

# Objectives

Friday, August 11, 2017 1:43 PM



Linux  
Essentials...



Quizlet  
Review



Linux Basic  
Commands



Commands



LPI Book



Objectives

<https://www.theurbanpenguin.com/>

<https://www.canonical.com/>

Support for Open Source Software



Certificatio  
n



Munisteri\_L  
inux\_Esse...

<https://www.youtube.com/watch?v=TILMEERsu60&list=PLtGnc4I6s8dssa8hF4yMTAa4BrSJCSwux&index=1>

1.1 Linux Evolution and Popular Operating Systems	Weight: 2																									
Key Knowledge Areas:																										
• Open source Philosophy	When we call software “free,” we mean that it respects the <a href="#">users' essential freedoms</a> : <b>the freedom to run it, to study and change it, and to redistribute copies with or without changes.</b> This is a matter of freedom, not price, so think of “free speech,” not “free beer.”																									
• Distributions	<p>Many different Linux distributions are available, each consisting of a Linux kernel along with a set of utilities and configuration files. Distribution maintainers also offer patch kernels—that is, they make small changes to fix bugs or add features. A Linux distribution is a collection of many programs that may use different individual licenses. No one license takes priority over the others.</p> <table><tr><th>Distribution</th><th>Availability</th><th>Package format</th><th>Release cycle</th><th>Administrator skill requirements</th></tr><tr><td>Arch</td><td>Free</td><td>pacman</td><td>Rolling</td><td>Expert</td></tr><tr><td>CentOS</td><td>Free</td><td>RPM</td><td>Approximately 2-year</td><td>Intermediate</td></tr><tr><td>Debian</td><td>Free</td><td>Debian</td><td>2-year</td><td>Intermediate to expert</td></tr><tr><td>Fedora</td><td>Free</td><td>RPM</td><td>Approximately 6-month</td><td>Intermediate</td></tr></table>	Distribution	Availability	Package format	Release cycle	Administrator skill requirements	Arch	Free	pacman	Rolling	Expert	CentOS	Free	RPM	Approximately 2-year	Intermediate	Debian	Free	Debian	2-year	Intermediate to expert	Fedora	Free	RPM	Approximately 6-month	Intermediate
Distribution	Availability	Package format	Release cycle	Administrator skill requirements																						
Arch	Free	pacman	Rolling	Expert																						
CentOS	Free	RPM	Approximately 2-year	Intermediate																						
Debian	Free	Debian	2-year	Intermediate to expert																						
Fedora	Free	RPM	Approximately 6-month	Intermediate																						

Distribution	Availability	Package format	Release cycle	Administrator skill requirements
Gentoo	Free	ebuild	Rolling	Expert
Mint	Free	Debian	6-month	Novice to intermediate
openSUSE	Free	RPM	8-month	Intermediate
Red Hat Enterprise	Commercial	RPM	Approximately 2-year	Intermediate
Scientific	Free	RPM	Approximately 6-month	Intermediate to expert
Slackware	Free	tarballs	Irregular	Expert
SUSE Enterprise	Commercial	RPM	2–3 years	Intermediate
Ubuntu	Free	Debian	6-month	Novice to intermediate

- Embedded Systems

Such devices typically require little or no administrative work from users. These devices have fixed basic configurations and guided setup tools to help inexperienced users set critical basic options, such as network settings and time zones (mobile phones, e-readers, DVRs, etc)

**The following is a partial list of the used files, terms and utilities:**

- Android

Its user interface is similar to that of other smartphones, but underneath lies a Linux kernel and a significant amount of the same Linux infrastructure that you'll find on a PC.

- Some tablets and e-book readers, for instance, run Android.

- Debian, Ubuntu (LTS)

- Popular distribution packages.
- Ubuntu releases long-term support (LTS) versions in April of even numbered years. Ubuntu come out every six months, like clockwork.

- CentOS, openSUSE, Red Hat

- CentOS: based on Red Hat Enterprise Linux (RHEL). Red Hat will sell support for this OS distribution
- openSUSE: Serious platform, brief life cycle, freely available.
- Red Hat: established in 1993, Linux was purchased in '94 by Marc Ewing, and called his Linux distribution Red Hat Linux. Later went enterprise, and sells RHEL per server license

- Linux Mint, Scientific Linux

## 1.2 Major open source Applications

Weight 2

### Key Knowledge Areas:

- Desktop Applications

- The X Window System GUI (X for short)
- Desktop environment, such as GNOME, KDE, Xfce, or Unity
- Web browser, such as Mozilla Firefox
- Email client, such as Mozilla Thunderbird or Evolution
- Graphics editor, such as the GNU Image Manipulation Program (GIMP)
- Office suite, such as Apache OpenOffice.org or the similar LibreOffice

- Server Applications

- Web servers, such as Apache
- Email servers, such as Sendmail and Postfix

	<ul style="list-style-type: none"> <li>• Email servers, such as Sendmail and Postfix</li> <li>• Databases, such as MySQL</li> <li>• File servers, such as the Network File System (NFS) or Samba</li> <li>• Print servers, such as the Common Unix Printing System (CUPS) or Samba</li> <li>• Domain Name System (DNS) servers, such as the Berkeley Internet Name Domain (BIND)</li> <li>• Dynamic Host Configuration Protocol (DHCP) servers, such as the Internet Systems Consortium's (ISC's) dhcpd</li> <li>• Time servers, such as the Network Time Protocol (NTP)</li> <li>• Remote login servers, such as Secure Shell (SSH) or Virtual Network Computing (VNC)</li> </ul>
• Development Languages	<ul style="list-style-type: none"> <li>• GNU compiler family: C, C++, Objective C, Java, Fortran and Ada</li> <li>• Scripting languages such as Perl, Python, Tcl/Tk, Ruby, Lua, or PHP are supported</li> <li>• Less common languages such as Lisp, Scheme, Haskell, Prolog, or Ocaml</li> </ul>
• Package Management Tools and repositories	<ul style="list-style-type: none"> <li>• Package Management: packages the computer is aware of and which of those are currently installed</li> <li>• Repositories: servers containing packages, find out which of the packages on your computer are out of date because the distribution offers newer versions.</li> <li>• CentOS / Red Hat / Fedora : <b>yum</b></li> <li>• SUSE / openSUSE : <b>zypper</b></li> <li>• Debian / Ubuntu : <b>apt</b></li> </ul>
<b>Terms and Utilities:</b>	
• OpenOffice.org, LibreOffice, Thunderbird, Firefox, GIMP	<ul style="list-style-type: none"> <li>• Oracle stopped supporting commercial development of the project and donated it to the Apache group. The official name is currently Apache OpenOffice . It provides six applications: Writer (word processor), Calc (spreadsheet), Impress (presentation), Base (database), Draw (vector graphics), and Math (equation editor).</li> <li>• This office suite was created as a fork of the older pre ApacheOpenOffice.org. It's becoming the most popular office suite in Linux. It provides six applications: Writer (word processor), Calc (spreadsheet), Impress (presentation), Base (database), Draw (vector graphics), and Math (equation editor).</li> <li>• Thunderbird is email an client that's closely associated with the Firefox web browser.</li> <li>• The GNU Image Manipulation Program (GIMP) (<a href="http://www.gimp.org">www.gimp.org</a>) is a still-image manipulation program similar in broad strokes to Adobe Photoshop.</li> </ul>
• Apache HTTPD, NGINX, MySQL, NFS, Samba	<ul style="list-style-type: none"> <li>• Apache HTTPD: server is part of the prevalent <b>Linux, Apache, MySQL, PHP (LAMP)</b> stack for web applications. It's a webserver, very popular, stable, and secure.</li> <li>• Nginx (pronounced Engine X ) web server is a newcomer to the market. Nginx can retrieve resources on behalf of a client from one or more servers, as well as operate as a mail server.</li> <li>• MySQL: freely available relational database servers. Best used for web sites</li> <li>• NFS: Unix environment to Samba and allows other Linux and Unix machines on the network access to a Linux server's disks.</li> <li>• Samba: turns a Linux machine into a server for Windows clients which makes disk space and printers available to all Windows machines on the network</li> </ul>
• C, Java, Perl, shell, Python, Samba	<ul style="list-style-type: none"> <li>• <b>C</b> is the most important compiled language for Linux <ul style="list-style-type: none"> <li>• Fairly efficient code, but it's also easy to write buggy programs in C because it lacks some error-checking features</li> <li>• Files typically have file names that end in .c or .h <ul style="list-style-type: none"> <li>◦ .c files are the main source code files</li> </ul> </li> </ul> </li> </ul>

	<ul style="list-style-type: none"> <li>○ .h files are header files, which contain short definitions of the functions in the .c files, for reference by other files in a program</li> <li>○ compiled with the gcc program</li> <li>• <b>C++</b> extension to C that adds object-oriented features, meaning that greater emphasis is given to data structures and their interactions than to the procedures used to control the flow of the program. <ul style="list-style-type: none"> <li>• source code files can have file names that end in .cc, .cpp, .cxx, or .c++, with header files ending in .h, .hh, .hpp, .hxx, or .h++.</li> <li>• C++ is generally compiled with the g++ program</li> </ul> </li> <li>• <b>Java</b> was created by Sun Microsystems (now owned by Oracle) as a cross platform language that's somewhere between being compiled and interpreted. It's become popular as a language for small applications delivered via websites, Although the Linux kernel is mostly written in C, parts of it are written in assembly language. Although some other programs are Java based as well. Java source code usually has a name that ends in .java.</li> <li>• <b>Perl</b> This interpreted language is designed for easy manipulation of text, but it's a general-purpose language that can be used for many other tasks as well. Perl programs typically have file names that end in .pl, .pm, or .t.</li> <li>• <b>PHP</b>: Hypertext Preprocessor, or PHP (a recursive acronym), language was created for use on web servers in order to generate dynamic content—that is, content that varies depending on the user, the time of day, or some other criterion. PHP is an interpreted language, and it requires a PHP aware web server, such as Apache. Given such a server and appropriate configuration, a website can support user logins, shopping carts, different content based on users' locations, and so on. PHP files most often have names that end in .php, although several variants are common.</li> <li>• <b>Python</b> This interpreted language makes code readability a major goal. It supports (but does not require) object orientation. It's often used for scripting purposes, but it can be used to write more-complex programs, too. Python programs often use .py file name extensions, although several variants of this are common too.</li> <li>• <b>Shell Scripting</b> Most Linux text-mode shells—the programs that enable entirely keyboard-based use of the computer—provide their own interpreted languages. Of these, the Bourne Again Shell (Bash) is the most common, so Bash scripting is quite common. Many of the files that control the Linux startup process are in fact Bash scripts. Such scripts frequently have no unique file name extension, although some use a .sh extension.</li> </ul>
• dpkg, apt-get, rpm, yum	<ul style="list-style-type: none"> <li>• <b>dpkg</b> A low-level package tool used as the foundation of Debian-based family of PMS tools. Used directly to <ul style="list-style-type: none"> <li>• Install, manage, and remove software packages.</li> <li>• Cannot download software packages from the repositories.</li> </ul> </li> <li>• <b>apt-get</b> This is a text-mode tool for the Debian PMS. <ul style="list-style-type: none"> <li>• Install from repositories and remove software packages from your local Linux system. In addition, you can perform package upgrades for individual packages, all of the packages on your system, or your entire distribution.</li> <li>• Need to use the apt-cache text-mode tool for determining various pieces of information concerning software packages.</li> </ul> </li> <li>• <b>rpm</b> The rpm tool is also a low-level package tool similar in function to the dpkg utility. However, it is used as the foundation of the Red Hat Linux package management system. Though you can use rpm to manage packages, it's best to use a higher-level PMS utility.</li> <li>• <b>yum</b> This is a text-mode tool for the Red Hat PMS. It is used on distributions, such as Red Hat Enterprise Linux (RHEL), Fedora, and CentOS. <ul style="list-style-type: none"> <li>• Install from repositories, remove software packages from your local Linux system, upgrade packages, and so on.</li> </ul> </li> </ul>

	<ul style="list-style-type: none"> <li>• Determining various pieces of information concerning packages and their management, such as displaying a list of the PMS's configured repositories.</li> </ul>
<b>1.3 Understanding open source Software and Licensing</b>	Weight: 1
<b>Key Knowledge Areas:</b>	
• Licensing	Legal document that claims to modify the rights granted by copyright law.
• Free Software Foundation (FSF), open source Initiative (OSI)	<ul style="list-style-type: none"> <li>• FSF advocates what it calls free software</li> <li>• Freedom to do things you want to do with the software, not the price of the software. <ul style="list-style-type: none"> <li>• A common phrase used to make this distinction clear is “free as in speech, not free as in beer.”</li> </ul> </li> <li>• Open source is a development method for software that harnesses the power of distributed <b>peer review</b> and transparency of process. <ul style="list-style-type: none"> <li>• The promise of open source is better quality, higher reliability, more flexibility, lower cost, and an end to predatory vendor lock-in.</li> </ul> </li> </ul>
<b>Terms and Utilities:</b>	
• GPL, BSD, Creative Commons	<p><b><u>General Public License (GPL):</u></b> license used by the Linux kernel, grants you the right to redistribute the software, including both the source code and binaries. Changes you make to the software should be contributed back to the project.</p> <p><b><u>BSD</u></b> The Berkeley Software Distribution License is currently a 3 clause agreement that allows for redistribution of software provided it includes the original copyright notice.</p> <p><b><u>Creative Commons:</u></b> The creative commons license allow for content creators to be able to collaborate on projects to improve the quality and diversity of content.</p>
• Free Software, open source Software, FOSS, FLOSS	<p>Equally as important as the licenses are the organizations that support the ethos of “free” software and for us, it is important to recognize their work and support them where we can, be it through membership, donations, purchases or simply promoting these organizations with friends and colleagues.</p> <ul style="list-style-type: none"> <li>• The free software foundation: <a href="http://www.fsf.org">www.fsf.org</a></li> <li>• The Open Source Initiative: <a href="http://opensource.org">opensource.org</a></li> </ul>
• Open source business models	<p><b>Services and Support</b> The product itself can be open source, and even given away for free, while the company sells services and support, such as training and a technical support phone line. For instance, a game might be open source but require a subscription to an online service to provide a full set of features.</p> <p><b>Dual Licensing</b> A company can create two versions of the product: one version is completely open source, and another adds features that are not available in the open source version. The open source version is then akin to the free samples that supermarkets often provide—it's a way to draw in paying customers.</p> <p><b>Multiple Products</b> The open source product may be just one offering from the company, with revenue being generated by other product lines. These other product lines could be other software or some other product, such as manuals.</p> <p><b>Open Source Drivers</b> A special case of the preceding one is that of hardware vendors. They might opt to release drivers, or perhaps even hardware-specific applications, as open source as a way to promote their hardware.</p>

	<p><b>Bounties</b> Bounties are a crowdfunding method. Users can drive open source creation by offering to pay for new software or new features in existing software. Sites such as FOSS Factory ( <a href="http://www.fossfactory.org">www.fossfactory.org</a> ) and Bounty source ( <a href="http://www.bountysource.com">www.bountysource.com</a> ) can help bring together users, each of whom individually might not be able to offer enough money to motivate development, to entice programmers to write the desired code. With bounties, the programmer who completes the project first is allowed to collect the project's accumulated funds.</p> <p><b>Donations</b> Many open source projects accept donations to help fund development. Although this isn't a commercial funding model in the usual sense, it does help fund the operations of organizations such as the FSF.</p>
<b>1.4 ICT Skills and Working in Linux</b>	
<b>Key Knowledge Areas:</b>	
<ul style="list-style-type: none"> <li>• Desktop Skills</li> </ul>	<p><b>Desktop Menus</b> Many desktop environments provide menus along a top, bottom, or side edge of the screen. One or more items in these menus can give you access to a preselected set of applications.</p> <p><b>Desktop Icons</b> Some desktop environments enable you to place icons in the main area of the desktop. Clicking or double-clicking these icons then launches the applications that they represent. This approach generally requires customization. Some default configurations place a few applications in the main desktop area.</p> <p><b>Panels</b> Some desktop environments provide panels, typically located on the sides of the screen, in which icons for common applications appear. Unity uses such a configuration by default, as does GNOME 3 (a version of the GNOME desktop environment)—although in the case of GNOME 3, the panel appears only when you click the Activities item in the upper-left corner of the screen.</p> <p><b>Context Menus</b> You can sometimes right-click in an unused part of the screen to obtain a context menu with a variety of options, which may include the option to run programs.</p> <p><b>Searching for Programs</b> Some desktop environments, such as GNOME 3, provide a prominent search feature that you can use to find a program by name. Typically, you type part of a program's name, and programs whose names match appear in a list. You can then select the program that you want to run from that list.</p>
<ul style="list-style-type: none"> <li>• Getting to the Command Line</li> </ul>	<p>You can launch a program called a terminal , which provides a text mode user interface inside a window. You can then run either text-mode or GUI programs by typing their file names in this window.</p>
<ul style="list-style-type: none"> <li>• Industry uses of Linux, Cloud Computing and Virtualization</li> </ul>	<p>Cloud computing is the storage of computer software and/or data over the Internet, rather than locally storing it on your computer. Cloud represents the Internet and computing represents what you are doing over the Internet.</p>
<b>2.1 Command Line Basics</b>	
<b>Key Knowledge Areas:</b>	
<ul style="list-style-type: none"> <li>• Basic shell</li> </ul>	<p>A Linux command line, or shell as it's more properly called, is a program. You can start a shell in a GUI window called a terminal program or you can log in to the computer locally via a text-mode console.</p>
<ul style="list-style-type: none"> <li>• Command line syntax</li> </ul>	<p>Arguments or options a command can use</p>
<ul style="list-style-type: none"> <li>• Variables</li> </ul>	<p>A variable is a placeholder in a script for a value that will be determined when</p>

	the script runs.
• Globbing	A wildcard is a symbol or set of symbols that stands in for other characters. You can use wildcards to refer to files. (Using wildcards is also sometimes called globbing.)
• Quoting	<p>The preceding command illustrates another feature that you may need to use: shell quoting. Because the shell uses certain characters, such as the vertical bar (   ) and the asterisk ( * ), for its own purposes, you must enclose certain regular expressions in quotes.</p> <ul style="list-style-type: none"> <li>• <code>\$ grep -E "(games mail).*nologin" /etc/passwd</code></li> </ul>
<b>Terms and Utilities:</b>	
• Bash	The default shell in most Linux distributions is the Bourne Again Shell Bash or bash ), which is based on an older shell called the Bourne Shell.
• echo	You can pass various options to echo or just a string to be shown to the user. <b>echo “Press the Enter key”</b>
• history	<p>Bash remembers the recent commands that you’ve typed, and you can use this fact to save yourself some effort if you need to type a command that’s similar to one that you’ve typed recently.</p> <p>The command history itself can be used to display and clear the command line history for a given user, each user has their own unique .bash_history file which is created in the user’s home directory.</p> <ul style="list-style-type: none"> <li>• history without any arguments displays the users’ history</li> <li>• history 3 show the last 3 lines of the users’ history</li> <li>• history -c will clear the users’ history</li> <li>• wistory -w -- write history to file</li> <li>• HISTCONTROL="" -- list each duplicate entry</li> <li>• HISCONTROL="erasedups" -- shows multiple entries once</li> <li>• .bash_history -- writing to the file</li> </ul>

Keystroke	Effect
Up arrow	Retrieves the previous entry from the command history.
Down arrow	Retrieves an earlier entry bypassed when using the up arrow.
Left arrow	Moves the cursor left one character.



	<table> <tr> <th>Keystroke</th><th>Effect</th></tr> <tr> <td>Right arrow</td><td>Moves the cursor right one character.</td></tr> <tr> <td>Ctrl+A</td><td>Moves the cursor to the start of the line.</td></tr> <tr> <td>Ctrl+E</td><td>Moves the cursor to the end of the line.</td></tr> <tr> <td>Delete key</td><td>Deletes the character under the cursor.</td></tr> <tr> <td>Backspace key</td><td>Deletes the character to the left of the cursor.</td></tr> <tr> <td>Ctrl+T</td><td>Swaps the character under the cursor with the one to the left of the cursor.</td></tr> <tr> <td>Ctrl+X and then Ctrl+E</td><td>Launches a full-fledged editor on the current command line.</td></tr> <tr> <td>Ctrl+R</td><td>Searches for a command. Type a few characters, and the shell will locate the latest command to include those characters. You can search for the next-most-recent command to include those characters by pressing Ctrl+R again.</td></tr> </table>	Keystroke	Effect	Right arrow	Moves the cursor right one character.	Ctrl+A	Moves the cursor to the start of the line.	Ctrl+E	Moves the cursor to the end of the line.	Delete key	Deletes the character under the cursor.	Backspace key	Deletes the character to the left of the cursor.	Ctrl+T	Swaps the character under the cursor with the one to the left of the cursor.	Ctrl+X and then Ctrl+E	Launches a full-fledged editor on the current command line.	Ctrl+R	Searches for a command. Type a few characters, and the shell will locate the latest command to include those characters. You can search for the next-most-recent command to include those characters by pressing Ctrl+R again.
Keystroke	Effect																		
Right arrow	Moves the cursor right one character.																		
Ctrl+A	Moves the cursor to the start of the line.																		
Ctrl+E	Moves the cursor to the end of the line.																		
Delete key	Deletes the character under the cursor.																		
Backspace key	Deletes the character to the left of the cursor.																		
Ctrl+T	Swaps the character under the cursor with the one to the left of the cursor.																		
Ctrl+X and then Ctrl+E	Launches a full-fledged editor on the current command line.																		
Ctrl+R	Searches for a command. Type a few characters, and the shell will locate the latest command to include those characters. You can search for the next-most-recent command to include those characters by pressing Ctrl+R again.																		
• PATH env variable	\$PATH is an environment variable, and it is called the PATH env variable																		
• export	One special type of variable is an environment variable , which is assigned and accessed just like a shell script variable. The difference is that the script or command that sets an environment variable uses Bash’s export command to make the value of the variable accessible to programs launched from the shell or shell script that made the assignment. In other words, you can set an environment variable in one script and use it in another script that the first script launches.																		
• Type	<p>If you would like to determine how an executable program would be handled, you can use the type command as follows: type free</p> <p>The result will show you the program’s directory location</p>																		
<b>2.2 Using the Command Line to Get Help</b>																			
<b>Key Knowledge Areas:</b>																			
• Man	<ul style="list-style-type: none"> <li>• Manual pages describe not only programs, but also configuration files and other features of a Linux installation.</li> <li>• Intended as quick references to help somebody who’s already at least somewhat familiar with a command, configuration file, or other OS feature.</li> </ul>																		
• Info	<ul style="list-style-type: none"> <li>• The basic design of man pages dates back decades, so it predates some important developments in managing information. Most notably, man pages are not hyperlinked.</li> <li>• <b>The goal of info pages is to overcome these problems by supporting hyperlinking.</b></li> </ul>																		
<b>Terms and Utilities:</b>																			
• man	<p>Manual pages.</p> <ul style="list-style-type: none"> <li>• man ls -- displays the manual page for the program ls</li> <li>• Man history</li> </ul>																		
• info	Info provides more details, and supports hyperlinks																		



<ul style="list-style-type: none"><li>• Man pages</li></ul>	Additional to the documentation that you find on your computer <ul style="list-style-type: none"><li>• apropos is very similar to man pages</li></ul>									
<ul style="list-style-type: none"><li>• /usr/share/doc/</li></ul>	<p>The /usr/doc and /usr/share/ doc directories often contain a great deal of useful information. If you cannot find the information you are seeking in the man or info pages, look in these directories.</p> <p>The most likely places are as follows: /usr/doc/packageName /usr/share/doc/packageName /usr/share/doc/packages/packageName</p>									
<ul style="list-style-type: none"><li>• locate</li></ul>	Locate command searches a database of file names that Linux maintains. It can therefore do its job much quicker than find can, but you can't control the part of the computer that the system searches.									
<b>2.3 Using Directories and Listing Files</b>										
<b>Key Knowledge Areas:</b>										
<ul style="list-style-type: none"><li>• Files, directories</li></ul>	<div>Everything is a file, directories are just a special type of file. Files that point to devices such as USB ports and hard drives. These files are located within the <b>/dev</b> directory.</div> <p>Using the command <b>file</b>, we can determine the file type for a given file or files.</p>									
<ul style="list-style-type: none"><li>• Hidden files and directories</li></ul>										
<ul style="list-style-type: none"><li>• Home</li></ul>										
<ul style="list-style-type: none"><li>• Absolute and relative paths</li></ul>										
<b>Terms and Utilities:</b>										
<ul style="list-style-type: none"><li>• Common options for ls</li></ul>	<p>The ls command, whose name is short for list , provides this information by displaying the names of files in a directory.</p> <table><tr><th>Option (long form)</th><th>Option (short form)</th><th>Description</th></tr><tr><td>--all</td><td>-a</td><td>Normally, ls omits files whose names begin with a dot ( . ). These <i>dot files</i> (also known as <i>hidden files</i>) are often configuration files that aren't usually of interest. Adding this parameter displays dot files.</td></tr><tr><td>--color</td><td>N/A</td><td>This option produces a color-coded listing that differentiates directories and other special file types by displaying them in different colors. Some Linux distributions configure their shells to use this option by default.</td></tr></table>	Option (long form)	Option (short form)	Description	--all	-a	Normally, ls omits files whose names begin with a dot ( . ). These <i>dot files</i> (also known as <i>hidden files</i> ) are often configuration files that aren't usually of interest. Adding this parameter displays dot files.	--color	N/A	This option produces a color-coded listing that differentiates directories and other special file types by displaying them in different colors. Some Linux distributions configure their shells to use this option by default.
Option (long form)	Option (short form)	Description								
--all	-a	Normally, ls omits files whose names begin with a dot ( . ). These <i>dot files</i> (also known as <i>hidden files</i> ) are often configuration files that aren't usually of interest. Adding this parameter displays dot files.								
--color	N/A	This option produces a color-coded listing that differentiates directories and other special file types by displaying them in different colors. Some Linux distributions configure their shells to use this option by default.								

Option (long form)	Option (short form)	Description
<code>--directory</code>	<code>-d</code>	Normally, if you type a directory name as an option, <code>ls</code> displays the contents of that directory. The same thing happens if a directory name matches a wildcard. Adding this parameter changes this behavior to list only the directory name, which is sometimes preferable.
N/A	<code>-l</code>	The <code>ls</code> command normally displays filenames only. The <code>-l</code> parameter (a lowercase <i>l</i> ) produces a long listing that includes information such as the file's permission string, owner, group, size, and creation date.
<code>--classify</code>	<code>-F</code>	This option appends an indicator code to the end of each name so you know what type of file it is.
<code>--recursive</code>	<code>-R</code>	The <code>-R</code> or <code>--recursive</code> option causes <code>ls</code> to display directory contents recursively: if the target directory contains a subdirectory, <code>ls</code> displays both the files in the target directory <i>and</i> the files in its subdirectory. The result can be a huge listing if a directory has many subdirectories.

- Recursive listings

- `cd`

- The `cd` command changes the current directory in which you're working.
  - `cd /usr/bin`

- `.` and `..`

- `home` and `~`

## 2.4 Creating, Moving and Deleting Files

### Key Knowledge Areas:

- Files and directories

- Case sensitivity

File system is case sensitive. `Russ.txt` is different compared to `russ.txt`

- Simple globbing and quoting

### Terms and Utilities:

- `mv`, `cp`, `rm`, `touch`

- You can type this program's name followed by the name of a file that you want to create, such as **`touch`** `newfile.txt` to create an empty file called `newfile.txt`
- If you're working in a text-mode shell, the **`cp`** command copies a file.
- In a text-mode shell, the same command, **`mv`**, is used both to move and rename files and directories.
  - Its use is similar to that of `cp`; for instance, if you wanted to move `outline.pdf` to `~/publication`, you would type the following:
    - **`mv outline.pdf ~/publication`**
- The `rm` command deletes files in a text-mode shell.
  - `rm -r oldstuff/`

<ul style="list-style-type: none"><li>• mkdir, rmdir</li></ul>	<ul style="list-style-type: none"><li>• You can use the mkdir command to create a directory.<ul style="list-style-type: none"><li>• mkdir newdir</li><li>• mkdir dirone newdir/dirtwo</li></ul></li><li>• The rmdir command is the opposite of mkdir ; it destroys a directory.<ul style="list-style-type: none"><li>• rmdir dirone</li><li>• rmdir newdir/dirtwo newdir</li></ul></li></ul>																								
<b>3.1 Archiving Files on the Command Line</b>																									
<ul style="list-style-type: none"><li>• Files, directories</li></ul>																									
<ul style="list-style-type: none"><li>• Archives, compression</li></ul>	The tar program is a popular tool used to archive various data files into a single file, called an archive file (the original files remain on your disk)																								
<b>Terms and Utilities:</b>																									
<ul style="list-style-type: none"><li>• Tar</li></ul>	The tar program’s name stands for tape archiver. Regardless of its name, you can use tar to back up (also called archive ) data to your hard disk or other media, not just to tapes. Archiving and compressing files to create a <b>tarball</b>																								
<ul style="list-style-type: none"><li>• Common tar options</li></ul>	<table><tr><th>Command</th><th>Abbreviation</th><th>Description</th></tr><tr><td>--create</td><td>c</td><td>Creates an archive</td></tr><tr><td>--concatenate</td><td>A</td><td>Appends tar files to an archive</td></tr><tr><td>--append</td><td>r</td><td>Appends non-tar files to an archive</td></tr><tr><td>--update</td><td>u</td><td>Appends files that are newer than those in an archive</td></tr><tr><td>--diff or --compare</td><td>d</td><td>Compares an archive to files on disk</td></tr><tr><td>--list</td><td>t</td><td>Lists an archive’s contents</td></tr><tr><td>--extract or --get</td><td>x</td><td>Extracts files from an archive</td></tr></table>	Command	Abbreviation	Description	--create	c	Creates an archive	--concatenate	A	Appends tar files to an archive	--append	r	Appends non-tar files to an archive	--update	u	Appends files that are newer than those in an archive	--diff or --compare	d	Compares an archive to files on disk	--list	t	Lists an archive’s contents	--extract or --get	x	Extracts files from an archive
Command	Abbreviation	Description																							
--create	c	Creates an archive																							
--concatenate	A	Appends tar files to an archive																							
--append	r	Appends non-tar files to an archive																							
--update	u	Appends files that are newer than those in an archive																							
--diff or --compare	d	Compares an archive to files on disk																							
--list	t	Lists an archive’s contents																							
--extract or --get	x	Extracts files from an archive																							

<code>--directory <i>dir</i></code>	C	Changes to directory <i>dir</i> before performing operations
<code>--file [<i>host:</i>]<i>file</i></code>	f	Uses the file called <i>file</i> on the computer called <i>host</i> as the archive file
<code>--listed-incremental <i>file</i></code>	g	Performs an incremental backup or restore, using <i>file</i> as a list of previously archived files
<code>--one-file-system</code>	(none)	Backs up or restores only one file-system (partition)
<code>--multi-volume</code>	M	Creates or extracts a multitape archive
<code>--tape-length N</code>	L	Changes tapes after N kilobytes
<code>--same-permissions</code>	p	Preserves all protection information
<code>--absolute-names</code>	P	Retains the leading / on filenames
<code>--verbose</code>	v	Lists all files read or extracted; when used with <code>--list</code> , displays file sizes, ownership, and time stamps
<code>--verify</code>	W	Verifies the archive after writing it
<code>--exclude <i>file</i></code>	(none)	Excludes <i>file</i> from the archive
<code>--exclude-from <i>file</i></code>	X	Excludes files listed in <i>file</i> from the archive
<code>--gzip</code> or <code>--ungzip</code>	z	Processes an archive through gzip
<code>--bzip2</code>	j (some older versions used I or y)	Processes an archive through bzip2
<code>--xz</code>	J	Processes an archive through xz

- gzip, bzip2

gzip, bzip2, and xz programs all compress individual files. For instance, you might compress a large graphics file like this: `xz biggraphics.tiff`

Compression program	Uncompression program	Filename extension
gzip	gunzip	.gz
bzip2	bunzip2	.bz2
xz	unxz	.xz

- zip, unzip

Linux provides the zip command to create zip files, and the unzip utility to extract files from a zip archive. Zip files typically have file name extensions of .zip.

## 3.2 Searching and Extracting Data

from Files	
Key Knowledge Areas:	
• Command line pipes	
• I/O re-direction	
• Basic Regular Expressions ., [ ], *, ?	<ul style="list-style-type: none"> <li>• Regular expressions , which are a way to describe patterns that you might want to look for in data files. <ul style="list-style-type: none"> <li>• Two forms of regular expression are common: basic and extended</li> <li>• <b>Bracket Expressions Characters</b> enclosed in square brackets ( [ ] ) constitute bracket expressions, which match any one character within the brackets. For instance, the regular expression b[aeiou]g matches the words bag , beg , big , bog , and bug . Including a caret ( ^ ) after the opening square bracket matches against any character except the ones specified. For instance, b[^aeiou]g matches bbg or bAg but not bag or beg.</li> <li>• <b>Range Expressions</b> A range expression is a variant on a bracket expression. Instead of listing every character that matches, range expressions list the start and end points separated by a dash ( - ), as in a[2-4]z . This regular expression matches a2z , a3z , and a4z .</li> <li>• <b>Any Single Character The dot ( . )</b> represents any single character except a newline. For instance, a.z matches a2z , abz , aQz , or any other three-character string that begins with a and ends with z.</li> <li>• <b>Start and End of Line</b> The caret ( ^ ) represents the start of a line, and the dollar sign ( \$ ) denotes the end of a line. For instance, ^bag matches bag only if it is first in a line of characters, while bag\$ matches bag only if it is last in a line of characters.</li> <li>• <b>Repetition</b> A full or partial regular expression may be followed by a special symbol to denote repetition of the matched item. Specifically, an asterisk ( * ) denotes zero or more matches. The asterisk is often combined with the dot (as in .* ) to specify a match with any substring. For instance, A.*Lincoln matches any string that contains A and Lincoln , in that order—Abe Lincoln and Abraham Lincoln are just two possible matches. <ul style="list-style-type: none"> <li>○ <b>Additional Repetition Operators</b> These operators work like an asterisk, but they match only certain numbers of matches. Specifically, a plus sign ( + ) matches one or more occurrences, and a question mark ( ? ) specifies zero or one match.</li> </ul> </li> </ul> </li> </ul>
• Grep	The grep command searches for files that contain a specified string and returns the name of the file and (if it's a text file) the line containing that string.

Option (long form)	Option (short form)	Description
<code>--count</code>	<code>-c</code>	Instead of displaying the lines that contain matches to the regular expression, displays the number of lines that match.
<code>--file=file</code>	<code>-f file</code>	This option takes pattern input from the specified file rather than from the command line. The <code>fgrep</code> command is a shortcut for this option.
<code>--ignore-case</code>	<code>-i</code>	You can perform a case-insensitive search, rather than the default case-sensitive search, by using the <code>-i</code> or <code>--ignore-case</code> option.
<code>--recursive</code>	<code>-R</code> or <code>-r</code>	This option searches in the specified directory and all of the subdirectories rather than simply the specified directory. You can use <code>rgrep</code> rather than specify this option.
<code>--extended-regexp</code>	<code>-E</code>	The <code>grep</code> command uses basic regular expressions by default. To use an extended regular expression, you can pass this option. Alternatively, you can call <code>egrep</code> rather than <code>grep</code> ; this variant command uses extended regular expressions by default.

- `grep -r eth0 /etc/*`

- **The find utility** implements a brute-force approach to finding files. This program finds files by searching through the specified directory tree, checking file names, file creation dates, and so on to locate the files that match the specified criteria. Because of this operation method, `find` tends to be slow.

- Less

- sort

When dealing with a large amount of data, being able to sort it is often useful.

```
$ cat pets.txt
```

```
fish
```

```
dog
```

```
cat
```

```
bird
```

```
$ sort pets.txt
```

```
bird
```

```
cat
```

```
dog
```

```
fish
```

```

Terminal
christine@server01: ~
$ cat numbers.txt
2
1
3
10
12
20
$ sort numbers.txt
1
10
12
2
20
3
$ sort -n numbers.txt
1
2
3
10
12
20
$

```

sorting numeric data

Option (long form)	Option (short form)	Description
--dictionary-order	-d	Considers only blanks and alphanumeric characters; doesn't consider special characters
--ignore-case	-f	Ignores case (default is to consider case and order capitalized letters first)
--numeric-sort	-n	Sorts by string numeric value
--output=FILE	-o	Writes results to file specified
--reverse	-r	Sorts in descending order (default is to sort ascending)

• cut	It extracts text from fields in a file record. It's frequently used to extract variable information from a file whose contents are highly patterned.  cut -f 6 -d ":" /etc/passwd
• wc	You might need to know how many words or lines are in a text file—say, because you want to know how many pages a text document will consume when printed at 52 lines per page.  wc newfile.txt 37 59 1990 newfile.txt

3.3 Turning Commands into a Script

Key Knowledge Areas:

- Basic shell scripting
- Awareness of common text editors



<b>Terms and Utilities:</b>	
• <code>#!</code> (shebang)	The first two characters are a special code that tells the Linux kernel that this is a script and to use the rest of the line as a pathname to the program that's to interpret the script. <b><code>#!/bin/bash</code></b>
• <code>/bin/bash</code>	On most systems, <code>/bin/sh</code> is a symbolic link that points to <code>/bin/bash</code> , but it can point to another shell. Specifying the script as using <code>/bin/sh</code> guarantees that any Linux system will have a shell program to run the script.
• Variables	Is a placeholder in a script for a value that will be determined when the script runs.
• Arguments	Variables that are passed to the script are frequently called parameters or arguments. They're represented in the script by a dollar sign ( <code>\$</code> ) followed by a number from 0 up— <code>\$0</code> stands for the name of the script, <code>\$1</code> is the first parameter to the script, <code>\$2</code> is the second parameter, and so on.
• for loops	Loops are structures that tell the script to perform the same task repeatedly until a particular condition is met (or until some condition is no longer met).  A script that executes a command on every matching file in a directory <b><code>#!/bin/bash</code></b> <code>for d in `ls *.wav`; do</code> <code>  aplay \$d</code> <code>done</code>
• <code>echo</code>	
• Exit status	Control the exit value, or exit from the script at any point. Used without any options, <code>exit</code> causes immediate termination of the script, with the usual exit value of <code>\$?</code> .
<b>4.1 Choosing an Operating System</b>	
<b>Key Knowledge Areas:</b>	
• Windows, Mac, Linux differences	
• Distribution life cycle management	
<b>Terms and Utilities:</b>	
• GUI versus command line, desktop configuration	
• Maintenance cycles, Beta and Stable	
<b>4.2 Understanding Computer Hardware</b>	
• Hardware	The capabilities and limitations of your hardware will influence the capabilities and limitations of Linux running on that hardware.
• Motherboards, processors, power supplies, optical drives, peripherals	<ul style="list-style-type: none"> <li>• Motherboard is a large circuit board inside the computer. It's dominated by a chipset, which is one or more chips that provide key functionality for the computer—they handle the hard disk interfaces, the USB interfaces, the network devices, and so on. Some chipsets include video circuitry for video cards, although this functionality is sometimes separate, and sometimes it's built into the CPU. <b>Lspci</b></li> <li>• CPU (sometimes called the processor) is the "brain" of your computer—it does most of the computer's actual computing</li> </ul>

	<ul style="list-style-type: none"> <li>• <code>uname -a</code></li> <li>• <code>lscpu</code></li> <li>• <code>cat /proc/cpuinfo</code></li> </ul> <ul style="list-style-type: none"> <li>• Computer's power supply takes the alternating current (AC) power from a wall outlet and converts it to the direct current (DC) that your motherboard and everything you plug into it uses.</li> <li>• If you insert a removable disk into a computer that's running most modern Linux distributions, the computer will probably detect that fact, mount the disk in a subdirectory of <code>/media</code>, and launch a file manager on the disk. This behavior makes the system work in a way that's familiar to users of Windows or Mac OS.</li> <li>• Most modern computers use USB ports as the primary interface for external peripherals. Keyboards, mice, cameras, flash storage, hard disks, network adapters, scanners, printers, and more can all connect via USB. For the most part, USB devices work in a plug-and-play manner—you plug them in, and they work.</li> </ul>
• Hard drives and partitions, <code>/dev/sd*</code>	<ul style="list-style-type: none"> <li>• <b>PATA</b> This interface was common in the past, but it's fading in popularity. It features wide 40- or 80-pin cables that transfer several bits of data simultaneously—hence the word parallel in the name Parallel ATA (PATA)</li> <li>• <b>SATA</b> is more or less software compatible with PATA, but it uses thinner cables that can handle just one hard disk per cable.</li> <li>• Disk partitions exist to help subdivide the disk into pieces with broadly different purposes, such as partitions for different OSs or for different types of data within an OS. <ul style="list-style-type: none"> <li>• <b>Primary</b> This is the simplest type of partition. A disk can have zero to four primary partitions, one of which may be an extended partition.</li> <li>• <b>Extended</b> This is a special type of primary partition that serves as a placeholder for logical partitions. A disk may have at most one extended partition.</li> <li>• <b>Logical</b> These partitions are contained within an extended partition. In theory, a disk can have billions of logical partitions, thus overcoming the limit of four primary partitions, but in practice you're unlikely to see more than about a dozen of them.</li> </ul> </li> <li>• <b>To partition a disk</b>, you must know the disk's device file name. In Linux, these file names are normally <code>/dev/sda</code>, <code>/dev/sdb</code>, and so on, with each disk taking on a new letter. Partitions are numbered starting with 1, so you might refer to <code>/dev/sda2</code>, <code>/dev/sdb6</code>, and so on. When using MBR, partitions 1 through 4 are reserved for primary or extended partitions, whereas logical partitions take numbers 5 and up.</li> </ul>
• Drivers	A piece of software that “talks” to hardware is known as a driver
<b>4.3 Where Data is Stored</b>	
• Programs and configuration, packages and package databases	
• Processes, memory addresses, system messaging and logging	
<b>Terms and Utilities:</b>	
• <code>ps</code> , <code>top</code> , <code>free</code>	<ul style="list-style-type: none"> <li>• <code>ps</code>: shows list of running programs</li> <li>• <code>top</code>: interactive live display of processes</li> <li>• <code>free</code>: shows free memory</li> </ul>
• <code>syslog</code> , <code>dmesg</code>	<ul style="list-style-type: none"> <li>• program that manages storage of logs for daemons on the system</li> <li>• used to view kernel messages</li> </ul>

• /etc/, /var/log/	<ul style="list-style-type: none"> <li>• /etc: contains config files</li> <li>• /var/log/: log files</li> </ul>
• /boot/, /proc/, /dev/, /sys/	<ul style="list-style-type: none"> <li>• /boot/: contains the actual operating system</li> <li>• /proc/: does not occupy space on disk. Data about running processes as well as other information the kernel possesses about the computer's hardware <ul style="list-style-type: none"> <li>• /proc/cpuinfo: This contains information about the CPU's type and clock frequency.</li> <li>• /proc/devices: This is a complete list of devices supported by the kernel including their major device numbers.</li> </ul> </li> <li>• /dev/: entries for device files.</li> <li>• /sys/: hardware control</li> </ul>
<b>4.4 Your Computer on the Network</b>	
<b>Key Knowledge Areas:</b>	
• Internet, network, routers	
• Querying DNS client configuration	
• Querying Network configuration	
<b>Terms and Utilities:</b>	
• route, ip route show	
• ifconfig, ip addr show	<ul style="list-style-type: none"> <li>• ifconfig: command for network configuration. Used to query the setup of a network interface</li> <li>• ip addr show:</li> </ul>
• netstat, ip route show	<ul style="list-style-type: none"> <li>• netstat: provides information about your computer and its network connection.</li> <li>• ip route show: view / display routing table</li> </ul>
• /etc/resolv.conf, /etc/hosts	
• IPv4, IPv6	<p>IPv4: 32-bit address, 4 billion addresses, decimal, 127.0.0.1</p> <p>IPv6: 128-bit address, hexadecimal, ::1</p>
• Ping	low-level (IP) connectivity checks between your computer and others.
• Host	
<b>Topic 5: Security and File Permissions</b> <b>(weight: 7)</b>	
<b>5.1 Basic Security and Identifying User Types</b>	
<b>Key Knowledge Areas:</b>	
• Root and Standard Users	
• System users	
<b>Terms and Utilities:</b>	
• /etc/passwd, /etc/group	
• id, who, w	Id:
• sudo, su	do command with root privileges
<b>5.2 Creating Users and Groups</b>	

<b>Key Knowledge Areas:</b>	
• User and group commands	
• User IDs	
<b>Terms and Utilities:</b>	
• /etc/passwd, /etc/shadow, /etc/group, /etc/skel/	<ul style="list-style-type: none"> <li>• /etc/passwd: contains root:X:0:0:/root:/bin/bash also <b>contains user ID</b></li> <li>• /etc/shadow: passwords (encrypted)</li> <li>• /etc/group: group info</li> <li>• /etc/skel/: contains default files for useradd command, copies files to home directory of new user</li> </ul>
• id, last	<ul style="list-style-type: none"> <li>• id: find out a user account's UID, the primary and secondary groups and the corresponding GIDs</li> <li>• last: who logged into your computer and when (and, in the case of logins via the network, from where)</li> </ul>
• useradd, groupadd	<ul style="list-style-type: none"> <li>• Useradd: create a new user or update default new user information</li> <li>• Groupadd: create a new group</li> </ul>
• Passwd	Change user password
<b>5.3 Managing File Permissions and Ownership</b>	
<b>Key Knowledge Areas:</b>	
• File/directory permissions and owners	
• Terms and Utilities:	
• ls -l, ls -a	Ls-l: command to list files in directory, -l - long listing (more info)
• chmod, chown	<ul style="list-style-type: none"> <li>• Chmod: change the permissions of file to octal, which can be found separately for user, group, and world by adding: <ul style="list-style-type: none"> <li>• 4 – read ®</li> <li>• 2 – write (w)</li> <li>• 1 – execute (x)</li> </ul> Examples:  chmod 777 – read, write, execute for all  chmod 755 – rwx for owner, rx for group and world  For more options, see man chmod.</li> <li>• Chown: change file owner and group</li> </ul>
<b>5.4 Special Directories and Files</b>	
<b>Key Knowledge Areas:</b>	
• Using temporary files and directories	
• Symbolic links	
<b>Terms and Utilities:</b>	
• /tmp/, /var/tmp/ and Sticky Bit	<ul style="list-style-type: none"> <li>• <b>/tmp/</b>: short term storage, gets erased on boot. All users can write too so they can delete others things</li> <li>• <b>/var/tmp/</b>: short term storage but doesn't get deleted on boot</li> <li>• <b>Sticky Bit</b>: Only owners of file can delete, chmod 0 + t, chmod 1777</li> </ul>
• ls -d	command to list files in directory -d */ list directories

- ln -s

file location for link, symbolic link

# Notes on Ubuntu

Thursday, September 28, 2017 6:53 AM

RDP:

How to RDP into Ubuntu:

<https://askubuntu.com/questions/592537/can-i-access-ubuntu-from-windows-remotely>

Fix:

<https://askubuntu.com/questions/797973/error-problem-connecting-windows-10-rdp-into-xrdp>

# Linux Essentials (010-150)

Wednesday, January 10, 2018 5:26 PM

## **EXAM OBJECTIVES** (40 Questions / 60 minutes / 500 to pass)

**\*\* "WEIGHT" UNDER EACH SECTION INDICATES HOW MANY QUESTION ON THAT TOPIC WILL BE ON THE EXAM\*\***

<https://www.theurbanpenguin.com/lpi-training-from-theurbanpenguin/linux-essentials/>

### **Topic 1: The Linux Community and a Career in open source (weight: 7)**

#### **1.1 Linux Evolution and Popular Operating Systems**

<https://www.theurbanpenguin.com/linux-evolution-and-popular-operating-systems/>

<https://youtu.be/ngA7N97Ly34>

#### **Weight: 2**

**Description:** Knowledge of Linux development and major distributions

#### **Key Knowledge Areas:**

- Open source Philosophy
  - software for which the source code (the inner workings of the program) is freely available for anyone to download, modify, and redistribute
- Distributions (often abbreviated as distro)
  - command **uname -r** will show your kernel version and **lsb\_release -a** will show your Distribution
  - **Red Hat:** Provide paid support and is often a safe choice for the enterprise
  - **Ubuntu LTS:** Long Term Support versions are released every two years and supported for five years. The current version is 14.04 and that will be supported until 2019. Optional paid support is available
  - **CentOS:** Is a Red Hat rebuild and provide long term updates and community support
  - **Debian:** Is a respected Enterprise version with community Support. The Raspberry Pi OS, Raspbian, is based on Debian
  - Widely Used Distributions
    - Debian, a non-commercial distribution and one of the earliest, maintained by a volunteer developer community with a strong commitment to free software principles and democratic project management
      - Uses a dpkg management system
      - Knoppix, the first Live CD distribution to run completely from removable media without installation to a hard disk, derived from Debian
      - Linux Mint Debian Edition (LMDE) uses Debian packages directly (rather than Ubuntu's)
      - Ubuntu, a desktop and server distribution derived from Debian, maintained by British company Canonical Ltd.
        - ◆ Kubuntu, the KDE version of Ubuntu
        - ◆ Linux Mint, a distribution based on and compatible with Ubuntu. Supports multiple desktop environments, among others GNOME Shell fork Cinnamon and GNOME 2 fork MATE.
        - ◆ Trisquel, an Ubuntu-based distribution based on Linux-libre kernel composed entirely of free software
        - ◆ Elementary OS, an Ubuntu-based distribution with strong focus on the visual experience without sacrificing performance.
    - Fedora, a community distribution sponsored by American company Red Hat and the successor to the company's previous offering, Red Hat Linux. It aims to be a technology testbed for Red Hat's commercial Linux offering, where new open source software is prototyped, developed, and tested in a communal setting before maturing into Red Hat Enterprise Linux.
      - Red Hat Enterprise Linux (RHEL), a derivative of Fedora, maintained and commercially supported by Red Hat. It seeks to provide tested, secure, and stable Linux server and workstation support to businesses.
        - ◆ CentOS, a distribution derived from the same sources used by Red Hat, maintained by a dedicated volunteer community of developers with both 100% Red Hat-compatible versions and an upgraded version that is not always 100% upstream compatible.
        - ◆ Oracle Linux, which is a derivative of Red Hat Enterprise Linux, maintained and commercially



- supported by Oracle
  - ◆ Scientific Linux, a distribution derived from the same sources used by Red Hat, maintained by Fermilab
- Mandriva Linux was a Red Hat derivative popular in several European countries and Brazil, backed by the French company of the same name. After the company went bankrupt, it was superseded by OpenMandriva Lx, although a number of derivatives now have a larger user base.
  - Mageia, a community fork of Mandriva Linux created in 2010
  - PCLinuxOS, a derivative of Mandriva, which grew from a group of packages into a community-spawned desktop distribution
  - ROSA Linux, another former derivative of Mandriva, now developed independently
- openSUSE, a community distribution mainly sponsored by German company SUSE
  - SUSE Linux Enterprise, derived from openSUSE, maintained and commercially supported by SUSE
- Arch Linux, a rolling release distribution targeted at experienced Linux users and maintained by a volunteer community, offers official binary packages and a wide range of unofficial user-submitted source packages. Packages are usually defined by a single PKGBUILD text file
  - Manjaro Linux, a derivative of Arch Linux that includes a graphical installer and other ease-of-use features for less experienced Linux users. Rolling release packages from Arch repositories are held for further testing to achieve increased stability, and packages identified as addressing security issues of critical or high severity are "fast-tracked" to the stable branch
- Gentoo, a distribution targeted at power users, known for its FreeBSD Ports-like automated system for compiling applications from source code
  - Chrome OS, Google's commercial operating system (using Gentoo and its Portage) that primarily runs web applications
    - ◆ Chromium OS, the fully open-source version of Chrome OS
- Slackware, created in 1993, one of the first Linux distributions and among the earliest still maintained, committed to remain highly Unix-like and easily modifiable by end users
- Niche Distributions
  - Routers – for example, targeted by the tiny embedded router distribution OpenWrt
  - Internet of things – for example, targeted by Ubuntu Core
  - Home theater PCs – for example, targeted by KnoppMyth, Kodi (former XBMC) and Mythbuntu
  - Specific platforms – for example, Raspbian targets the Raspberry Pi platform
  - Education – examples are Edubuntu and Karoshi, server systems based on PCLinuxOS
  - Scientific computer servers and workstations – for example, targeted by Scientific Linux
  - Digital audio workstations for music production – for example, targeted by Ubuntu Studio
  - Computer Security, digital forensics and penetration testing – examples are Kali Linux and Parrot Security OS
  - Privacy and anonymity – for example, targeted by Tails
  - Offline use – for example, Endless OS
- Android and non-GNU distributions
  - Whether Google's Android counts as a Linux distribution is a matter of definition
  - It uses the Linux kernel, so the Linux Foundation agrees that Android is a Linux distribution
  - Others, disagree by noting the lack of support for many GNU tools in Android, including glibc
  - Other non-GNU distributions include Cyanogenmod, its fork LineageOS, Android-x86 and recently Tizen
- Embedded Systems

**The following is a partial list of the used files, terms and utilities:**

- Android
- Debian, Ubuntu (LTS)
- CentOS, openSUSE, Red Hat
- Linux Mint, Scientific Linux

## 1.2 Major open source Applications

<https://www.theurbanpenguin.com/297/>

<https://youtu.be/6KzdmrKdow>

<https://youtu.be/ZMxFwZMjdpU>

**Weight: 2**

**Description:** Awareness of major applications as well as their uses and development.

**Key Knowledge Areas:**

- Desktop Applications
  - Browser - Firefox
  - Office Suite - LibreOffice; Apache OpenOffice.org (discontinued)
  - Music Manager - Clementine
  - Video Player - VLC
  - Text Editor - Gedit
  - Image Editor - GIMP
  - Photo Manager - Shotwell
  - Mail Client - Thunderbird
  - BitTorrent - Transmission
- Server Applications
  - XBMC
  - Apache HTTPD
  - NGINX
  - MySQL
  - NFS
  - Samba
- Development Languages
  - C
  - C++
  - Python
  - Java
  - C#
  - Fortran
  - Pascal
  - COBOL
  - Lisp
  - Perl
- Package Management Tools and repositories

**Terms and Utilities:**

- OpenOffice.org, LibreOffice, Thunderbird, Firefox, GIMP
- Apache HTTPD, NGINX, MySQL, NFS, Samba
- C, Java, Perl, shell, Python, Samba
- dpkg, apt-get, rpm, yum

**1.3 Understanding open source Software and Licensing**

<https://www.theurbanpenguin.com/linux-essentials-13-understanding-open-source-software-and-licensing/>

<https://youtu.be/o6Ct6JaHcRk>

**Weight: 1**

Description: Open communities and licensing open source Software for business.

**Key Knowledge Areas:**

- Licensing
  - GPL (General Public License)
    - A copyleft license
    - type of license that attempts to ensure that the public retains the freedom to use, modify, extend and redistribute a creative work and all derivative works (i.e., works based on or derived from it) rather than to restrict such freedoms
  - BSD (Berkley Software Distribution)
    - Permissive free software licenses
    - Imposes minimal restrictions on the use and redistribution of covered software
    - A simple license that merely requires that all code licensed under the BSD license be licensed under the BSD license if redistributed in source code format
    - BSD (unlike some other licenses) does not require that source code be distributed at all

- Creative Commons
  - Public copyright licenses that enable the free distribution of an otherwise copyrighted work
  - A CC license is used when an author wants to give people the right to share, use, and build upon a work that they have created
  - CC provides an author flexibility (for example, they might choose to allow only non-commercial uses of their own work) and protects the people who use or redistribute an author's work from concerns of copyright infringement as long as they abide by the conditions that are specified in the license by which the author distributes the work
- Free Software Foundation (FSF)
  - A 501(c)(3) non-profit organization founded by Richard Stallman on 4 October 1985 to support the free software movement
  - Promotes the universal freedom to study, distribute, create, and modify computer software, with the organization's preference for software being distributed under copyleft ("share alike") terms, such as with its own GNU General Public License
  - The FSF was incorporated in Massachusetts, USA, where it is also based
- Open source Initiative (OSI)
  - A non-profit organization dedicated to promoting open-source software
  - The Open Source Initiative chose the term "open source" to "dump the moralizing and confrontational attitude that had been associated with 'free software'" and instead promote open source ideas on "pragmatic, business-case grounds"
  - Open Source Business Models:
    - Dual-licensing
    - Selling professional services
    - Selling of branded merchandise
    - Selling of certificates and trademark use
    - Selling software as a service
    - Partnership with funding organizations
    - Voluntary donations
    - Bounty driven development
    - Pre-order/crowdfunding/reverse-bounty model
    - Crowdsourcing
    - Advertising-supported software
    - Selling of optional proprietary extensions
    - Selling of required proprietary parts of a software product
    - Selling of proprietary update systems
    - Re-licensing under a proprietary license
    - Obfuscation of source code
    - Delayed open-sourcing
    - Open sourcing on end-of-life

#### **Terms and Utilities:**

- GPL, BSD, Creative Commons
- Free Software, open source Software, FOSS, FLOSS
- open source business models

#### **1.4 ICT Skills and Working in Linux**

<https://www.theurbanpenguin.com/linux-essentials-14-ict-skills-for-working-with-linux/>

<https://youtu.be/-kRON17Kul8>

**Weight: 2**

**Description:** Basic Information and Communication Technology (ICT) skills and working in Linux.

#### **Key Knowledge Areas:**

- Desktop Skills
  - Window Managers
    - GNOME
    - KDE

- Unity
- Getting to the Command Line
  - You can do this through Window Manager accessing some form of graphic console such as the KDE console program or on the Gnome-terminal
  - If you have physical access to the desktop or server you can usually access the physical terminal screens or terminals using the Alt + Ctrl + F1 for tty1, Alt + Ctrl + F2 for tty2, etc.
  - You can return to the comfort of the graphic console with Alt + Ctrl + F7
- Industry uses of Linux, Cloud Computing and Virtualization

#### Terms and Utilities:

- Using a browser, privacy concerns, configuration options, searching the web and saving content
- Terminal and Console
- Password issues
- Privacy issues and tools
- Use of common open source applications in presentations and projects

## Topic 2: Finding Your Way on a Linux System (weight: 9)

### 2.1 Command Line Basics

<https://www.theurbanpenguin.com/linux-essentials-21-command-line-basics/>

<https://youtu.be/cKMiVkBxfKE>

[https://youtu.be/ch\\_uplv3mMU](https://youtu.be/ch_uplv3mMU)

<https://youtu.be/1Ukw0ljGKsl>

#### Weight: 3

**Description:** Basics of using the Linux command line.

#### Key Knowledge Areas:

- Basic shell
  - Open a shell and exit with the exit command or Ctrl + d
  - These shells can be physical terminals on the server or remote terminals using SSH Connections
  - If using the GUI, open Pseudo-Terminals in the GUI
  - Within this GUI Terminal, you can easily adjust font size with Ctrl + Shift + + and reduce with Ctrl + -
  - To clear a screen we have the clear command to just ctrl + l
    - Ctrl + l is a short-cut key sequence within the bash shell but not all shells, so check the shell that you are using
    - The output from the command: **echo \$SHELL** will let you know your current shell
  - All of the commands typed are stored in a history file, .bash\_history so they persist even after reboots
  - The command history itself can be used to display and clear the command line history for a given user
  - Each user has their own unique .bash\_history file which is created in the user's home directory
    - history
      - ◻ Without any arguments displays the users history
    - history 3
      - ◻ Shows the last 3 lines of the users history
    - history -c
      - ◻ Clears the users history
  - A simple way to get used to the history is just using the up arrow key to scroll through last used commands
- Command line syntax
- Variables
  - Variables come into play to store information about our session
  - When reading variables they are prefixed with a \$ symbol
  - Typing \$ followed by **TAB TAB** in quick succession will list all variables as will the env (/usr/bin/env) command
  - Additionally the command env displays the variables and their values
  - The \$ symbol identifies variables that need to be expanded in the command line
  - \$\$ is a special variable that identifies the current process id
  - This can be used to find out what when is currently running

- Globbing
  - <http://tldp.org/LDP/abs/html/globbingref.html>
- Quoting
  - mkdir dir1 dir2
    - Spacing out options will make them seem as two options which means that we can create two directories with the one command
  - mkdir "dir1 dir2"
    - If you want a space in the directory name use quotes or the (backslash) character
    - This will make a single directory named dir1 dir2
  - mkdir dir && cd dir1
    - This will make the directory and if the command succeeds will enter the directory with the command cd
  - id bob || useradd bob
    - Using the double vertical bar or pipe will id the first command in the case of there not being a user bob then the useradd command will run and we create the user named bob

#### Terms and Utilities:

- Bash
- echo
- history
- PATH env variable
- export
- type

## 2.2 Using the Command Line to Get Help

<https://www.theurbanpenguin.com/linux-essentials-22-command-line-help/>

<https://youtu.be/LsWWsCva7hY>

### Weight: 2

**Description:** Running help commands and navigation of the various help systems.

#### Key Knowledge Areas:

- Man (manual pages)
  - The traditional longer help options, where we need more detailed help
  - These pages are broken into sections
  - These sections help identify how the command may be used, for example as a standard user or as root
    - Section 1: for help on if used as a standard user
    - Section 5: for help on the configuration file
    - Section 8: for help on using this as root
    - Other sections exist but these are most common
  - Using the command `whatis (/usr/bin/whatism)` it is easy to find man pages that exist for a given command
- Info
  - Some commands may have info pages instead of, or in addition to, man pages
  - Info pages are accessed with the command **info** (/usr/bin/info)
  - If a specific info page does not exist for a given command then /usr/bin/info will open the corresponding man page
  - The idea of the info pages is that they can be hyper-linked to provide for easier reading when the help manual is large
- Locating Man Pages and Programs
  - **whatism** can be used to locate man pages
  - Programs like **man -k** or **apropos** will list where the man pages are located
  - **whereism** will list the executable's and man pages are for a given command

#### Terms and Utilities:

- man
- info
- Man pages
- /usr/share/doc/

- locate

## 2.3 Using Directories and Listing Files

<https://www.theurbanpenguin.com/linux-essentials-23-using-directories-and-listing-files/>

[https://youtu.be/eW9N\\_H0vfyA](https://youtu.be/eW9N_H0vfyA)

**Weight: 2**

**Description:** Navigation of home and system directories and listing files in various locations.

**Key Knowledge Areas:**

- Files, directories
  - We also have files that point to devices such as USB ports and hard drives
  - These files are located within the **/dev** directory
  - Using the command **file**, we can determine the file type for a given file or files
  - **cd -**
    - The **-** option takes you to your previous directory.
    - In this way, it becomes very easy to toggle between two directories just using **cd -**
    - This is maintained by the variable **\$OLDPWD**
- Hidden files and directories
  - hidden files are denoted by those whose file names begin with a dot
  - Using the **-a** option with **ls** you can list all files including hidden files
  - To get a little clever with our command and piping we could try this: **ls -a | grep '^.'**
    - **ls -a** : list all files in the current directory
    - **|** : pipe or redirect the output of **ls** to the next command, in this case **grep**
    - **grep** : used to search text
    - **‘^.’** : **^** lines that start with **.** means literally a dot. The backslash ensure that **grep** reads the dot as a dot and removes any special meaning from the dot character may have
- Home
  - A users home directory will normally be located at **/home/USERNAME**
  - This where they can store their own files including personal login scripts and the bash history (a text file that contains a list of recently entered commands)
  - Typically a user will be taken to their home directory upon logon to a text terminal or remote SSH (Secure **Shell**) session
  - The environment variable **\$HOME** will list the path to a user's home directory
  - We can print to the screen the contents of a variable using the **echo** command
  - Echo is a shell built-in or a command that is part of bash itself. **echo \$HOME**
  - Users may return to their home directory using the bash builtin-in **cd** with no arguments, alternatively using: **cd ~**
    - The tilde (**~**) becomes more useful though when you want to reference another users home directory, **cd ~bob**
    - The above command would take you to Bob's home directory, assuming there was a user named Bob
    - Using the command **cd ~/Documents** would take you to your home directory and the subdirectory Documents
- Absolute and relative paths
  - When referencing files (of any type) we can use full path or a path relative to where we are
  - the simplest form of a relative path is just the file-name itself which we can access when we are in the same directory as that file
  - The **.** by itself means *this directory* and **..** *the directory above or parent directory*
    - **.** = This directory
    - **..** = The parent directory to the current directory
    - **../..** = The directory two directories above the current directory.
    - **../etc** = The etc directory is referenced in the parent directory of the current directory
    - If our current directory is **/home/user1**, we can refer the file in **/etc/hosts** in two ways:
      - **../etc/hosts** : a relative path
      - **/etc/hosts** : a full path, the full path will access the file from any directory and will always start

with a forward slash, /

- Directory Listings Using ls
  - -R : recursive listing
  - -l : long listing
  - -a : include all files including hidden files
  - -r : reverse sort the listing
  - -n : numerically sort on the User ID and the Group ID
  - -h : used with -l and prints the size rounded up to KB, MB, GB etc

#### Terms and Utilities:

- Common options for ls
- Recursive listings
- cd
- . and ..
- home and ~

## 2.4 Creating, Moving and Deleting Files

<https://www.theurbanpenguin.com/linux-essentials-24-working-with-files-and-directories/>

<https://youtu.be/DFreHo3UCD0>

[https://youtu.be/xxkEeu\\_gdxY](https://youtu.be/xxkEeu_gdxY)

<https://youtu.be/JxrhWk9xQGw>

<https://youtu.be/hobvwsqIwis>

**Weight: 2**

**Description:** Create, move and delete files and directories under the home directory.

#### Key Knowledge Areas:

- Files and directories
  - In Linux *everything is a file* and file names are case-sensitive
  - Can have files called test, Test and TESTall in the same directory as the names are NOT the same, as they are in different case!
  - Talking of directories, a directory is a special type of file, as is a symbolic link
  - Can use the command **file** to identify the type of file that we are looking at
  - Listing files with ls
    - **ls** can help with identifying the file type
    - Consider ls with the **--color** option enabled
      - ls --color=auto
      - The above command is often turned on by default with ls by means of an alias
      - When you type the command ls, the alias is found in RAM before the command
      - Either way, the color option will display files, directories and links in different colors as shown

```
andrew@D630:~$ ls
bin      Documents  Dropbox    labs      mypipe
Desktop  Downloads  examples.desktop  Music    Pictures
andrew@D630:~$
```

- Another option you may find useful with ls if you do not have a terminal that supports color is the -F option
  - ls -F
  - The -F is a little redundant with the color option so is less used
  - With this option enabled:
    - ◆ directories are listed with a trailing /
    - ◆ executable files with a trailing \*
    - ◆ named pipes with a vertical bar, etc.
- Listing a directory
  - If you want a long listing of a directory rather than the contents of that directory use the command ls -ld /etc
  - This will provide a long listing of the directory /etc rather than a long listing of all the files within /etc



```
andrew@D630:~$ ls -ld /etc
drwxr-xr-x 137 root root 12288 Sep  7 16:08 /etc
andrew@D630:~$
```

- Pipes
  - The command `ls` does not stop here in identifying file types
  - a long listing with `ls -l` will identify the file type as well
  - The very first character before the permissions is an indicator of the file type
    - **-** Indicates a regular file
    - **d** Indicates a directory
    - **l** Indicates a symbolic link
    - **c** Indicates a character device, a terminal
    - **b** Indicates a block device, a disk drive
    - **s** Indicates a socket, network connection
    - **p** Indicates a named pipe
      - ◆ Pipes are communication processes between applications
      - ◆ The simplest form of piping is with unnamed pipes
      - ◆ `cat file | less` is an example of an unnamed pipe
- Using `mkdir`
  - Commonly used in Linux is `mkdir`, the command to create directories
  - Really important when creating, moving, deleting files
  - The **-m** option is there and you can create the directory **and** set the permissions in one
    - **-v** : provides positive feedback that the directory as created
    - **-m** : Sets the permissions of the directory, don't forget the sticky bit that ensures users can delete only the files they own and the group ID bit when set on a directory can control file group ownership
    - **-p** : creates the parent directory if required
    - **-Z** : set the SELinux context
- Using `rmdir`
  - Removes a directory, but can only remove empty directories
  - If you need to remove a directory and its contents then you use the **rm** command
    - Used to delete files
    - Can delete a directory and its content with **rm -rf** used against the target directory
    - Take great care when logged in as root, the Linux super-user, `rm -rf` can delete all files on your system
- The Copy Command or `cp`
  - In copying a file the original file remains intact and new copy is made in the target directory
  - Using the **-R** option we can recursively copy which is including files from subdirectories
  - Using the **-i** option, for interactive, we will be prompted before files are copied
  - Other options include:
    - **-a** : Is probably well known as the archive option, maintaining ownership of the files, but did you know it also turned on the recursion option so that sub-directories are included.
    - **-b** : If a target file is overwritten then back that file up and append a `~` to the end of the name.
    - **-n** : no-clobber, do not overwrite target file
    - **-u** : update the target file if source file is newer
    - **-s** : create symbolic links as targets
- Moving with `mv`
  - Using **mv** (/bin/mv) you can move or rename files
  - The following command renames the files as the file has not been placed in another director
    - `mv file1 file1.txt`
  - The file is renamed from `file1` to the new name `file1.txt`
  - If you use the following command then the file is moved from `/data` to the directory `/salesdata`, as there is no new basename to the file you leave that blank in the second argument

- `mv /data/file /salesdata/`
- If the move is on the same hard drive then no new data file is created
- A move on the same disk partition amounts only to a change in the file's meta-data
- Touch
  - You can use the command **touch** to create empty files
  - Using on an existing file you can change the date stamps with **touch -a** or **touch -m**
- Quoting
  - If you are creating a directory or a regular file you may need to quote the file name
  - For example, **mkdir new dir** will create two directories, one named new and one named dir
  - There are 3 types of quoting mechanisms:
    - Double Quotes
      - ◆ `mkdir "new dir"`
        - ◇ This will protect most characters from any shell special meaning
        - ◇ The quotes protect the space from being the argument separator
    - Single Quotes
      - ◆ `mkdir '$USER dir'`
        - ◇ The single quotes work in the same way as double quotes except that single quotes protect all characters from the shell
        - ◇ This means that should you want to include a variable within the quotes the contents of the variable would not be expanded and the directory name would be named after the variable name
        - ◇ `mkdir "$USER dir"` with double quotes would create the directory *andrew dir*
        - ◇ `mkdir '$USER dir'` with single quotes would create the directory *\$USER dir*
    - Backslash
      - ◆ This escapes the character immediately following the slash
      - ◆ Only that single character is protected
      - ◆ With a variable in the directory name as well as the space you could use the following code to create *andrew dir*
        - ◇ `mkdir $USER\ dir`
- Case sensitivity
- Simple globbing and quoting
  - Globbing provides techniques to collectively group files by using elements in their names as the criteria
  - Using different globbing techniques can display different collections of files
    - **ls \*.py** : show files with the extension py
    - **ls \*.[!p]** : show files where the extension does not start with a p
    - **ls ??.\*** : show files where the name starts with two characters followed by a dot and then any characters
  - **\*** refers to zero or more instances of any character
  - **?** Refers to exactly a single character

#### Terms and Utilities:

- mv, cp, rm, touch
- mkdir, rmdir

## Topic 3: The Power of the Command Line (weight: 9)

### 3.1 Archiving Files on the Command Line

<https://www.theurbanpenguin.com/linux-essentials-31-archiving-files-from-the-linux-command-line/>

<https://youtu.be/Ky6t89FXDxQ>

**Weight: 2**

**Description:** Archiving files in the user home directory.

#### Key Knowledge Areas:

- Files, directories
  - TAR (Tape Archive)
    - The archive itself is a single file that can represent many files
    - Common tar options:
      - **-c** : create a new archive

- `-x` : extract an archive
- `-t` : verify or *test* and archive
- tar files do not have to be compressed but they often are
- even if it is **not zipped up** it will often **consume less disk space** than the files stored individually
- Consider the following screenshot and look at the size of the directory and then create a tar archive file, uncompressed, and view the size of the tar file: it is smaller

```
andrew@D630:~$ du -sh labs
52K    labs
andrew@D630:~$ tar -cf labs.tar labs
andrew@D630:~$ du -h labs.tar
20K    labs.tar
andrew@D630:~$
```

- The output shows the directory to be 52K and the tar archive to be just 20K and no compression has been used
- This relates to the way the filesystem uses blocks of disk space
- Each new file has to start with its own new block
- Often the block size is 4KB; this means for each 1KB file, for instance, will consume 4KB of disk space
- If we combine these files into one file, a TAR file, then less space can be used to store the same amount of data
- Creating a TAR file
  - We normally include `.tar` as the last four characters of the file name so we easily identify this type of file
  - The option `-f` specifies the file name and must be followed by the same
    - `tar -cf labs.tar labs`
    - In this example we archive the `labs` directory within current directory
    - The target file for the archive is: `labs.tar`, also within the current directory
    - The source directory `labs` remains intact and unaffected by the operation other than updating the *last accessed time* attribute of each file included in the archive
    - In order to back up a file it has to be read, hence the last accessed time of each file we archive will be updated to the time of the backup
- Viewing a TAR file
  - Once you have created the file you can verify the file contents with the `-t` option
    - `tar -tf labs.tar`
  - A large TAR file may be ready directly with the command **less**
  - This allows for the file contents to be paged through without the explicit use of `tar` and `less` together
    - `less labs.tar`
- Extracting a TAR file
  - to extract the complete archive the command makes use of the `-x` option
    - `tar -xf labs.tar`
  - The files will expand within the current directory unless the option `-P` is used both when the archive is created and expanded, in which case the files are expanded to the full path of the original files
  - If there are concerns that you may overwrite files then a couple of options exist that may help
    - `-k` : prevents existing files being overwritten
    - **–keep-newer-files** : will not overwrite if the target file is newer than the archive file
  - Should you want to extract only a single file or certain files from the archive you could use code similar to the following:
    - `tar -xf labs.tar labs/file.sh`
    - This would extract just the single file from the archive
    - you can use the `-t` option if you need to confirm the path to the file in the archive
- Archives, compression
  - Compressing the archives
    - archives can be compressed and uncompressed within the same TAR process
    - Options
      - `-z` : uses `gzip` for compression and `gunzip` for decompressing the file
      - `-j` : uses `bzip2` for compression and `bunzip2` for decompressing

- `tar -czf labs labs.tgz`
  - creates the zipped archive
  - the option `-z` must be used and with `-t` for viewing
- `tar -xzf labs.tgz`
  - extracts the archive
  - the option `-z` must be used and with `-t` for viewing
- Common endings for file names
  - `.tar` : indicates an uncompressed file
  - `.tar.gz` or `.tgz` : indicates a file compressed with `gzip`
  - `.tar.bz2` or `.tbz2` : indicates a file where `bzip2` was used to compress the archive

#### Terms and Utilities:

- `tar`
- Common `tar` options
- `gzip`, `bzip2`
- `zip`, `unzip`

### 3.2 Searching and Extracting Data from Files

<https://www.theurbanpenguin.com/32-searching-and-extracting-data-from-files/>

<https://youtu.be/fcWWfOYQPG0>

#### Weight: 3

**Description:** Search and extract data from files in the home directory

#### Key Knowledge Areas:

- Command line pipes
  - Two types of pipes:
    - Unnamed
      - to make use of an unnamed pipe, use the vertical bar `|` between two commands
      - `ls -l | wc -l`
        - ◆ The above command in a command line pipe, the output of `ls` is sent to the input the command `wc`, in this case, counting the lines of output from `ls`
        - ◆ This is also known as an unnamed pipe as it is created on the fly without the existence of a pipe file
        - ◆ This is convenient for us on the command line but not so convenient for applications to be able to communicate
        - ◆ This is where **named pipes** can be used where the applications can create pipe files and connect to each other by means of these pipes
        - ◆ The pipe files will never store data but marshals data, (controls the data movement), from one application to another
        - ◆ We can create our own named pipes using `mkfifo (/usr/bin/mkfifo)`
        - ◆ These are known as named pipes as they are represented by files of type `PIPE` in the file system and as such have a name
        - ◆ As an example the **LastPass** password manager uses a named pipe to communicate with Firefox on my Ubuntu system
        - ◆ This can be seen by using the `find (/usr/bin/find)` command to search for files of type `p`:
          - ◇ `find /home -type p 2> /dev/null`
- Named
  - commonly used between processes on your PC (one application talking to another)
- I/O re-direction
  - Redirection takes the output of a command and sends the output to a text file
  - Alternatively a command may redirect a text file to its input using the file as input.
  - Each command has three channels that can be used for redirection:

```
andrew@D630:~$ find /home -type p 2> /dev/null
/home/andrew/.lastpass/pipes/lastpassffplugin
andrew@D630:~$
```

- Standard Input : Channel 0
  - Standard Error : Channel 1
  - Error Output : Channel 2
  - We only need to use the channel number when redirecting error output; the symbol < indicates standard input when used without a number and > represents standard output without a number in use
  - As such:
    - **cat < file1** : file1 is read into standard input for the command cat
    - **ls /etc > file1** : the standard output of ls is sent to file1, errors are shown on the screen and not redirected
    - **ls /etc 2> file1** : Standard output is shown to the screen but errors are written to file1
  - We can use the >> symbols to append to files or create the files if they do not exist
  - When using the single greater than symbol > we can create the file and overwrite the file if it exists
  - If you are concerned about overwriting existing files in error you may set the shell option *noclobber*
  - When set, new files can be created but if the file exists normal operation will not permit you to overwrite the existing file
  - Using >| allows the file to be over-written
  - The noclobber option may usually be set in a login script or from the command line
    - set -o noclobber
      - The -o option sets the option to on
    - set +o noclobber
      - The option +o turns the option off. To view the current setting you can use the command
    - set -o
      - The above command will show all settings and their current state
    - Taking what we have learned about piping, we now know that it is possible to pipe the output of the **set** (shell built-in command) command to **grep**(/bin/grep) which can search then for the particular option we wish to view
- ```
andrew@D630:~$ set -o | grep noclobber
noclobber      off
andrew@D630:~$
```
- Currently the option is disabled on the system as we can see from the above graphic
  - If you want this permanently added to your environment consider turning the option on in your personal login script: **.bashrc** in your home directory
- Basic Regular Expressions ., [ ], \*, ?
    - The command **grep** (/bin/grep) becomes a simple tool that we can make use of both practically in every day Linux usage as well as here in the course to help demonstrate **regular expressions**
    - To test regular expressions fully we may want to use **egrep** (/bin/egrep) or more simply **grep -E** to allow for extended regular expression matches
    - If we need to view the top of a file we can use **head** (/usr/bin/head) and should we need to view the end of a file we can use **tail** (/usr/bin/tail)
    - we can use **cat** (/bin/cat) and **tac** (/usr/bin/tac)
      - cat list or concatenates the file from top to bottom
      - tac from bottom to top
      - If your focus is on the bottom of the file use cat, you will be left at the bottom of the file
      - If your focus is on the top use tac as you will be left at the top of the file
    - The command **wc** (/usr/bin/wc) can be used to count the lines, words and characters in a file
    - wc used without options and the filename as the argument will show all three
      - wc test.txt
      - wc -l test.txt
        - counts just lines
      - wc -w test.txt
        - counts just words
      - wc -c test.txt
        - counts just characters

- The command **cut** (/usr/bin/cut) can be useful where viewing every field in a file is not required
- We may only want to see certain fields even with the output of a command we can pipe the output to cut
- Suppose we only need the line count from wc not the file name:
  - `wc -l test.txt | cut -d " " -f1`
- With cut we use the **-d** option to say that the output is space delimited and the **-f** option to display only the first field

#### Terms and Utilities:

- grep
- less
- cat, head, tail
- sort
- cut
- wc

### 3.3 Turning Commands into a Script

<https://www.theurbanpenguin.com/33-turning-commands-into-a-script/>

<https://youtu.be/HMpB28l1sLc>

<https://youtu.be/8Ub4I6WKpm0>

<https://youtu.be/yxQx4R4WF04>

[https://youtu.be/hHQR8X4U\\_jE](https://youtu.be/hHQR8X4U_jE)

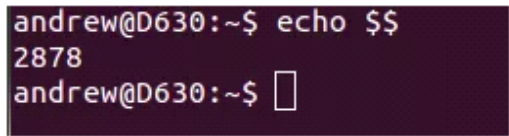
**Weight: 4**

**Description:** Turning repetitive commands into simple scripts.

#### Key Knowledge Areas:

- Basic shell scripting
  - Writing your First Script
    - When turning commands into scripts we often expect to print **Hello World**
    - As a change, we can print the Process ID or PID that the script runs in using the variable **\$\$**
    - We will also learn that scripts will normally start with the **shebang**, identifying the command line shell to use when executing
    - This is a commented line, starting with a hash #, but is read when the script runs
 

```
#!/bin/bash
#!/bin/sh
#!/bin/ksh
```
    - The above lines could be used as the shebang
      - The first being Bash , then the Bourne Shell and the final would be if we wanted to run the script in the Korn shell
      - Only one would be used per script and, if used, the shebang must be the first line of the script
  - System Variables
    - Certain system variables exist for us to use
    - **\$\$** represents the PID of the script, for long running scripts this may be useful to write out to a file so if we need to send a signal to end the script we know the process to send the signal to



```
andrew@D630:~$ echo $$
2878
andrew@D630:~$
```

- From the above graphic we can see the output from the command line, within a script we may use something like this
 

```
#!/bin/bash
echo $$ > /var/run/script.pid
```
- Executing the script, the file, script.pid, would contain the process id of the script
- This would only be useful if the script were to be long lived
- Other system variables we may use
  - **\$0** : The script name
  - **\$1** : The first argument passed to the script

- **\$#** : The number of arguments passed to the script
- **\$\*** : The list of arguments passed to the script
- Passing Arguments to a Script
  - Building up a script so that it may accept arguments we could write something similar to the following code:
 

```
#!/bin/bash
COUNT=$(wc -l $1)
echo $COUNT
```
  - Executing the script with the hosts files as arguments would look like this:
 

```
scriptname.sh /etc/hosts
```
  - We would count the number of lines in the local hosts file and print the count out to the screen
- Conditional Statements
  - The previous code identifies an issue that we may have; what if the user input just **/etc** rather than a regular file
  - This is where we could add in IF or other conditional statements:
 

```
#!/bin/bash
if [ -f $1 ]
then
COUNT=$(wc -l $1)
else
echo "$1 is not a file"
exit 53
fi
echo $COUNT
```
  - The script tests that the supplied argument is a regular file
  - If it is we can run the word count if it is not we echo out a warning and exit the script with an error code
  - We can now start to see the power of turning commands into scripts
- Awareness of common text editors
  - Using Vi
    - Vi or Vi Improved (vim) is one of the classical editors in Linux and Unix
    - It may be difficult to use at first; however is very powerful and quick to use once mastered
    - Most people only use a little of its functionality but that can be said of all editors really
    - Understanding the basics of this editor is going to be useful as most distributions are going to ship with this editor
  - Pico and Nano
    - Pico, nano and joe are all command line editors but with the addition of a basic menu system
    - This menu helps when you are starting but also can slow the process down and certainly do not have the features available in vi

#### Terms and Utilities:

- **#!** (shebang)
- **/bin/bash**
- Variables
- Arguments
- for loops
- echo
- Exit status

## Topic 4: The Linux Operating System (weight: 8)

### 4.1 Choosing an Operating System

<https://www.theurbanpenguin.com/41-choosing-an-operating-system/>

<https://youtu.be/PXMLYz8Xxa8>

**Weight: 1**

**Description:** Knowledge of major operating systems and Linux distributions.

**Key Knowledge Areas:**



- Windows, Mac, Linux differences
  - Linux Distributions
    - With Linux, you have many choices and again this is a controversial area
    - Some Linux distributions include non-open source software
    - To some, this is not within the ideals of Linux and you should avoid such distributions
    - Other distributions provide support at a cost
      - Often the Enterprise version such as Red Hat Enterprise Linux or SUSE Linux Enterprise Server
  - OS-X
    - MACs or Apple PCs have traditionally had their market in design and desktop publishing but during the early 2000, OSX broke that mold
    - 2001 saw the release of MAC OSX 10.0 “Cheetah”
      - Based on BSD Unix with some elements from NeXT that Steve Jobs had developed from an earlier position
      - This became NeXTSTEP in OSX and the GUI took off and over 10 years later the OS is still very successful.
    - Apple has control over the hardware and the OS, they sell the Desktop or Laptop with the OS so they are able to manage the complete supply chain
      - This helps for the integration and reliability
      - So even now where the OSX OS saw then transition to the Intel platform their you but the complete Desktop package from Apple.
    - As OSX has UNIX at its core much of the command line functionality that we have in Linux is there also in OSX
    - the tools that we use with find, cut, sed, awk are all there with OSX
  - Windows
    - Windows certainly still rules the corporate world and if we use an OS at work we are more likely to use it at home
    - In the same way, if we use a certain OS at school and college then we are likely to continue with the Operating System that we are already familiar with
    - SO even though we normally have to pay for Windows, we still use it because this is what we learned at school
    - If we were to use and learn Linux at school, the uptake of Linux in the home will be more
    - We need to understand the culture of free and open at an early age
    - To be able to build a true open digital economy of the future we need to loosen the grip that proprietary software has on the world
    - Choosing an operating system needs to start at school and the choice has to be there
    - While Windows is seemingly ill-suited to education, it still has a lot to offer in other areas
    - As a corporate desktop, the well-established desktop management tools and skill needed to manage 100's of units is going to be hard to overcome
    - However, that will start with experience at home and in the classroom
    - Microsoft Windows and commercial software will always have a place and they have left their mark in history as revolutionizing the PC industry
    - Where then, does Linux fit in? Although Microsoft has undoubtedly revolutionized the PC industry, one must also recognize that the PC desktop is on its way out
    - Last year (2012) IDC recorded a global drop in PC sales by 14%
    - This marks the first ever recorded drop in sales
    - This can be put down to the rise of smart phones and tablets, and a shift to a more mobile form of computing
    - The mobile platform is where Linux performs equally as well as other OS's, if not better as Android
    - As Android grows in popularity and desktop improvements with Desktop Linux OSs such as Unity in Ubuntu and KDE4 in openSUSE, we can see that Linux continues to develop in the home and the corporate market
    - For the server market even Microsoft sells SUSE Linux to their customers where they feel it an appropriate solution
    - Microsoft is, in fact, the largest single customer to SUSE

- It is unlikely that you will ever work in a Corporate environment that does not have Linux as an OS somewhere, even if it the embedded OS in a switch or router
- The power of the command line and the way in which the OS can be stripped to the bare essentials needed to dot the job and be frugal on resources has made the Linux Server a popular OS in businesses of all sizes and has led to a rise in system admin and support roles in for Linux
- The LPI quite rightly focuses on command line skills in this course as that is where the position lies at the moment, with less support positional available for desktops
- The speed of the command line and the fact that tasks can be scripted make it an obvious choice over the GUI
- The difference in creating users from the command line compared to the GUI, the time factor is enormous even for a single user
- This is also recognized in the Microsoft world with the advent of PowerShell and GUI-less Server Core version of Server 2012
- Distribution life cycle management

#### Terms and Utilities:

- GUI versus command line, desktop configuration
- Maintenance cycles, Beta and Stable

## 4.2 Understanding Computer Hardware

<https://www.theurbanpenguin.com/42-understanding-computer-hardware/>

<https://youtu.be/GQPKxKKEjrY>

### Weight: 2

**Description:** Familiarity with the components that go into building desktop and server computers.

#### Key Knowledge Areas:

- Hardware
  - procfs
    - Firstly we can look at how the **/proc** directory, also known as the procfs, which holds so much information about the running system
    - Taking a look at the **/proc/partitions** file we will see that it lists partitions, perhaps this is more easily shown with the command `fdisk`
    - This is feature of both Linux and the procfs
    - Developers will write commands to format and extract information stored within files in the procfs

```
andrew@D630:~$ cat /proc/partitions
major minor #blocks name
11        0    1048575 sr0
8         0   117220824 sda
8         1   115133440 sda1
8         2         1 sda2
8         5    2084864 sda5
andrew@D630:~$
```

- Another fine example of this is to use the command **lsmod**
- Used to list the loaded Kernel modules
- This command reads from the file **/proc/modules** and formats the data for us

```
andrew@D630:~$ head -n 3 /proc/modules
rfcomm 42641 0 - Live 0x0000000000000000
bnep 18036 2 - Live 0x0000000000000000
parport_pc 28152 0 - Live 0x0000000000000000 (F)
andrew@D630:~$
```

- Kernel Modules
  - Kernel Modules act as the interface between the Operating System and the hardware
  - Some people will refer to these modules as drivers
  - As the hardware is detected the driver is usually automatically loaded; however we can manually load

- and unload modules with the command **modprobe**
- Which is in turn an interface to the commands **insmod** and **rmmod**
- To load the parallel port driver you could use:
 

```
modprobe parport_pc
```
- Alternatively you could use **insmod**

```
Insmod <full path to module>.ko
```
- It is the full path that becomes problematic and why **modprobe** is such a useful tool
- To unload drivers we use the **-r** option with **modprobe** (the **-r** is remove)
- We can also use **rmmod** but we do not need the full path to remove a module
 

```
modinfo parport_pc
```

or

```
rmmod parport_pc
```
- To obtain information on a driver such as its purpose, author and options we can use the **modinfo**
- The module does not need to be loaded to see this information
 

```
modinfo parport_pc
```
- Using **HWINFO**
  - The command **hwinfo** list the hardware and the driver used to access the hardware, including the command and any options used in loading the driver
  - **hwinfo** is a very useful reporting and diagnostics tool that you shouldn't overlook
 

```
hwinfo --help
```

```
hwinfo --netcard
```

```
hwinfo --gfxcard
```

#### Terms and Utilities:

- Motherboards, processors, power supplies, optical drives, peripherals
- Hard drives and partitions, `/dev/sd*`
- Drivers

### 4.3 Where Data is Stored

<https://www.theurbanpenguin.com/43-where-data-is-stored/>

<https://youtu.be/N7PyeNulBVE>

#### Weight: 3

**Description:** Where various types of information are stored on a Linux system.

#### Key Knowledge Areas:

- Programs and configuration, packages and package databases
- Processes, memory addresses, system messaging and logging

#### Terms and Utilities:

- **ps**, **top**, **free**
- **syslog**, **dmesg**
- `/etc/`, `/var/log/`
- `/boot/`, `/proc/`, `/dev/`, `/sys/`

### 4.4 Your Computer on the Network

<https://www.theurbanpenguin.com/44-your-computer-on-the-network/>

<https://youtu.be/iMepiyIGATg>

#### Weight: 2

**Description:** Querying vital networking configuration and determining the basic requirements for a computer on a Local Area Network (LAN).

#### Key Knowledge Areas:

- Internet, network, routers
  - Internet Protocol
    - An IPv4 address would look similar to this: **192.168.1.100**, the address is written in what we call a dotted decimal format that represents 32 bit address space
    - An address made up from 32 ones or zeros, computer bits
    - Along with the address there is something called the **subnet mask**, this identifies which part of the

address represents the networks and what part the host

- In the simplest forms having a subnet mask of **255.255.255.0** applied to the previous IP address would give us a network of 192.168.1.0 and a host 100 on that network
- If a host is on the same network as the originating device communication can happen directly; if it is seen as being on a different network then, usually this would mean that communication would be mediated by the **default gateway** or router on the network
- An IP Address and subnet mask are required, the default gateway is optional
- Without a default gateway though you may struggle to connect to hosts that are not on the same network
  - **IP Address:** 32 bit Address to locate the computer's location or address on the network
  - **Subnet Mask:** 32 bit address denoting which parts of an address are the network and which parts of an address are the address of the host
  - **Default Gateway:** Router to use where the network address is not known

#### ○ IFCONFIG

- To display the IP address configuration of a host we can use the tools ifconfig or ip
- Traditionally Linux and UNIX administrators have become used to using **ifconfig (/sbin/ifconfig)** to both display and set an IP address
- As this is in the **sbin** directory it may not be in the PATH variable for standard users and as such becomes an admin only tool even for just displaying information
- To display information for the first ethernet card, **eth0** use the command:

ifconfig eth0

```
php:~/Desktop # ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0C:29:AA:71:4E
          inet addr:192.168.9.128  Bcast:192.168.9.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:feaa:714e/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:22 errors:0 dropped:0 overruns:0 frame:0
          TX packets:36 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2931 (2.8 Kb)  TX bytes:3250 (3.1 Kb)
          Interrupt:19 Base address:0x2000
```

#### Display IP address with /sbin/ifconfig

- You can use ifconfig to both display and set address information
- Permanent address configuration has to be made through configuration files not with commands like ip or ifconfig

#### ○ IP

- The new kid on the block is ip, (**/sbin/ip**), this can be used to display address configuration with:  
ip address show eth0
- This can be abbreviated to:  
ip a s eth0
- Again it can be used to configure the address as well as displaying it
- like ifconfig these changes are not persistent if not made in the configuration files for the network interface
- Although the command is in the /sbin directory there is also a symbolic or soft link in the /bin making it easily accessible to standard users who can use it to display information

```
bob@php:~> ls -l $(which ip)
lrwxrwxrwx 1 root root 8 Aug 10 14:51 /bin/ip -> /sbin/ip
bob@php:~>
```

#### ip has a symlink in the /bin directory

- The real winner with the ip command is that it can be used to manage more than just the ip address, routes links, multicast addresses, ARP tables and more can be managed with ip
  - ip route show
  - ip neighbor show

- ip maddr show
- ip link show
- The IP address configuration can be **STATIC** or **DYNAMIC**
- client computers are often configured with dynamic addresses from a DHCP or **Dynamic Host Control Protocol**
- servers are often configured with static addresses
- With a dynamic address you can, for example, connect your android phone to your WiFi network at home to gain internet access
- connecting to the internet at your favorite coffee shop is just as easy, no changes to the configuration is required as the assignment is dynamic
- the address, subnet mask and default gateway comes from the DHCP server on each network you connect to
- **ROUTE**
  - Like ifconfig the command route (**/sbin/route**), is the tradition command used in UNIX and Linux to display route information for a host
  - Both ip route show and the command route on its own will display the route table on the host
  - Usually the most important route is the default route, if no other route can be found to a network the default route is used

```
php:~/Desktop # route
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref
default          192.168.9.2    0.0.0.0         UG    0     0
loopback         *              255.0.0.0       U     0     0
link-local       *              255.255.0.0     U     0     0
192.168.9.0      *              255.255.255.0   U     0     0
php:~/Desktop #
```

#### Displaying the route table with /sbin/route

- Querying DNS client configuration
  - **Name Resolution**
    - The host will be configured with and IP Address but mainly users will find it easier to access hosts using DNS names (**Domain Name Server**)
    - This way we can access our favorite websites with just a name, [www.theurbanpenguin.com](http://www.theurbanpenguin.com), rather than having to know its address
    - The system has to be configured with the Ip address of the DNS server or servers
    - This is most often in the file **/etc/resolv.conf**.
    - The file uses the nameserver directive followed by the address of the server
    - The address of the servers can be delivered by DHCP for dynamically configured hosts

Nameserver 192.168.1.1

```
[andrew@labs ~]$ cat /etc/resolv.conf
# Generated by NetworkManager
domain members.linode.com
search members.linode.com
nameserver 109.74.194.20
nameserver 109.74.193.20
nameserver 109.74.192.20
[andrew@labs ~]$
```

#### /etc/resolv.conf

- Querying Network configuration

#### Terms and Utilities:

- route, ip route show
- ifconfig, ip addr show
- netstat, ip route show
- /etc/resolv.conf, /etc/hosts
- IPv4, IPv6

- ping
- host

## Topic 5: Security and File Permissions (weight: 7)

### 5.1 Basic Security and Identifying User Types

<https://www.theurbanpenguin.com/51-basic-security-and-identifying-user-types/>

<https://youtu.be/xWxObO2-6Ko>

#### Weight: 2

**Description:** Various types of users on a Linux system.

#### Key Knowledge Areas:

- Root and Standard Users
  - The predominate administrative account on Linux systems is the root account with the user ID (UID) of 0
  - To manage the Linux system you will need access to this account either directly or via **sudo**
  - sudo is the preferred method of delegated administration as, in this way, the administrative users do not need access to the root password
  - Logging into the system directly as root or using the substitute user command, **su**, knowledge of the root password is required
  - Each Linux host must, at the minimum, have a local root account defined in the `/etc/passwd` file
  - The local user account store is `/etc/passwd`, passwords, on the other hand passwords are usually held in the `/etc/shadow` file
  - Each user, including the root user will need both a UID and a GID group ID
  - Users must belong to a minimum of one group; some systems such a Red Hat run a private group system where users belong to their own private groups, other systems including SUSE have a public group system where users belong to a shared groups : users
  - Local groups are recorded in the file `/etc/group`
  - With root privileges users can be created and managed with the command **useradd** and groups with **groupadd**
  - Even though you would be expected to use the tools provided to manage users and groups there is nothing stopping changes being made directly to the appropriate file

```
root:x:0:0:root:/root:/bin/bash
sshd:x:71:65:SSH daemon:/var/lib/ssh:/bin/
suse-ncc:x:105:108:Novell Customer Center U
uucp:x:10:14:Unix-to-Unix CoPy system:/etc/
uidd:x:102:104:User for uidd:/var/run/uui
wwwrun:x:30:8:WWW daemon apache:/var/lib/ww
andrew:x:1000:100:~/home/andrew:/bin/bash
```

#### `/etc/passwd` file

- These are text files writable by the root account. The seven fields of the passwd file are delimited with a colon and are described as:
  1. user name
  2. password or a single x denotes the password is stored in `/etc/shadow`
  3. UID
  4. GID
  5. Comments
  6. Home directory path
  7. Default user shell, (command line environment)
- To display information about a user account, whether your own or another account the command **id**, (`/usr/bin/id`)
- By default the command will display the user name uid and gid and secondarygroups, but more specific information can be honed in upon with additional switches
- The command **finger**, (`/usr/bin/finger`), can be used to display further information about users account information
- To return information on currently logged in users, perhaps if you need to bring a server down for maintenance, the use of the commands **who** (`/usr/bin/who`) or **w** (`/usr/bin/w`) are useful

- Controlling access to sudo, (/usr/bin/sudo) ,and what you are allowed to do with it, is a task that the root user will achieve through editing the file **/etc/sudoers**
- So that mistakes are less likely to occur, root is encouraged to edit the file using visudo (/usr/sbin/visudo) ; as this program closes the file is syntax checked helping prevent errors
- Users can be delegated rights to run certain commands through the **/etc/sudoers** file; this way knowledge of the root password is not required when administering the host but the commands must be prefaced with sudo
- On some systems, such as Ubuntu, the root account password is not shown during installation and all tasks are managed thorough sudo
- This, in many ways is a correct administration model avoiding directly accessing the root account

```
andrew@tup:~> sudo /usr/sbin/useradd -m bob_
```

- For ease of access to many administration commands consider adding the /sbin and usr/sbin directories into the PATH variable of your delegated administrators
- In this way the previous command in the screenshot could be reduced to : **sudo useradd -m bob**
- When required if you have access to the root password you may use the **su** (/bin/su) command to substitute your current user id with the root user ID (note the command su can be used to access any account you know the password to , not just root)
- It is possible to disallow remote access via SSH to root, you can still logon on as a standard account and su to root as required

- System users

#### Terms and Utilities:

- /etc/passwd, /etc/group
- id, who, w
- sudo, su

## 5.2 Creating Users and Groups

<https://www.theurbanpenguin.com/52-creating-users-and-groups/>

<https://youtu.be/ewrZi44Kd68>

**Weight: 2**

**Description:** Creating users and groups on a Linux system.

#### Key Knowledge Areas:

- User and group commands
  - Migrating Passwords to and from /etc/shadow
    - Traditionally the users password was stored with their account details in the /etc/passwd file but more recently the passwords have been stored in the /etc/shadow file
    - Either way they were encrypted but the shadow file is only accessible by the root user whereas the passwd file needs to be readable by all
    - Passwords can be migrated to and from the /etc/passwd file with the **pwconv** (/usr/sbin/pwconv) and **pwunconv** (/usr/sbin/pwunconv) commands
  - Public and Private Groups
    - Red Hat and CentOS use a private group scheme where each user is a member of their own private group
    - If we create a user named bob then a corresponding group also named bob is created with the user bob as the only member
    - Other distributions such as SUSE use a public group system and it would be normal for users to default to belonging to the users group
    - If you are using a Red Hat style distribution with private groups then using the **-N** switch with useradd will disable the private group for that user and they will belong to the normal users group
      - useradd -N joe** : will create the user joe as a member of the default users group
      - useradd joe** : will create the user and group bob with the user joe as a member of the private group joe

- User IDs



- Managing Users and Groups
  - User management is maintained by:
    - useradd
    - usermod
    - userdel
  - Group management with:
    - groupadd
    - groupmod
    - groupdel
  - Passwords are managed with:
    - passwd
    - change

#### Terms and Utilities:

- /etc/passwd, /etc/shadow, /etc/group, /etc/skel/
- id, last
- useradd, groupadd
- passwd

### 5.3 Managing File Permissions and Ownership

<https://www.theurbanpenguin.com/53-managing-file-permissions-and-ownership/>

<https://youtu.be/7BP4GtBZv1M>

#### Weight: 2

**Description:** Understanding and manipulating file permissions and ownership settings.

#### Key Knowledge Areas:

- File/directory permissions and owners
  - For this objective we take a look at the permissions or “*mode*” of a file, (or directory)
  - The permissions can be set for the:
    - **user** owner
    - **group** owner
    - and **others**
  - The available permissions are simply read, write and execute
  - Execute permissions to a directory mean that you can enter the directory, or more specifically *traverse the directory link*, the word *enter* works perfectly well for me
  - The permissions for a file are normally read from the long listing of ls (/bin/ls)
  - The permissions shown from ls are displayed in symbolic form, ie rwx
  - Permissions can be set in symbolic format or octal format such as 755
  - If you would like to display the permissions in octal format, the command stat (/usr/bin/stat) could be used

**stat -c %a /etc (octal)**

**stat -c %A /etc (symbolic)**

```
php:~ # stat -c %A /etc
drwxr-xr-x
php:~ # stat -c %a /etc
755
```

#### output from stat

- From the output above we can see the **755** permissions relate to **rwxr-xr-x**
  - We can use chmod (/bin/chmod) to set the permissions whereas we use ls -l to display permissions
  - ls -ld is needed to display the permissions for a directory
  - When using chmod there are two methods of setting permissions, with octal or symbolic notation:
    - chmod 755 file1
    - chmod u=rwx,g=rx,o=rx file
  - Ownership of files can be changed using chown (/bin/chown) to change the user owner and chgrp (/bin/chgrp)

#### Terms and Utilities:



- ls -l, ls -a
- chmod, chown

## 5.4 Special Directories and Files

<https://www.theurbanpenguin.com/54-special-directories-and-files/>

[https://youtu.be/\\_6VJ8WfWl4k](https://youtu.be/_6VJ8WfWl4k)

[https://youtu.be/l\\_1Q3DG3uoE](https://youtu.be/l_1Q3DG3uoE)

### Weight: 1

**Description:** Special directories and files on a Linux system including special permissions.

### Key Knowledge Areas:

- Using temporary files and directories
  - Organization of the File System
    - The Linux file system is separated into directories that reflect the purpose of the files stored within that directory
    - The directory **/etc** is the server configuration directory, generally users can read but not write to files in this directory
    - In addition, even the user root would not need to make changes frequently to this directory
    - The directory **/var**, on the other hand, holds log files, print spool files and perhaps database files to this directory structure would be written to frequently but by services rather than users
    - The directory **/tmp** is used by all users and they will need write access to this directory
  - Special Permissions
    - Users who can write to the directory can also delete files from the directory
    - To control deletions the special permission known as the **sticky bit** is assigned to the **/tmp** directory
    - When the sticky bit is set on a directory on the owner of the file can delete the file
    - When the sticky bit is set it shows in the space for the execution permission in the others block, where the execute permission is also set then it will show as a **lowercase t**, where execute is not set it will show as an **uppercase t**
    - To set the permission using symbolic notation this can be set with chmod  
**chmod o+t /tmp**
    - The octal permissions would be 1777 for the **/tmp** directory
    - This can be seen with the stat command

```
php:~/Desktop # stat -c %a /tmp
1777
```

- Special permissions also exist for the group and user block
- These are known as the set UID and set GID permissions
- When a file is set with these permissions, executing the file will result in the permissions granted to the user or the group are used, user permissions if the SUID bit is set, group permissions if the SGID bit is set
- If we look at the executable file wall (**/usr/bin/wall**) we see the permissions show as **rwxr-tr-x root tty** ;
- Anyone executing the file will do so as the group tty so access to terminals will be allowed
- The **SGID** bit may also be set on a directory and, in which case, it will affect the group ownership of all new files created in the directory ensuring that new files are group owned by the group owner of the directory
- The **SUID** bit is set commonly on the file **/usr/bin/passwd** so that users can change their own password and write to the **/etc/shadow** file that is writable on by root
- Hard Links
  - Hard links are files that have multiple names
  - A file name in Linux points to an inode, hard links are files inodes that have more than one file name linking to them
  - As such, hard links, do not take any extra disk space as there is no additional data used
  - File-systems also restrict the number of inodes that can be created and hard links do not create any extra data or inodes
  - Running ls -l or using the find command hard links will show as regular files, however from the output of

ls -l we can view in the 2nd field how many names link to the inode; if it is more than 1 there are hard links to the file

```
geeko@dai:~/Documents> ls -l
total 24
-rw-r--r-- 2 geeko users 17 2013-05-21 20:10 document.txt
lrwxrwxrwx 1 geeko users 12 2013-05-21 20:14 fred.txt -> document.txt
-rw-r--r-- 2 geeko users 17 2013-05-21 20:10 hello.txt
-rw-r--r-- 1 geeko users 6943 2013-05-22 10:26 New Document.ott
-rw-r--r-- 1 geeko users 6391 2013-05-22 10:26 New Spreadsheet.ots
geeko@dai:~/Documents>
```

- From the output above we can see the document.txt and hello.txt are hard links, not necessarily to each other but they both contain 2 names linked to the one inode entry in the file-system as highlighted
- Hard links are restricted to the same file-system and most file-systems do not support hard linking to directories and are less commonly used than soft links
- To create a hard link use the command ln:

**ln <target file> <link name>**

**ln document.txt hello.txt**

- Symbolic links

- Soft Links

- Symbolic links are actual links and show as the file type link within the ls -l output and that of the command file
  - If directory colors are turned on then symlinks will show as light blue but with or without colors on you will see the first character in the output of ls -l for the symlink name is "l"
  - This denotes a link
  - This is different to hard links which are regular files
  - From the following screenshot we can see that the file fred.txt is linked to document.txt and shows as a file type of link, denoted by the 1st character on the output of ls -l being l instead of - for regular files and hard links

```
geeko@dai:~/Documents> ls -l
total 24
-rw-r--r-- 2 geeko users 17 2013-05-21 20:10 document.txt
lrwxrwxrwx 1 geeko users 12 2013-05-21 20:14 fred.txt -> document.txt
-rw-r--r-- 2 geeko users 17 2013-05-21 20:10 hello.txt
-rw-r--r-- 1 geeko users 6943 2013-05-22 10:26 New Document.ott
-rw-r--r-- 1 geeko users 6391 2013-05-22 10:26 New Spreadsheet.ots
geeko@dai:~/Documents>
```

- Symlinks are very similar to **Aliases** in the MAC OS and **Shortcuts** in Windows as with their counterparts in other OSs, these are completely separate files in the file-system and so have their own name, inode and data
- The data of the symlink points to the original or target file
- With hard links they are inodes that have more than one file name linking to them
- Symlinks are more flexible in that they can link across file-systems to different partitions and drives and link to directories
- To create symlinks use the command:  
**ln -s <target> <linkname>**  
such as  
**ln -s /usr/share/doc ./doc**
- The example creates a link called doc in the current directory to the directory /usr/share/doc

#### Terms and Utilities:

- /tmp/, /var/tmp/ and Sticky Bit
- ls -d
- ln -s

## COMMAND LINES

- **SYSTEM INFORMATION**

|                                      |                                                    |
|--------------------------------------|----------------------------------------------------|
| <code>uname -a</code>                | # Display Linux system information                 |
| <code>uname -r</code>                | # Display kernel release information               |
| <code>cat /etc/redhat-release</code> | # Show which version of redhat installed           |
| <code>uptime</code>                  | # Show how long the system has been running + load |
| <code>hostname</code>                | # Show system host name                            |
| <code>hostname -I</code>             | # Display the IP addresses of the host             |
| <code>last reboot</code>             | # Show system reboot history                       |
| <code>date</code>                    | # Show the current date and time                   |
| <code>cal</code>                     | # Show this month's calendar                       |
| <code>w</code>                       | # Display who is online                            |
| <code>whoami</code>                  | # Who you are logged in as                         |

- **HARDWARE INFORMATION**

|                                    |                                                                                |
|------------------------------------|--------------------------------------------------------------------------------|
| <code>dmesg</code>                 | # Display messages in kernel ring buffer                                       |
| <code>cat /proc/cpuinfo</code>     | # Display CPU information                                                      |
| <code>cat /proc/meminfo</code>     | # Display memory information                                                   |
| <code>free -h</code>               | # Display free and used memory ( -h for human readable, -m for MB, -g for GB.) |
| <code>lspci -tv</code>             | # Display PCI devices                                                          |
| <code>lsusb -tv</code>             | # Display USB devices                                                          |
| <code>dmidecode</code>             | # Display DMI/SMBIOS (hardware info) from the BIOS                             |
| <code>hdparm -i /dev/sda</code>    | # Show info about disk sda                                                     |
| <code>hdparm -tT /dev/sda</code>   | # Perform a read speed test on disk sda                                        |
| <code>badblocks -s /dev/sda</code> | # Test for unreadable blocks on disk sda                                       |

- **PERFORMANCE AND MONITORING STATISTICS**

|                                         |                                                                                        |
|-----------------------------------------|----------------------------------------------------------------------------------------|
| <code>top</code>                        | # Display and manage the top processes                                                 |
| <code>htop</code>                       | # Interactive process viewer (top alternative)                                         |
| <code>mpstat 1</code>                   | # Display processor related statistics                                                 |
| <code>vmstat 1</code>                   | # Display virtual memory statistics                                                    |
| <code>iostat 1</code>                   | # Display I/O statistics                                                               |
| <code>tail 100 /var/log/messages</code> | # Display the last 100 syslog messages (Use /var/log/syslog for Debian based systems.) |
| <code>tcpdump -i eth0</code>            | # Capture and display all packets on interface eth0                                    |
| <code>tcpdump -i eth0 'port 80'</code>  | # Monitor all traffic on port 80 ( HTTP )                                              |
| <code>lsof</code>                       | # List all open files on the system                                                    |
| <code>lsof -u user</code>               | # List files opened by user                                                            |
| <code>free -h</code>                    | # Display free and used memory ( -h for human readable, -m for MB, -g for GB.)         |
| <code>watch df -h</code>                | # Execute "df -h", showing periodic updates                                            |

- **USER INFORMATION AND MANAGEMENT**

|                                              |                                                                                                      |
|----------------------------------------------|------------------------------------------------------------------------------------------------------|
| <code>id</code>                              | # Display the user and group ids of your current user.                                               |
| <code>last</code>                            | # Display the last users who have logged onto the system.                                            |
| <code>who</code>                             | # Show who is logged into the system.                                                                |
| <code>w</code>                               | # Show who is logged in and what they are doing.                                                     |
| <code>groupadd test</code>                   | # Create a group named "test".                                                                       |
| <code>useradd -c "John Smith" -m john</code> | # Create an account named john, with a comment of "John Smith" and create the user's home directory. |
| <code>userdel john</code>                    | # Delete the john account.                                                                           |
| <code>usermod -aG sales john</code>          | # Add the john account to the sales group                                                            |

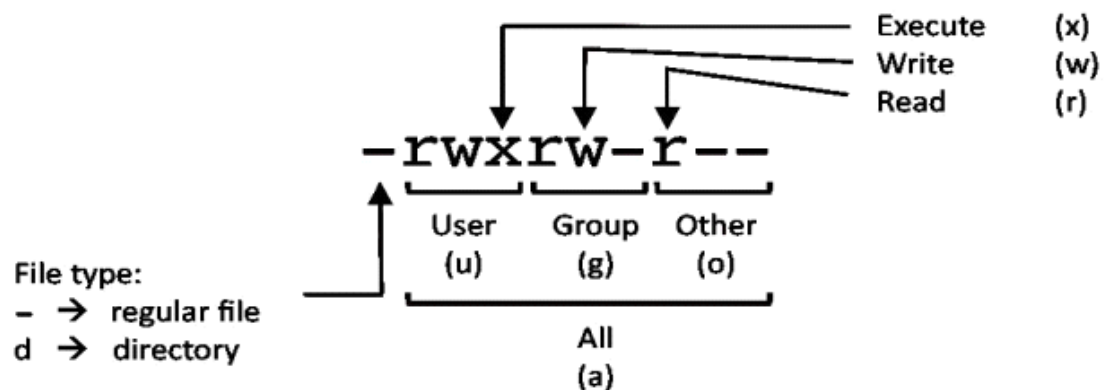
- **FILE AND DIRECTORY COMMANDS**

|                                                 |                                                                                                                                                                                        |
|-------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ls -al</code>                             | # List all files in a long listing (detailed) format                                                                                                                                   |
| <code>pwd</code>                                | # Display the present working directory                                                                                                                                                |
| <code>mkdir directory</code>                    | # Create a directory                                                                                                                                                                   |
| <code>rm file</code>                            | # Remove (delete) file                                                                                                                                                                 |
| <code>rm -r directory</code>                    | # Remove the directory and its contents recursively                                                                                                                                    |
| <code>rm -f file</code>                         | # Force removal of file without prompting for confirmation                                                                                                                             |
| <code>rm -rf directory</code>                   | # Forcefully remove directory recursively                                                                                                                                              |
| <code>cp file1 file2</code>                     | # Copy file1 to file2                                                                                                                                                                  |
| <code>cp -r source_directory destination</code> | # Copy source_directory recursively to destination. If destination exists, copy source_directory into destination, otherwise create destination with the contents of source_directory. |
| <code>mv file1 file2</code>                     | # Rename or move file1 to file2. If file2 is an existing directory, move file1 into directory file2                                                                                    |
| <code>ln -s /path/to/file linkname</code>       | # Create symbolic link to linkname                                                                                                                                                     |
| <code>touch file</code>                         | # Create an empty file or update the access and modification times of file.                                                                                                            |
| <code>cat file</code>                           | # View the contents of file                                                                                                                                                            |
| <code>less file</code>                          | # Browse through a text file                                                                                                                                                           |
| <code>head file</code>                          | # Display the first 10 lines of file                                                                                                                                                   |
| <code>tail file</code>                          | # Display the last 10 lines of file                                                                                                                                                    |
| <code>tail -f file</code>                       | # Display the last 10 lines of file and "follow" the file as it grows.                                                                                                                 |

- **PROCESS MANAGEMENT**

|                                        |                                                              |
|----------------------------------------|--------------------------------------------------------------|
| <code>ps</code>                        | # Display your currently running processes                   |
| <code>ps -ef</code>                    | # Display all the currently running processes on the system. |
| <code>ps -ef   grep processname</code> | # Display process information for processname                |
| <code>top</code>                       | # Display and manage the top processes                       |
| <code>htop</code>                      | # Interactive process viewer (top alternative)               |
| <code>kill pid</code>                  | # Kill process with process ID of pid                        |
| <code>killall processname</code>       | # Kill all processes named processname                       |
| <code>program &amp;</code>             | # Start program in the background                            |
| <code>bg</code>                        | # Display stopped or background jobs                         |
| <code>fg</code>                        | # Brings the most recent background job to foreground        |
| <code>fg n</code>                      | # Brings job n to the foreground                             |

- **FILE PERMISSIONS**



| PERMISSION  | EXAMPLE                             |
|-------------|-------------------------------------|
| U G W       |                                     |
| rwX rwX rwX | chmod 777 filename # Use sparingly! |
| rwX rwX r-X | chmod 775 filename                  |
| rwX r-X r-X | chmod 755 filename                  |
| rw- rw- r-- | chmod 664 filename                  |
| rw- r-- r-- | chmod 644 filename                  |

#### LEGEND

U = User  
 G = Group  
 W = World

r = Read  
 w = write  
 x = execute  
 - = no access

- **NETWORKING**

|                                          |                                                                  |
|------------------------------------------|------------------------------------------------------------------|
| <code>ifconfig -a</code>                 | # Display all network interfaces and ip address                  |
| <code>ifconfig eth0</code>               | # Display eth0 address and details                               |
| <code>ethtool eth0</code>                | # Query or control network driver and hardware settings          |
| <code>ping host</code>                   | # Send ICMP echo request to host                                 |
| <code>whois domain</code>                | # Display whois information for domain                           |
| <code>dig domain</code>                  | # Display DNS information for domain                             |
| <code>dig -x IP_ADDRESS</code>           | # Reverse lookup of IP_ADDRESS                                   |
| <code>host domain</code>                 | # Display DNS ip address for domain                              |
| <code>hostname -i</code>                 | # Display the network address of the host name.                  |
| <code>hostname -I</code>                 | # Display all local ip addresses                                 |
| <code>wget http://domain.com/file</code> | # Download http://domain.com/file                                |
| <code>netstat -nutlp</code>              | # Display listening tcp and udp ports and corresponding programs |

- **ARCHIVES (TAR FILES)**

|                                                |                                                          |
|------------------------------------------------|----------------------------------------------------------|
| <code>tar cf archive.tar directory</code>      | # Create tar named archive.tar containing directory.     |
| <code>tar xf archive.tar</code>                | # Extract the contents from archive.tar.                 |
| <code>tar czf archive.tar.gz directory</code>  | # Create a gzip compressed tar file name archive.tar.gz. |
| <code>tar xzf archive.tar.gz</code>            | # Extract a gzip compressed tar file.                    |
| <code>tar cjf archive.tar.bz2 directory</code> | # Create a tar file with bzip2 compression               |
| <code>tar xjf archive.tar.bz2</code>           | # Extract a bzip2 compressed tar file.                   |

- **INSTALLING PACKAGES**

|                                         |                                                              |
|-----------------------------------------|--------------------------------------------------------------|
| <code>yum search keyword</code>         | # Search for a package by keyword.                           |
| <code>yum install package</code>        | # Install package.                                           |
| <code>yum info package</code>           | # Display description and summary information about package. |
| <code>rpm -i package.rpm</code>         | # Install package from local file named package.rpm          |
| <code>yum remove package</code>         | # Remove/uninstall package.                                  |
| <code>tar zxvf sourcecode.tar.gz</code> | # Install software from source code.                         |
| <code>cd sourcecode</code>              |                                                              |
| <code>./configure</code>                |                                                              |
| <code>make</code>                       |                                                              |
| <code>make install</code>               |                                                              |

- **SEARCH**



- |                                              |                                                                   |
|----------------------------------------------|-------------------------------------------------------------------|
| <code>grep pattern file</code>               | <code># Search for pattern in file</code>                         |
| <code>grep -r pattern directory</code>       | <code># Search recursively for pattern in directory</code>        |
| <code>locate name</code>                     | <code># Find files and directories by name</code>                 |
| <code>find /home/john -name 'prefix*'</code> | <code># Find files in /home/john that start with "prefix".</code> |
| <code>find /home -size +100M</code>          | <code># Find files larger than 100MB in /home</code>              |
- **SSH LOGINS**

|                                    |                                                        |
|------------------------------------|--------------------------------------------------------|
| <code>ssh host</code>              | <code># Connect to host as your local username.</code> |
| <code>ssh user@host</code>         | <code># Connect to host as user</code>                 |
| <code>ssh -p port user@host</code> | <code># Connect to host using port</code>              |
  - **FILE TRANSFERS**

|                                                |                                                                                                            |
|------------------------------------------------|------------------------------------------------------------------------------------------------------------|
| <code>scp file.txt server:/tmp</code>          | <code># Secure copy file.txt to the /tmp folder on server</code>                                           |
| <code>scp server:/var/www/*.html /tmp</code>   | <code># Copy *.html files from server to the local /tmp folder.</code>                                     |
| <code>scp -r server:/var/www /tmp</code>       | <code># Copy all files and directories recursively from server to the current system's /tmp folder.</code> |
| <code>rsync -a /home /backups/</code>          | <code># Synchronize /home to /backups/home</code>                                                          |
| <code>rsync -avz /home server:/backups/</code> | <code># Synchronize files/directories between the local and remote system with compression enabled</code>  |
  - **DISK USAGE**

|                       |                                                                                          |
|-----------------------|------------------------------------------------------------------------------------------|
| <code>df -h</code>    | <code># Show free and used space on mounted filesystems</code>                           |
| <code>df -i</code>    | <code># Show free and used inodes on mounted filesystems</code>                          |
| <code>fdisk -l</code> | <code># Display disks partitions sizes and types</code>                                  |
| <code>du -ah</code>   | <code># Display disk usage for all files and directories in human readable format</code> |
| <code>du -sh</code>   | <code># Display total disk usage off the current directory</code>                        |
  - **DIRECTORY NAVIGATION**

|                      |                                                                                              |
|----------------------|----------------------------------------------------------------------------------------------|
| <code>cd ..</code>   | <code># To go up one level of the directory tree. (Change into the parent directory.)</code> |
| <code>cd</code>      | <code># Go to the \$HOME directory</code>                                                    |
| <code>cd /etc</code> | <code># Change to the /etc directory</code>                                                  |