shell, after Stephen Bourne, the creator of an earlier shell (/bin/sh).

c. Bash is substantially compatible with sh, but it provides many improvements in both function and programming capability.

d. No this has nothing to do with Jason Bourne.

2. To OPEN a TERMINAL

a. If you're running Unity: open the dash, type terminal, hit Return.

b. If you're on the old style menus, Applications → Accessories → Terminal.

c. Control + Alt + T.

1. **What is a command?**

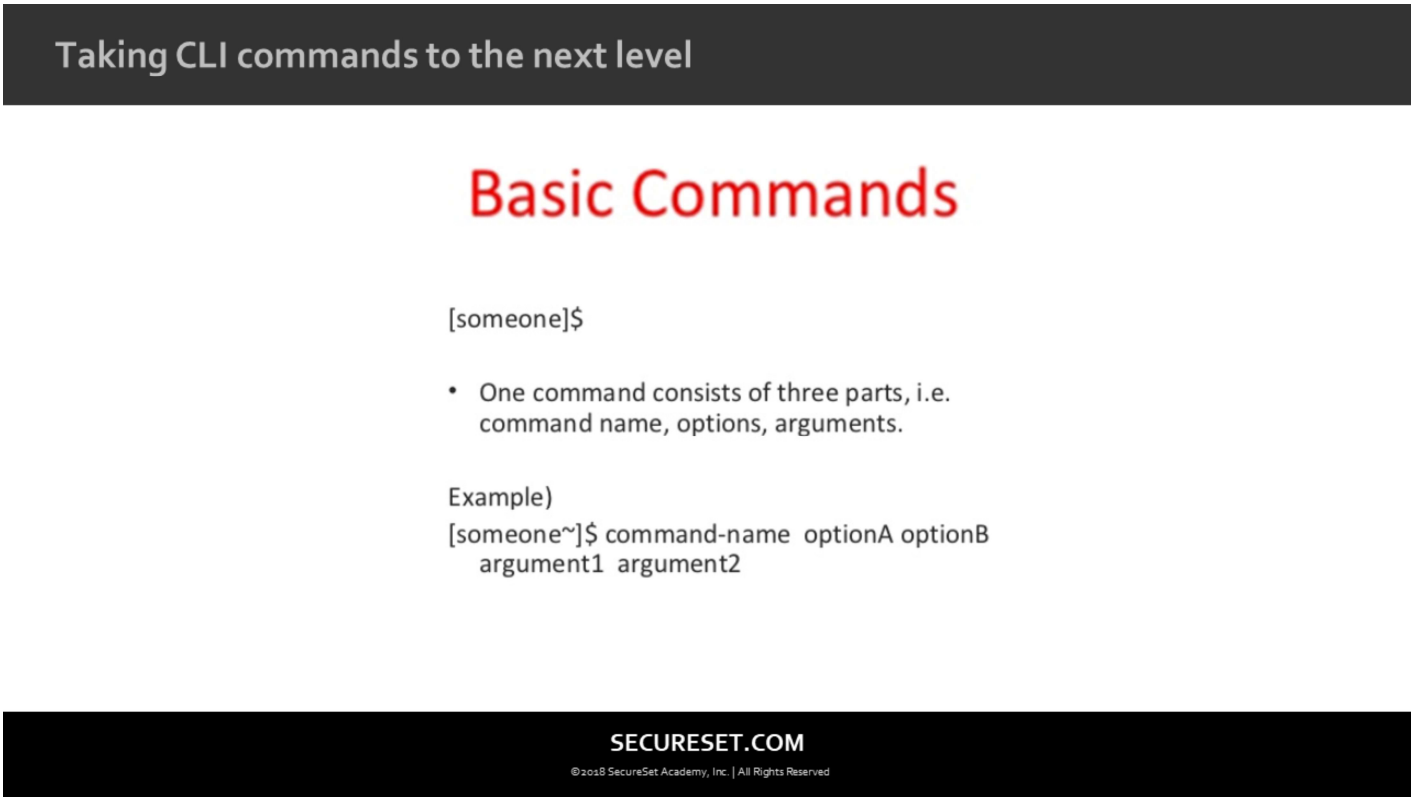
    a. **A simple command** consists of a sequence of words separated by spaces or tabs. The first word is taken to be the name of a command, and the remaining words are passed as arguments to the command.

    b. Before we delve deeper into bash, recall that a shell is a program that accepts and executes commands.

    c. It also supports programming constructs, allowing complex commands to be built from smaller parts.

    d. These complex commands, or scripts, can be saved as files to become new commands in their own right.

    e. Many commands on a typical Linux system are scripts.



a. **UNIX extends the power of commands by using special flags, arguments or switches.**

b. **These operators are one of the most powerful features of UNIX commands.**

c. Switches are usually preceded with a dash (-) and precede any filenames or other arguments on the command line.

d. Unlike the DOS (or Windows) command line, UNIX systems are case sensitive (upper and lower-case characters are considered different).

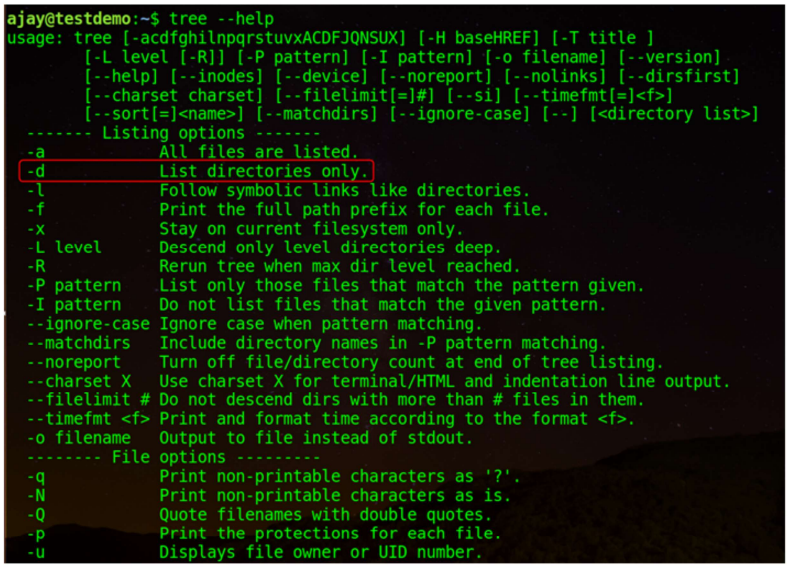
e. Nearly all command names and most of their command line switches will be in lowercase.

f. In this class all of the UNIX commands should be typed in lowercase

characters unless explicitly instructed otherwise.

g. UNIX systems have a hierarchical directory structure. This means that the hard disk area is divided into directories, much like a book is sub-divided into chapters and paragraphs. The directories form a tree-like structure, which simplifies the organization of the files on the system.

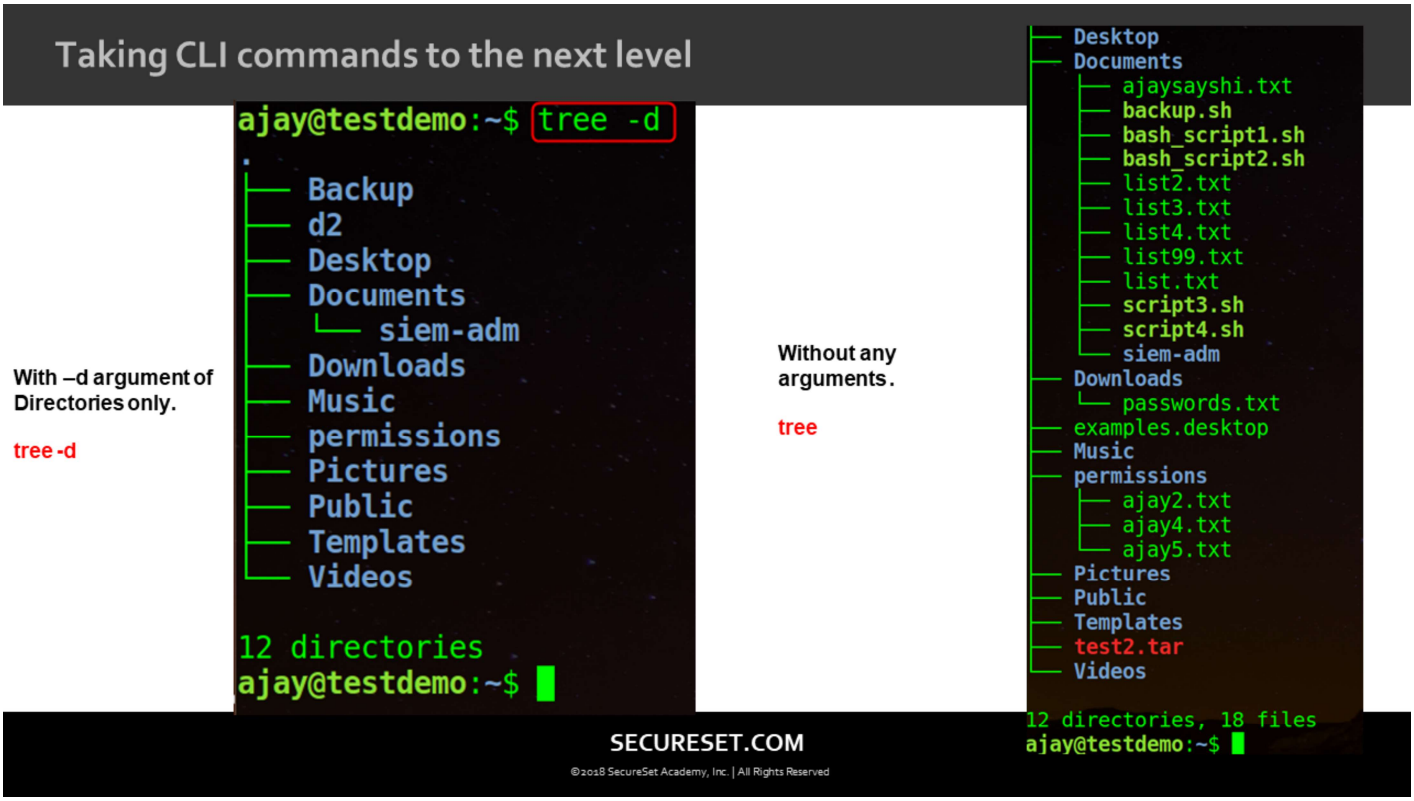
Taking CLI commands to the next level



SECURESET.COM

Please explain the tree command and how the switches / options / arguments allow you to manipulate the output.

Explain the - -help command.

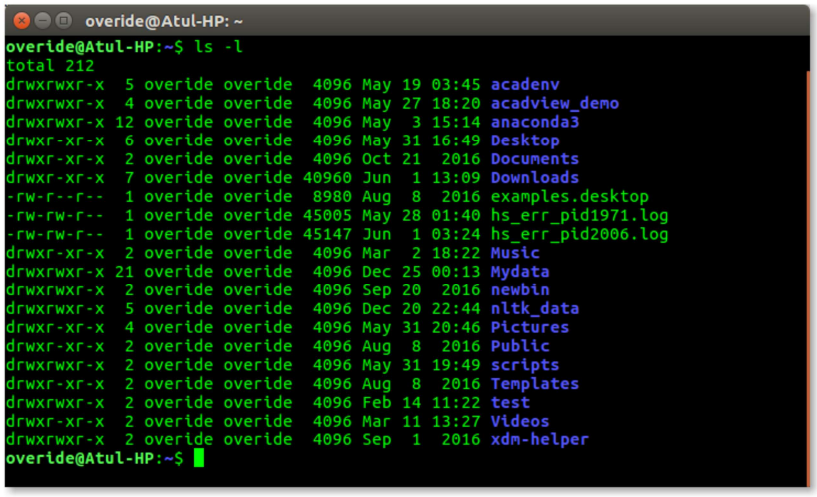


This slide shows what a single argument will do to the output of the TREE command.

(**TREE**) is not normally installed, it is easily installed.

It just gives a ANSI visualization of folders and files. Helpful to understand the file system, folders and files.

Shells part deux



1. **Shells** 2.0
    a. Every user has a unique username. When they logon to the system, they are placed in a HOME directory, which is a portion of the disk space reserved just for them.

    b. When you log onto a UNIX system, your main interface to the system is called the UNIX SHELL.

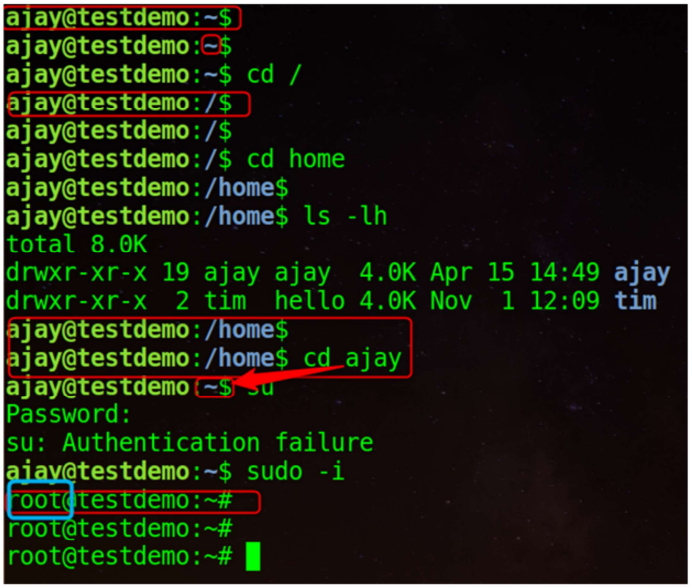
    c. This is the program that presents you with the dollar sign ($) prompt.

    d. This prompt means that the shell is ready to accept your typed commands.

    e. We know in Ubuntu Server or Desktop you will using one of the most standard UNIX shells called the **Bourne Shell**.

f. UNIX commands are strings of characters typed in at the keyboard. To run a command, you just type it in at the keyboard and press the ENTER key.

CLI deeper dive



SECURESET.COM

1. CLI START
    a. When you first log into a UNIX system, you are placed in your own personal directory space called your HOME directory.
        i. On most UNIX systems, the user HOME directories are located under the /home directory.

    b. Types of prompts

i. user@computername~$

ii. ajay@linux1:~$ Home Directory

iii. ajay@linux1:/$ Root of File System

iv. ajay@linux1:/home Home folder, but not my user home folder. <Know the structure! This is a trick>

v. root@linux1:/# If you are logged in as root or a superuser instead of $ you will get a #

1. The root user has considerable power, so use it with caution. When you have root privileges, most prompts include a trailing pound sign (#). Ordinary user

privileges are usually delineated by a different character, commonly a dollar sign ($). Your actual prompt may look different than the examples in this tutorial. Your prompt may include your user name, hostname, current directory, date, or time that the prompt was printed, and so on. We will discuss root and superusers shortly.

i. The shell's main function is to interpret your commands so you can interact with your Linux system.

ii. If a line contains a # character, then all remaining characters on the line are ignored. So, a # character may indicate a

comment as well as a root prompt. Which it is should be evident from the context.

iii. You need to quote strings, using either double quotes (") or single quotes (‘).

1. a **string** is any finite sequence of characters (i.e., letters, numerals, symbols and punctuation marks).

2. Bash uses white space, such as blanks, tabs, and new line characters, to separate your input line into tokens, which are then passed to your command.

3. Quoting strings preserves additional white space and makes the whole

string a single token.

1. CLI NAVIGATION
    a. **echo**
        i. The echo program displays text. It's a handy way to create customized output in your terminal.
        ii. echo is a fundamental command found in most operating systems. It is frequently used in scripts, batch files, and as part of individual commands; anywhere you may need to output text.
        iii. echo [SHORT-OPTION]... [STRING]...

    b. **pwd** (print working directory) – WHERE THE HECK AM I? =)
        i. Prints the current working directory on your screen. This is the directory where you are currently located. When you manipulate files and sub-directories, this is where they will (by default) be.
        ii. Here is an example of using pwd:
            1. pwd
            2. /homeb/bpowell

a. **cd** (change directory)

i. This command is used to change the current working directory.

ii. Here are some examples of using cd:

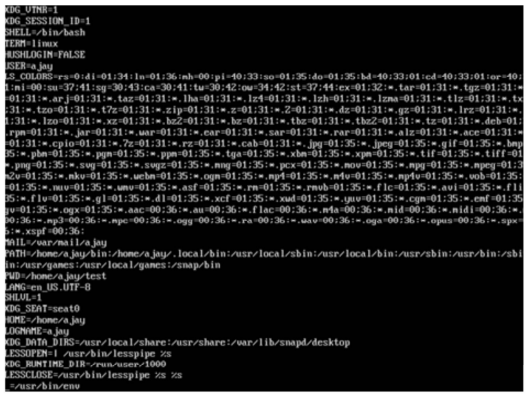
1. cd datafiles cd .. cd / cd $HOME

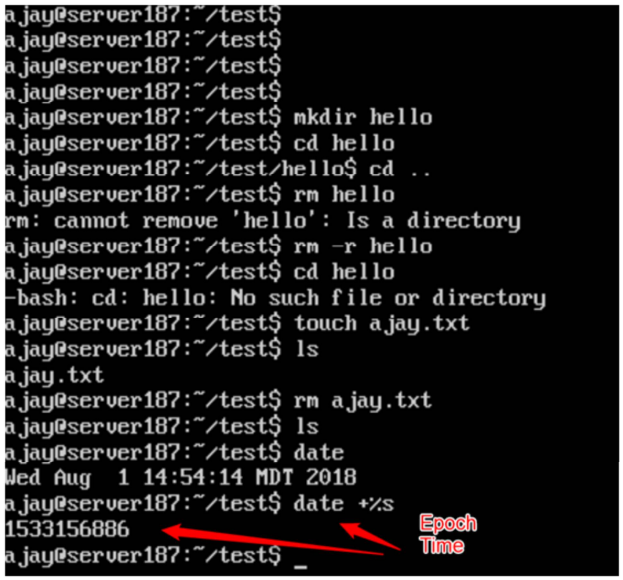
iii. The last example shows the use of a UNIX environment variable. The $HOME variable always contains the location of your HOME directory. (We will discuss BASH variables later in more detail)

iv. When an environment variable is used in a command, the contents of the variable and substituted for its name, so the above example would end up doing the same thing as if you typed in:

b. cd /homeb/bpowell

CLI Navigation and Basic Commands part deux





SECURESET.COM

a. **env**

i. You can print all of your environment variables by running the "env" command.

b. **mkdir** (make directory)

i. This command makes a sub-directory under the current working directory.

ii. mkdir junk

c. **rmdir** (remove directory)

i. This command removes (deletes) a sub-directory under the current working directory. The directory to be removed must be empty of all files and sub-directories.

ii. rmdir junk

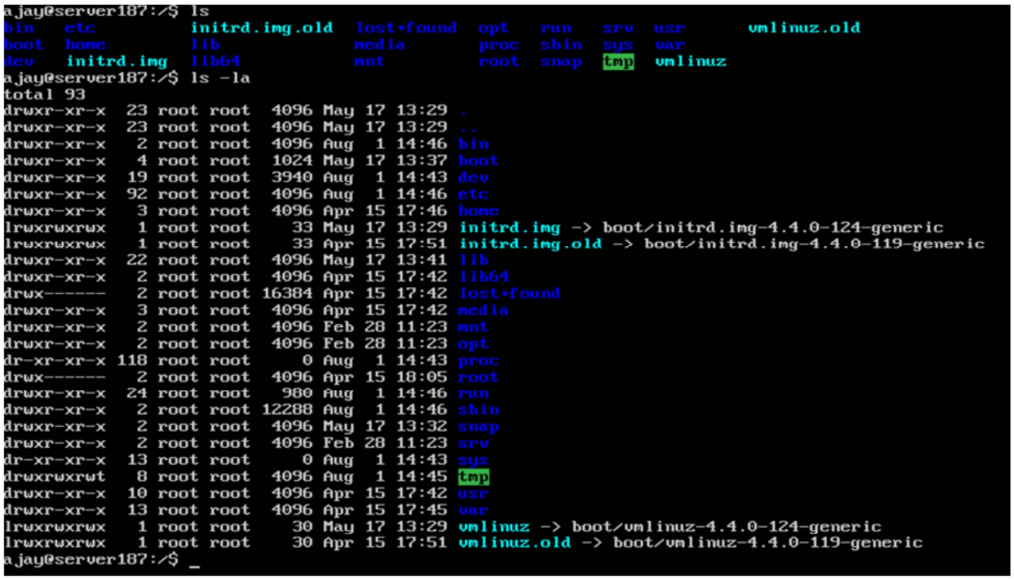
d. **touch**

i. touch changes file timestamps. It is also an easy way to create empty files.

a. date

i. date will display the time and date as well as time zone information.

CLI Navigation and Basic Commands part trois



SECURESET.COM

a. Manipulate files

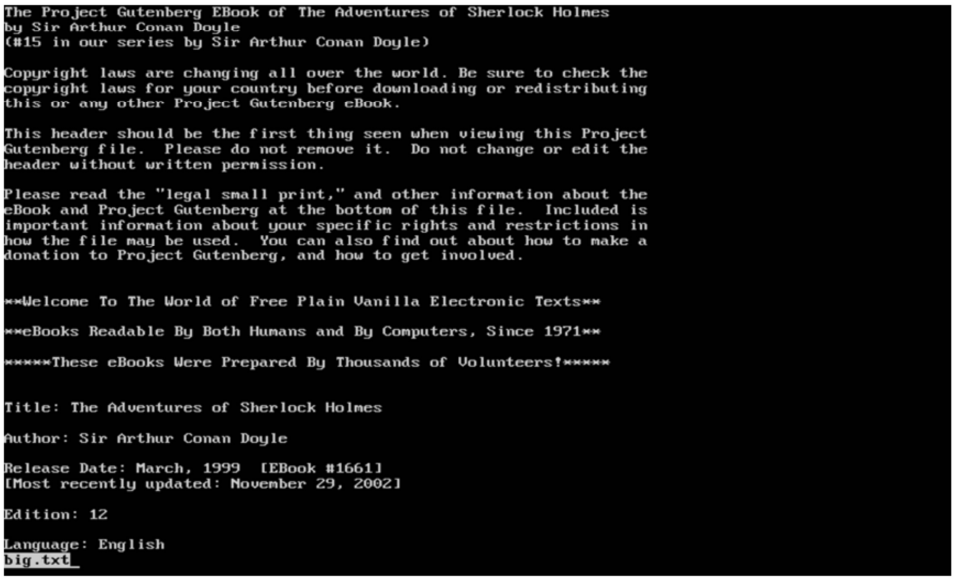
i. ls (list files)

1. This command is similar to DIR in DOS; it displays a list of all the files in the directory. New users don't have any files in their home directory.
2. Here is an example of using this command;
3. **ls**
4. You will note that the shell prompt reappears, and there is no file listing of the directory contents. This does not mean that there are no files stored there. Just like DOS, UNIX systems support hidden files.
5. Here is an example of using the command with switches:
   a. **ls -la**
6. The switch l stands for long listing, **and the a switch is for all files, including directories and hidden files.**
7. UNIX responds with the following listing of the directory

a. ls -la

b. total 6 -rw-rw-r-- 1 jmsmith staff 526 Apr 15 11:03 myletter

CLI Navigation and Basic Commands part quatre

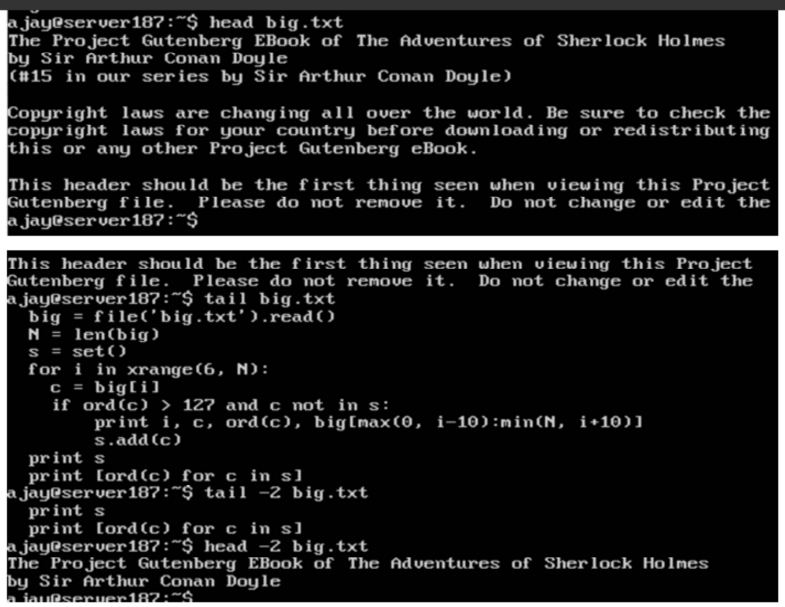


SECURESET.COM

i. **less** (page through a text file)
    1. This command allows you to view a text file without fear of accidentally modifying it, as you would with a text editor.
    2. **the key q allows you to exit**. Space bar to go from page to page.
    3. Example:
        a. **less /etc/hosts**

ii. **cat** (concatenate files)
    1. Print files to Screen.
    2. Note: if the file is very large or is not a plain text file, cat will try to print it to the screen anyway, sometimes with undesirable results. The less command is better suited for viewing files than the cat command.

CLI Navigation and Basic Commands part six



SECURESET.COM

i. head

1. This is used to view the first few lines of a file. It accepts a switch specifying the number of lines to view. The command
2. head -2 temp
3. would list the first 2 lines of the file temp on your screen.

ii. tail

1. This is used to view the last few lines of a file. It accepts a switch specifying the number of lines to view. The command
2. tail -2 temp
3. tail -f logfile.log
4. This forces the tail to follow the log, so it shows items being written to that file in real-time.

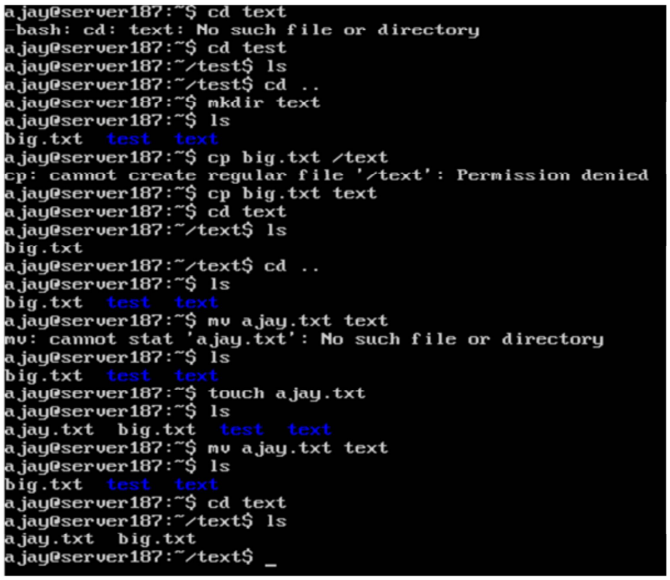
iii. diff

1. diff analyzes two files and prints the lines that are

different. Essentially, it outputs a set of instructions for how to change one file to make it identical to the second file.

2. diff file1.txt file2.txt

CLI Navigation and Basic Commands part cinq



SECURESET.COM

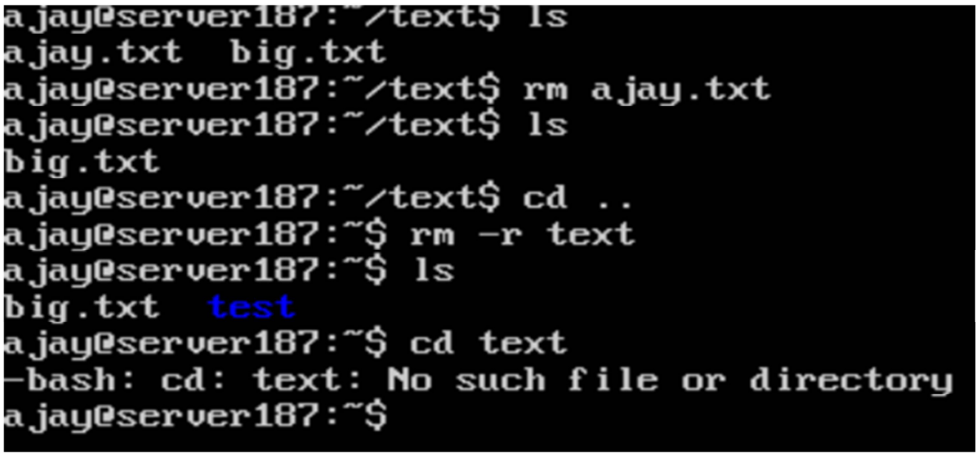
i. **cp** (copy files)
   1. This command stands for copy, and is used for copying one file to another.
   2. Example:
      a. cp .profile temp2
   3. This copies the file .profile to another called temp2. If temp2 had already existed, its previous contents would've been erased.
   4. Files can also be copied to another directory. The command
   5. **cp * /usr/tmp**
      a. would copy all the files in the current directory to the directory /usr/tmp.

ii. **mv** (move files)
   1. The mv command is used for moving or renaming files.
   2. Example:

a. mv temp temp2

1. This renames the file temp to temp2.
2. As an example (do not type this), the command:
3. **mv temp2 /tmp**
4. would move the file temp2 into the directory /tmp (it would no longer appear in your home directory).

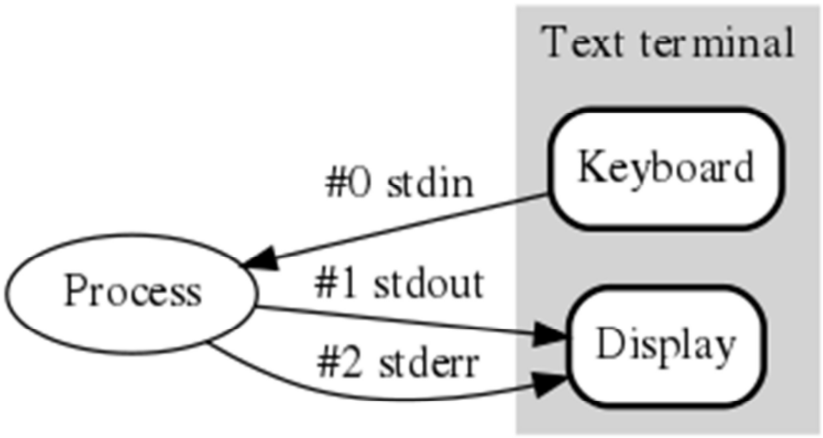
The subtle science and exact art of deleting things



i. **rm** (remove files)

1. The rm utility is used for erasing files and directories.
    a. Example:
        i. rm temp2
2. This removes the file. Once a file is removed, it cannot be restored. To cover situations where mistakes might occur, a switch -i appended to this command will request a Yes or No response before deleting the file.
3. Example:
    a. rm -i temp1
4. NOTE that switches are written before the filenames. Answer Y to the prompt so that temp1 is removed.

What is a shell?



SECURESET.COM

1. **Shells** have some built-in commands, such as cd, break, and exec.

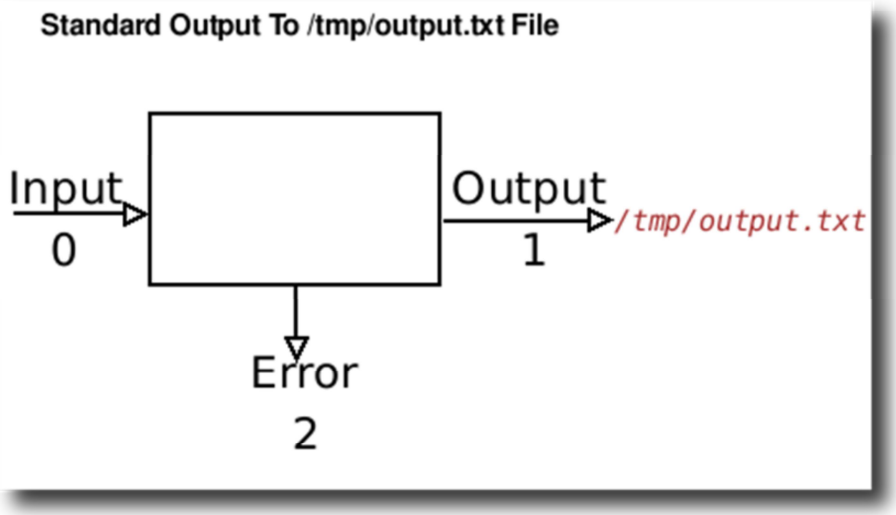
    a. Other commands are external. We will review most of the commands shortly, however we need to address I/O first.

2. **Shells** also use three standard I/O streams:

    a. the standard input stream, which provides input to commands is called **stdin**
    b. the standard output stream, which displays output from commands is called **stdout**
    c. the standard error stream, which displays error output from commands is called **stderr**
    d. Input streams provide input to programs, usually from terminal keystrokes. Output streams print text characters, usually to the terminal. The terminal was originally an ASCII typewriter or display terminal, but it

is now more often a window on a graphical desktop.

Standard I / O - Inputs and Outputs



SECURESET.COM

i. In order to understand about **Redirecting** operators in Linux we should know how we communicate with a computer.

ii. When we are communicating with a computer there should be a way to do it and this is achieved by STDIN (0), STDOUT (1), STDERR (2) file descriptors.
    i. These file descriptors are assigned with a number as shown below:
        i. STDIN –> 0
        ii. STDOUT –> 1
        iii. STDERR –> 2

Standard I / O - Inputs and Outputs part deux

SECURESET.COM

1. **STDIN**: stands for **STandarD INput**.
    1. By using this we can give an input to the computer to do some task. Whatever device we used to give input to a computer will come under STDIN.
    2. A STDIN can be A keyboard A mouse A scanner A floppy A CD/DVD ROM A touch screen A barcode reader or a card reader.

2. **STDOUT**: The abbreviation is **STandarD OUTput**.
    1. By using this we can see the output from a computer for the it just processed or executed. Whatever device we used to get the output from a computer will come under this STDOUT.
    2. A STDOUT can be A monitor A speaker Or a printer

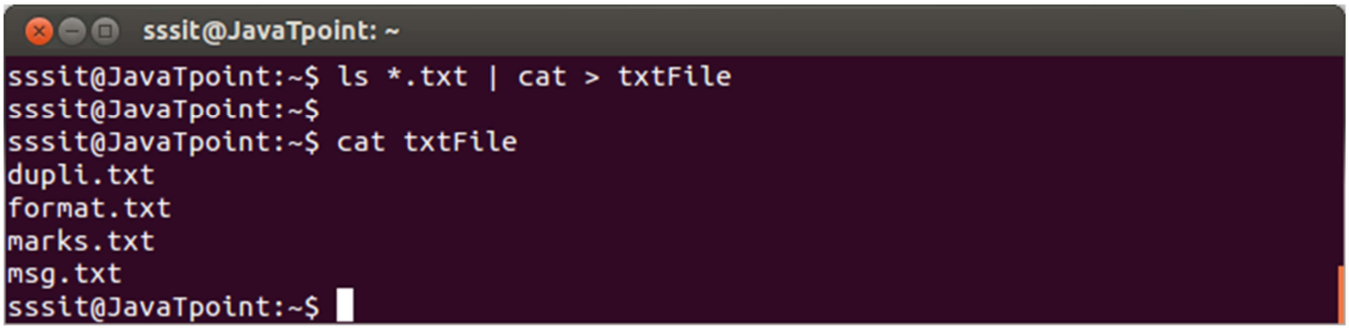
3. **STDERR**: This is abbreviated as **STandarD ERRor**.
    1. By using this, A computer can communicate with user to give him warning/error etc. that something went wrong when executing a task.

2. A STDERR can be A monitor A printer A log file A LED indicator Or a speaker.

Redirection – Commands Introduction

Redirection Operator	Explanation	Example
>	The greater-than sign is used to send to a file, or even a printer or other device, whatever information from the command would have been displayed in the Command Prompt window had you not used the operator.	**assoc > types.txt**
>>	The double greater-than sign works just like the single greater-than sign but the information is appended to the end of the file instead of overwriting it.	**ipconfig >> netdata.txt**
<	The less-than sign is used to read the input for a command from a file instead of from the keyboard.	**sort < data.txt**
|	The vertical pipe is used to read the output from one command and use if for the input of another.	**dir | sort**

Redirection



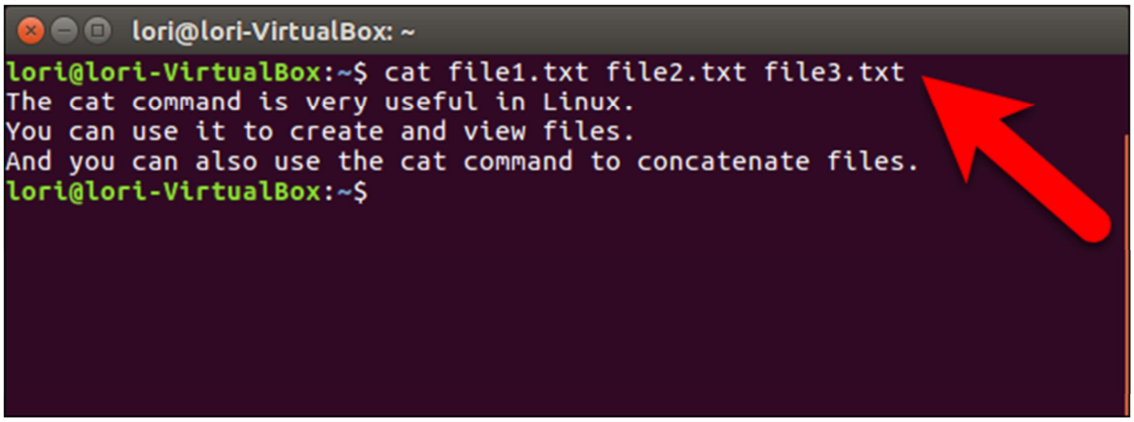
SECURESET.COM

1. Why we require redirecting operators?

    1. We require redirecting operators in some situations where our standard communication with computer will not meet our requirement.

2. For example, sometimes we want to store an error which is popping on a screen to a file for future reference.
3. At that time, we can use one of these redirecting operators to change the default way of communication between a user and a server.

Shells part deux



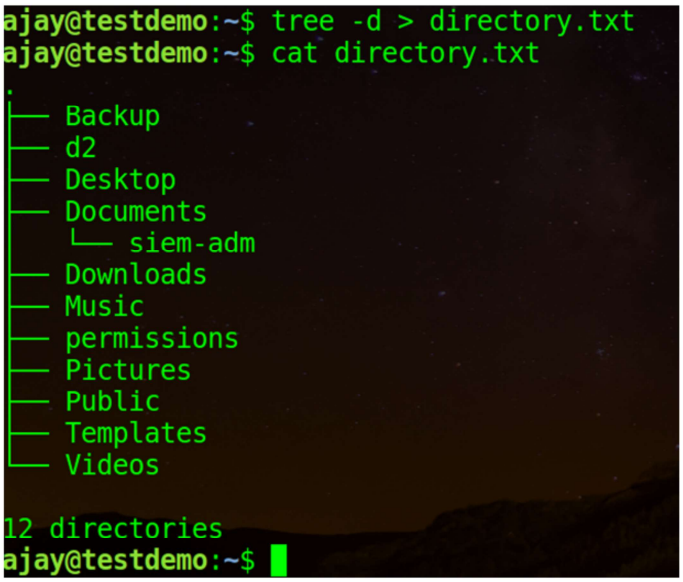
SECURESET.COM

The cat (short for “concatenate“) command is one of the most frequently used command in Linux/Unix like operating systems.

cat command allows us to

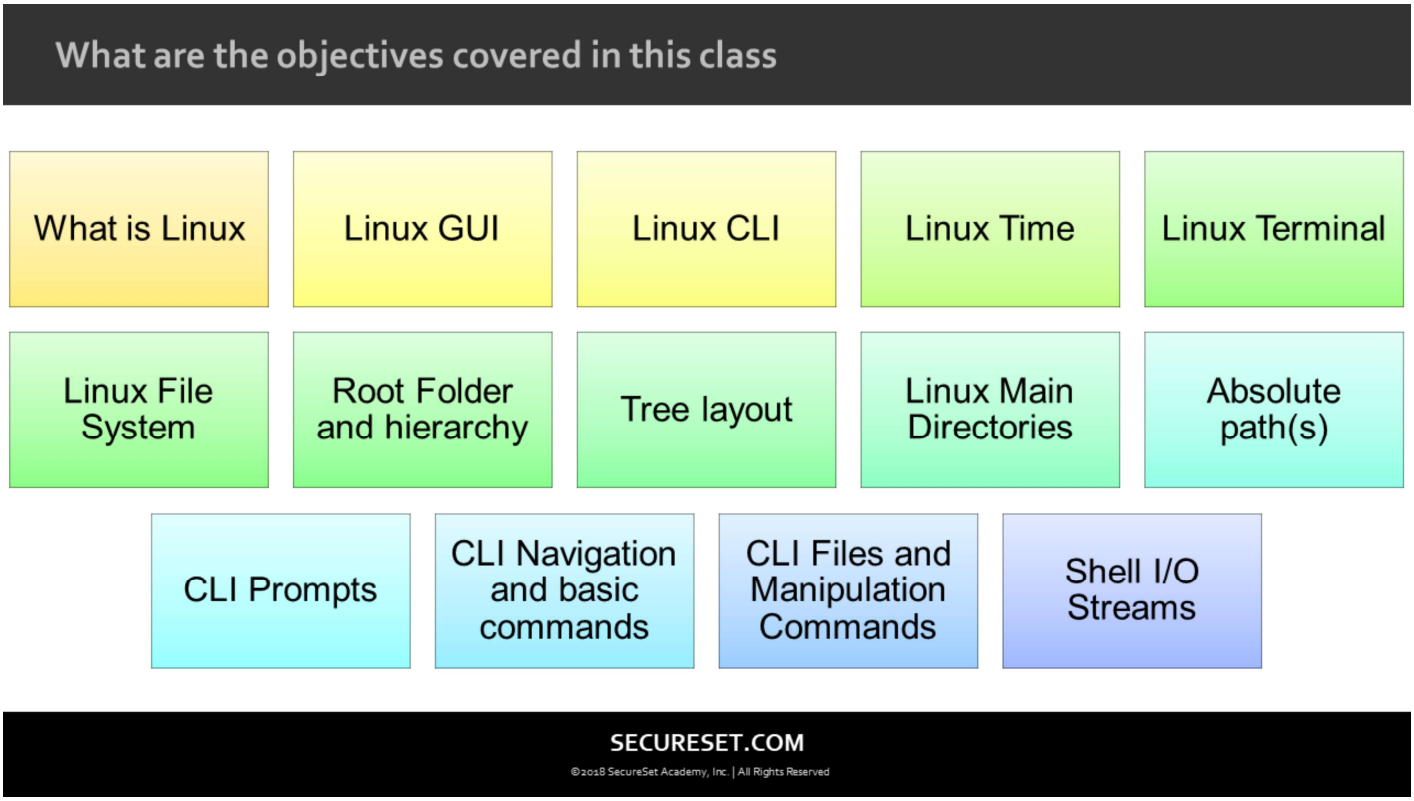
- create single or multiple files
- view contain of file
- concatenate files
- redirect output in terminal or files

Shells part deux



SECURESET.COM

Quick demo of using a redirection of the **TREE** command we used earlier to save the output to a file. Then we use the **cat** command to display what we saved in the file.



Here are a list of objectives that we aimed to cover in this SecureSet Systems 2 class.

You should feel comfortable with these topics in a conversational and assessment manner. We encourage you to continue studying on your own.

Amongst these above topics are a very large and numerous amount of jobs. Plus it is the foundation for the rest of your education here at SecureSet.