**Linux Practicals**

**Practical 1 – Introduction to Linux**

#### What Is Linux?

Linux is an **operating system (OS)**. You are most likely used to a Windows OS of some description so Linux is partly chosen for your labs for that specific reason – to **expose you to a different OS**. Also Linux is becoming more popular now on both the server and desktop areas.

Linux is closely based on the very popular **UNIX OS** that began development in the **late 1960s**, in an effort to provide a **multi-user**, **multiprogramming** system for use by programmers.

#### What Is Multiuser?

Linux is considered to be **multiuser** in two senses – the first being that **supplying a username/password is mandatory** for a Linux system whereas it is either impossible or just optional in many other systems such as Windows 98/2000/XP etc. It is also multiuser in another sense, in that **more than one user can carry out a session on a Linux system simultaneously**. In other words I can carry on a session on a Linux system and in the middle of it I might decide to log on as another user, perhaps for administration purposes to carry out an admin-specific task.

**What Is Multiprogramming?**

Multiprogramming-based OS are ones that **can carry out a number of processes/tasks/jobs at the same time** (more in lectures) – which pretty much describes all modern OS – Windows 3.1, DOS and the older MAC OS are all examples of older uni-programmed systems as they could only carry out one process at a time. Another process could only be started when the previous process had ended.

#### Linux – Free, Open-source, Free, Highly Portable, Free, POSIX Compliant (Did I mention Free?)

The philosophy behind the design of UNIX was to provide simple, yet powerful, utilities that could be pieced together in a flexible manner to perform a wide variety of tasks on a wide variety of target platforms and the development of UNIX continues to the present day with a very high level of support offered for it by both hardware and software vendors. Linux has come to prominence in the more recent past because it is a **freely available OS** (unlike UNIX and Windows and MAC and ……) and allows you to do everything that you could do on a UNIX system from an **ordinary PC** workstation. There are many servers online from which you can download the Linux OS

There are many different varieties of Linux such as **Red Hat**, **Mandrake**, **SuSE** and **Debian** but the one we deal with in this lab course will be **Fedora,** which is one of the more popular varieties.

## Logging In to Linux

In the college environment, **your Linux account is intertwined with your ITT network account** in so far as your Linux login will be identical to your Windows login and what you create in terms of files and folders under Linux will appear when you log back in under Windows, and of course the files you create under Windows will appear under your Linux account. Also, you will be able to see the **Xythos Drive** contents under Linux, including this lab sheet hopefully!

Logging in to a Linux system requires two pieces of information: A **username**, and a **password**. These will be the same username and password that you use elsewhere in college.

When you sit down for a Linux session, you are given a **console login prompt** that looks something like this – it indicates the **Linux distribution** being used (Fedora), its **release version** (16) and **project name** (Verne). The **Kernel version** is 3.2.6 and this kernel should operate on x86 Intel CPU architectures. The name given to this Linux station is simply “labpc”.

**Fedora release 16 (Verne)**

**Kernel 3.2.6** **on an i686**

**labpc login:**

Note that **Linux is** **case sensitive** so make sure you type your username and password as required.

When you have typed your unique t-number username at the login prompt, press the **return** key. The system will then ask you for your password. When you type your password, the screen **will not display** what you type. Again hit **return** once you have entered your password.

When you have successfully logged onto Linux there may be a delay if it is your first time logging on to your Linux account – this is because your account is being initialised and various Linux login files and, for example, Mozilla Firefox profile are being created automatically for you. After this, you may be informed about the last time you logged in to your Linux account (not if it is your first time) and on which **terminal** (tty) and then a **shell prompt** appears, usually a **dollar sign**.

**[t00012345 @ labpc ~]$**

This is the system’s way of telling you that it is now ready to accept typed commands from you for interpretation. You are now at the **console** mentioned earlier and operating in **text-mode**. You could do a lot of useful work from this interface but it is not very exciting for you (or myself) who generally prefer a GUI-based environment. Also in this environment you will not be able to appreciate how easy a modern Linux system is to use. Therefore, to launch the default GUI environment for yourself type

**startx**

at the dollar prompt. Now the Linux graphical windowing environment will start up. You should see something similar to the following screenshot. There should be a nice bright desktop GUI presented with a few icons on it, mostly acting as **shortcuts** to file systems – your own (Home) account and Xythos. As you might have guessed, “Trash” is the equivalent of Window’s “Recycle Bin” and “Computer” the equivalent of “My Computer”. The desktop environment you are using is called **GNOME** and, similar to a Windows desktop, has a panel offering various menus:

**Applications** - where you will find Office programs such as Word Processors, games, a Web broswer and even programming tools.

**Places** – which allows you to immediately go to a particular location or device in your system such as your memory stick or CD. There is also a useful “Search” facility that allows you to find files.

You should spend a few minutes now just looking through and familiarising yourself with the look and feel of the Fedora desktop and its menu options. One of the main goals of today’s lab session is to illustrate how a Linux-based system is a viable alternative to a Windows-based one.

There is also a **taskbar panel** at the bottom of the desktop containing tabs for applications that are currently running, similar to a Windows environment.

###### Getting used to Linux

This section should help you to quickly realise that Linux is more than suitable as a desktop environment.

**Exercise 1: Retrieve and launch this practical sheet.**

Here you can just double-click the “**Lab Share**” icon and navigate your way to the CathrynCasey/Operating Systems/Labs/Linux folder. Inside you will find this lab sheet. Just double-click on the icon for the lab sheet and the **Libre Office** **Writer** application should start and open up the document. This application behaves in much the same fashion as MS Word, visually it looks virtually identical to it and is compatible with it to a huge extent. However, like so many Linux packages, it is **completely** **free.** Have a quick look through the various Writer menu options and you will see the usual choices and maybe even one or two not available from MS Word.

**Exercise 2: Use Libre Office Writer to Create a Document**

Now use the Writer application to create a small test document – mix in various formatting options such as italics, bold, font size etc. and attempt to save the file in M**icrosoft Word 97/2000/XP** format. You may get a message indicating the possibility that some of your formatting may be lost as a result of this choice since there are some slight differences between the two applications but proceed anyhow. When you next log in to Windows (you need not do it now), you will see that when you open the document, it appears pretty much identical in MS Word.

**Exercise 3: Creating a Shortcut on the Desktop for Libre Office Writer**

Next you should create a desktop shortcut to the Writer application. Go to **Applications-**>**Office**->**Word** **Processor** and just drag the “Word Processor” option to the desktop. If Word Processor is already on your desktop, drag Presentation instead. This is an application similar to PowerPoint. Test out that the shortcut works by double-clicking on it – hopefully the Writer application will launch again. If you now right-click on the Word Processor icon you will see a “**Properties**” option given at the end of the list. Select this and you will see under the “**Basic**” tab where in the system the icon for “Word Processor” is actually stored. There is a folder called “**Desktop**” automatically created to store all of your personally added desktop icons and it resides in your **home directory**. In Linux systems, each account has a so-called home directory where data files and system files for that particular user are stored. In the ITT setup, your home directory probably resembles /home2/t00012345. So the icon that you have just dragged to the desktop is now stored by the system and it will “remember” it the next time you log in to Linux, provided that you have not altered the Desktop folder in the meantime. So the “Desktop” folder will contain icons for any desktop shortcuts you create and, of course, you can also launch the applications from this location also. If you have inserted a memory stick into a USB port, it should be automatically detected by the system and an icon added for it to the desktop also, similar to the situation under Windows XP.

**Exercise 4: Attempting to Remove a Desktop Shortcut**

Some other icons appearing on your desktop upon boot-up such as “Trash” and “Computer” are added automatically to the desktop by so-called boot time “startup scripts” which are out of your control as a “regular” user – you need to be logged in as the special “root” user to alter these files directly. Prove this to yourself now by trying to remove the “Computer” icon – right-click on the icon and you will see that the “**Move to Trash**” option is not available. Right-click on the “Word Processor” icon now, you should see that this icon can be removed if desired – this is because you have personally created this shortcut, it is located in your home directory and so it is under your direct control. There is a very useful utility called the **Configuration Editor**, which can be used, in regular user mode, to remove the likes of the “Computer” icon if this were required. This utility, if installed, is normally located in **Applications->System Tools**.

**Exercise 5: Using the File Browser**

Use the File Browser now, to navigate your way to your **home directory**. You should see all the files and folders here that you would see on your Windows system since your home directory in Linux equates to your X: drive under Windows. We will look at the Linux file system again and again throughout this practical course as there are some differences between its operation and the way things work under Windows. If you haven’t already done so, insert a memory stick into the computer and notice that it is detected automatically as per Windows. Now use the file browser utility to navigate your way around its file system, to ensure everything is as expected.

**Exercise 6: Launch the Web Browser**

Launch **Firefox**, another piece of free, open source software, that you will be familiar with from the Windows world (Click on Applications-Internet-Firefox). Go to [http://fedoraproject.org](http://fedoraproject.org/) where you will get all the latest news on the Fedora Linux project, there is also heaps of documentation on the OS, download information and even blogs. Answer the following questions based on this website:

1. What is the latest version of Fedora? **20**
2. How often does a new release of Fedora appear? **Every six months.**

(iii) List any 2 people /organisations who use Fedora\_ **US Postal Service, US Nuclear Security Administration**

(iv) Give any 2 reasons why contributing to the Fedora project can be beneficial to the contributor **Makes you a smarty pant, it makes you stand out in teh crowd**

**Exercise 7: Using Desktop Workspaces**

Your desktop may well be cluttered with all kinds of applications at this stage. You've been using the Nautilus File Manager in File Browser mode, you launched the OpenOffice Writer application twice and you've also launched Mozilla Firefox. However, all of these windows are attached to one “workspace”. You may notice in the bottom panel of the desktop there are a few “squares” in the right-hand side. One of these is highlighted in blue. This is your current workspace. Now click on the square next to the blue one. You should see that all the windows magically disappear. This is because you are now in a different workspace and none of the previously opened windows are attached to it. This is now in blue indicating it is the current workspace. Workspaces allow you to use your system in a more organised fashion, ensuring that your desktop doesn't become too cluttered with windows. Each workspace is like a virtual desktop. By default there are 2 workspaces available. In the new workspace, you should now find and open the “**calculator**” application in “**Scientific**” mode and calculate the value of **6(Sin30+Cos60)**. Now, switch back to your original workspace – you should find all your open applications are still there and running. Right-click on the workspaces and choose Preferences to see how you can alter the workspaces setup.

**Exercise 8: Changing System Settings**

Right-click on your name up in the top right corner of the screen and choose system settings. Look at the different configuration settings that you can change. Can also click on Applications-System Tools-System Settings.

**Exercise 9: Changing the System Date and Time**

If your Linux system is attached to the Internet, then setting the system date and time may be unnecessary as it can set automatically at boot-time via synchronising with a **NTP server**. However, if the system is not online it may be necessary to set the date and time manually. In this case, go to the time button in the right of the top panel and right-click on it. Now attempt to “**Adjust Date and Time**”. You must be the “**root**” user, mentioned earlier, to perform this task. Therefore, this window is effectively prompting you for the root password. In the ITT Linux environment this password is “**itt12345**”. So enter this now, and have a quick look at the time/date settings. The “**Network Time Protocol**” tab enables you to set up your machine so that it sets the system time using a time server. You have the option of adding to the list of time servers given.

**Package Management**

All software on a Fedora system is divided into RPM packages, which can be installed, upgraded, or removed. You can manage packages on Fedora using both **Yum** and the **PackageKit** suite of graphical package management tools.

**PackageKit**

Fedora provides **PackageKit** for viewing, managing, updating, installing and uninstalling packages compatible with your system. **PackageKit** consists of several graphical interfaces that can be opened from the GNOME panel menu, or from the Notification Area when **PackageKit** alerts you that updates are available.

**Updating Packages with Software Update**

You can open **Software Updates** by clicking **Applications****System Tools****Software Update** from the **Activities** menu, or running the **gpk-update-viewer** command at the shell prompt. In the **Software Updates** window, all available updates are listed along with the names of the packages being updated (minus the **.rpm** suffix, but including the CPU architecture), a short summary of the package, and, usually, short descriptions of the changes the update provides. Any updates you do not wish to install can be de-selected here by unchecking the checkbox corresponding to the update.

Figure Installing updates with Software Update

The updates presented in the **Software Updates** window only represent the currently-installed packages on your system for which updates are available; dependencies of those packages, whether they are existing packages on your system or new ones, are not shown until you click **Install Updates**.

**PackageKit** utilizes the fine-grained user authentication capabilities provided by the **PolicyKit** toolkit whenever you request it to make changes to the system. Whenever you instruct **PackageKit** to update, install or remove packages, you will be prompted to enter the superuser password before changes are made to the system. If you instruct **PackageKit** to update the **kernel** package, then it will prompt you after installation, asking you whether you want to reboot the system and thereby boot into the newly-installed kernel.

**Setting the Update-Checking Interval**

Selecting **Applications****Other****Software Updates** from the **Activities** menu opens the **Software Update Preferences** window. The **Update Settings** tab allows you to define the interval at which **PackageKit** checks for package updates, as well as whether or not to automatically install all updates or only security updates. Leaving the **Check for updates when using mobile broadband** box unchecked is handy for avoiding extraneous bandwidth usage when using a wireless connection on which you are charged for the amount of data you download.

Setting PackageKit's update-checking interval

**Setting the Software Sources**

To select which package repositories to use to install software updates, select **Applications****Other** **Software Updates** from the **Activities** menu, and click the **Software Sources** tab of the **Software Update Preferences** window.

Figure Setting PackageKit's software sources

**PackageKit** refers to **Yum** repositories as software sources. It obtains all packages from enabled software sources. The **Software Sources** tab shows the repository name, as written on the **name=*My Repository Name*** field of all [*repository*] sections in the **/etc/yum.conf** configuration file, and in all ***repository*.repo** files in the **/etc/yum.repos.d/** directory.

Entries which are checked in the **Enabled** column indicate that the corresponding repository will be used to locate packages to satisfy all update and installation requests (including dependency resolution). The **Enabled** column corresponds to the **enabled=*<1 or 0>*** field in [*repository*] sections. Checking an unchecked box enables the Yum repository, and unchecking it disables it.

Performing either function causes **PolicyKit** to prompt for superuser authentication to enable or disable the repository. **PackageKit** actually inserts the **enabled=*<1 or 0>*** line into the correct [*repository*] section if it does not exist, or changes the value if it does. This means that enabling or disabling a repository through the **Software Sources** window causes that change to persist after closing the window or rebooting the system. The ability to quickly enable and disable repositories based on our needs is a highly-convenient feature of **PackageKit**.

Note that it is not possible to add or remove **Yum** repositories through **PackageKit**.

**Using Add/Remove Software**

**PackageKit**'s **Software Update** GUI window is a separate application from its **Add/Remove Software** application, although the two have intuitively similar interfaces. To find and install a new package, select **Applications****System Tools****Add/Remove Software** from the **Activities** menu, or run the **gpk-application** command at the shell prompt.

Figure PackageKit's Add/Remove Software window

**Refreshing Software Sources (Yum Repositories)**

To enable or disable a **Yum** repository, open a dialog box by sclicking **System****Software Sources**, and select the **Software Sources** tab. After enabling and/or disabling the correct **Yum** repositories, make sure that you have the latest list of available packages. Click on **System****Refresh Package Lists** and **PackageKit** will obtain the latest lists of packages from all enabled software sources, that is, **Yum** repositories.

**Finding Packages with Filters**

You can view the list of all *configured* and unfiltered (see below) **Yum** repositories by opening **Add/Remove Software** and clicking **System****Software Sources**. Once the software sources have been updated, it is often beneficial to apply some filters so that **PackageKit** retrieves the results of our **Find** queries faster. This is especially helpful when performing many package searches. Four of the filters in the **Filters** drop-down menu are used to split results by matching or not matching a single criterion. By default when **PackageKit** starts, these filters are all unapplied (**No Filter**), but once you do filter by one of them, that filter remains set until you either change it or close **PackageKit**. Because you are usually searching for available packages that are *not* installed on the system, click **Filters****Installed** and select the **Only Available** radio button.

Figure Filtering out already-installed packages

Also, unless we require development files such as C header files, we can filter for **Only End User Files** and, in doing so, filter out all of the ***package\_name*-devel** packages we are not interested in.

Figure Filtering out development packages from the list of Find results

The two remaining filters with submenus are:

**Graphical**

Narrows the search to either applications which provide a GUI interface (**Only Graphical**) or those that do not. This filter is useful when browsing for GUI applications that perform a specific function.

**Free**

Search for packages which are considered to be free software Refer to the *Fedora Licensing List* for details on approved licenses.

The remaining checkbox filters are always either checked or unchecked. They are:

**Hide Subpackages**

Checking the **Hide Subpackages** checkbox filters out generally-uninteresting packages that are typically only dependencies of other packages that we want. For example, checking **Hide Subpackages** and searching for ***package*** would cause the following related packages to be filtered out of the **Find** results (if it exists):

• ***package*-devel**

• ***package*-libs**

• ***package*-libs-devel**

• ***package*-debuginfo**

**Only Newest Packages**

Checking **Only Newest Packages** filters out all older versions of the same package from the list of results, which is generally what we want.

**Using the Only Newest Packages filter**

**Only native packages**

Checking the **Only Native Packages** box on a multilib system causes **PackageKit** to omit listing results for packages compiled for the architecture that runs in *compatibility mode*. For example, enabling this filter on a 64-bit system with an AMD64 CPU would cause all packages built for the 32-bit x86 CPU architecture not to be shown in the list of results, even though those packages are able to run on an AMD64 machine. Packages which are architecture-agnostic (i.e. *noarch* packages such as **crontabs-1.10-32.1.el6.noarch.rpm**) are never filtered out by checking **Only Native Packages**. This filter has no affect on non-multilib systems, such as x86 machines.

**Installing and Removing Packages (and Dependencies)**

With the two filters selected, **Only Available** and **Only End User Files**, search for the **htop** interactive process viewer and highlight the package. You now have access to some very useful information about it, including: a clickable link to the project homepage; the **Yum** package group it is found in, if any; the license of the package; a pointer to the GNOME menu location from where the application can be opened, if applicable; and the size of the package, which is relevant when we download and install it.

Figure Viewing and installing a package with PackageKit's Add/Remove Software window

When the checkbox next to a package or group is checked, then that item is already installed on the system. Checking an unchecked box causes it to be *marked* for installation, which only occurs when the **Apply** button is clicked. In this way, you can search for and select multiple packages or package groups before performing the actual installation transactions. Additionally, you can remove installed packages by unchecking the checked box, and the removal will occur along with any pending installations when **Apply** is pressed. Dependency resolution , which may add additional packages to be installed or removed, is performed after pressing **Apply**. **PackageKit** will then display a window listing those additional packages to install or remove, and ask for confirmation to proceed. Check **htop** and click the **Apply** button. You will then be prompted for the superuser password; enter it, and **PackageKit** will install **htop**. One nice feature of **PackageKit** is that, following installation, it sometimes presents you with a list of your newly-installed applications and offer you the choice of running them immediately. Alternatively, you will remember that finding a package and selecting it in the **Add/Remove Software** window shows you the **Location** of where in the GNOME menus its application shortcut is located, which is helpful when you want to run it.

Once it is installed, you can run **htop**, a colorful and enhanced version of the **top** process viewer, by opening a shell prompt and entering:

**htop**

**htop** is nifty, but we decide that **top** is good enough for us and we want to uninstall it. Remembering that we need to change the **Only Available** filter we recently used to install it to **Only Installed** in **Filters****Installed**, we search for **htop** again and uncheck it. The program did not install any dependencies of its own; if it had, those would be automatically removed as well, as long as they were not also dependencies of any other packages still installed on our system.

Although **PackageKit** automatically resolves dependencies during package installation and removal, it is unable to remove a package without also removing packages which depend on it. This type of operation can only be performed by **RPM**, is not advised, and can potentially leave your system in a non-functioning state or cause applications to misbehave and/or crash.

Figure Removing a package with PackageKit's Add/Remove Software window

**Yum**

**Yum** is the The Fedora Project package manager that is able to query for information about packages, fetch packages from repositories, install and uninstall packages using automatic dependency resolution, and update an entire system to the latest available packages. Yum performs automatic dependency resolution on packages you are updating, installing or removing, and thus is able to automatically determine, fetch and install all available dependent packages. Yum can be configured with new, additional repositories, or *package sources*, and also provides many plug-ins which enhance and extend its capabilities. Yum is able to perform many of the same tasks that **RPM** can; additionally, many of the command line options are similar. Yum enables easy and simple package management on a single machine or on groups of them.

Yum also enables you to easily set up your own repositories of **RPM** packages for download and installation on other machines. Learning Yum is a worthwhile investment because it is often the fastest way to perform system administration tasks, and it provides capabilities beyond those provided by the **PackageKit** graphical package management tools.

You must have superuser privileges in order to use **yum** to install, update or remove packages on your system. All examples in this chapter assume that you have already obtained superuser privileges by using either the **su** or **sudo** command.

**Checking For Updates**

To see which installed packages on your system have updates available, use the following command:

**yum check-update**

For example:

The packages in the above output are listed as having updates available. The first package in the list is **PackageKit**, the graphical package manager. The line in the example output tells us:

• **PackageKit** — the name of the package

• **x86\_64** — the CPU architecture the package was built for

• **0.6.14** — the version of the updated package to be installed

• **fedora** — the repository in which the updated package is located

The output also shows us that we can update the kernel (the kernel package), Yum and **RPM** themselves (the **yum** and **rpm** packages), as well as their dependencies (such as the *kernel-firmware*, *rpm-libs*, and *rpm-python* packages), all using **yum**.

**Updating Packages**

You can choose to update a single package, multiple packages, or all packages at once. If any dependencies of the package (or packages) you update have updates available themselves, then they are updated too.

**Updating a Single Package**

To update a single package, run the following command as root:

**yum update** *package\_name*

For example, to update the *udev* package, type:

~]# **yum update udev**

Loaded plugins: langpacks, presto, refresh-packagekit

Updating Red Hat repositories.

INFO:rhsm-app.repolib:repos updated: 0

Setting up Update Process

Resolving Dependencies

--> Running transaction check

---> Package gdb.x86\_64 0:7.2.90.20110411-34.fc15 will be updated

---> Package gdb.x86\_64 0:7.2.90.20110429-36.fc15 will be an update

--> Finished Dependency Resolution

Dependencies Resolved

================================================================================

Package Arch Version Repository Size

================================================================================

Updating:

gdb x86\_64 7.2.90.20110429-36.fc15 fedora 1.9 M

Transaction Summary

================================================================================

Upgrade 1 Package(s)

Total download size: 1.9 M

Is this ok [y/N]:

**Updating All Packages and Their Dependencies**

To update all packages and their dependencies, simply enter **yum update** (without any arguments):

**Searching Packages**

You can search all RPM package names, descriptions and summaries by using the following command:

**yum search** *term*…

**Listing Packages**

**yum list** and related commands provide information about packages, package groups, and repositories. All of Yum's list commands allow you to filter the results by appending one or more *glob expressions* as arguments. Glob expressions are normal strings of characters which contain one or more of the wildcard characters **\*** (which expands to match any character multiple times) and **?** (which expands to match any one character).

**yum list all**

Lists all installed *and* available packages.

**yum list installed**

Lists all packages installed on your system. The rightmost column in the output lists the repository from which the package was retrieved.

**yum list available**

Lists all available packages in all enabled repositories.

**yum repolist**

Lists the repository ID, name, and number of packages it provides for each *enabled* repository.

**Installing Individual Packages**

To install a single package and all of its non-installed dependencies, enter a command in the following form:

**yum install** *package\_name*

You can also install multiple packages simultaneously by appending their names as arguments:

**yum install** *package\_name package\_name*…

**Removing Packages**

Similarly to package installation, Yum allows you to uninstall (remove in **RPM** and Yum terminology) both individual packages and a package group.

**Removing Individual Packages**

To uninstall a particular package, as well as any packages that depend on it, run the following command as root:

**yum remove** *package\_name*…

As when you install multiple packages, you can remove several at once by adding more package names to the command. For example, to remove *totem*, *rhythmbox*, and *sound-juicer*, type the following at a shell prompt:

~]# **yum remove totem rhythmbox sound-juicer**

Similar to **install**, **remove** can take these arguments:

• package names

• glob expressions

• file lists

• package provider

**Exercise 10: Messing Around**

If you have any remaining time, you should spend it customising the appearance of your desktop or just getting to know the Fedora GUI a little better by selecting various menu options etc.

**Exercise 11: Unmounting Devices, Logging Off and Shutting Down**

Before you log off it is good practice to close all open windows and to **unmount** any devices such as memory sticks that you have attached to the system. To unmount a device simply right-click its icon on the desktop and select “**Unmount Volume**”. Doing this helps to prevent possible data loss and makes the shutting down process cleaner.

Now, to log off simply go to **System->Log Out t00012345**. Doing this will log you off, save all your session settings, and take you back to the console but the system will still remain operational.

In order to shut the system down and reboot, you need to hit **ALT+CTRL+DEL** at the console. This will begin the shutdown process and reboot the machine.