Linux Documentation

1. Introduction to the Linux Environment

Welcome to the Linux workstation environment in the College of Engineering at Bucknell University. There are three Linux laboratories in the Dana and Breakiron Engineering Buildings: Dana 213, Breakiron 164 and Breakiron 167. Dana 213 and Breakiron 164 are college laboratories and are open for public use when no classes have been reserved. Breakiron 167 is a limited access lab for use by Computer Science faculty and junior and senior CS students. The Dana 213 lab has 25 workstations; Breakiron 164 has 36 workstations, and Breakiron 167 has 10 workstations. There are also 3 Linux servers available for you to remotely connect to: linuxremote1.eg.bucknell.edu, linuxremote2.eg.bucknell.edu, linuxremote3.eg.bucknell.edu.

This manual serves as an introduction to the facilities available on the Linux systems, including the graphical environment, called GNOME and the Linux command line. In the following sections you will learn how to access the Linux workstations and how to use them in creating, compiling, and running programs. You will also learn about the basic Linux commands and the facilities for printing files. This manual was written with the first time user in mind. It is written in a straight forward manner so you can understand the material and can quickly become comfortable

using the Linux workstations.

1.1. Operation Hours

The Linux labs are normally open 24 hours a day, 7 days a week. It is advised to check the printed weekly schedule before walking into the lab.

1.2. Seeking Assistance

Mike Harvey and Jeremy Dreese of the Engineering Computing Support Team (ECST) provide support for the Linux systems. They can be reached in one of the following ways:

1. Email ecst@bucknell.edu
2. Call x71460 or x73714
3. Visit Dana 215 or Dana 219

2. The GNOME Graphical Environment

GNOME is the graphical environment that you will use to interact with the Linux systems. This chapter will provide an overview of how to interact with the GNOME environment, including things such as logging in and out, customizing your desktop environment, and working with the file manager to manage your files and directories.

2.1. Logging In, Logging Out and Locking the Screen

In this section, we'll discuss how to begin using the system, how to lock your session so that no one else can use it while you are away from your computer, and how to end your session when you're finished.

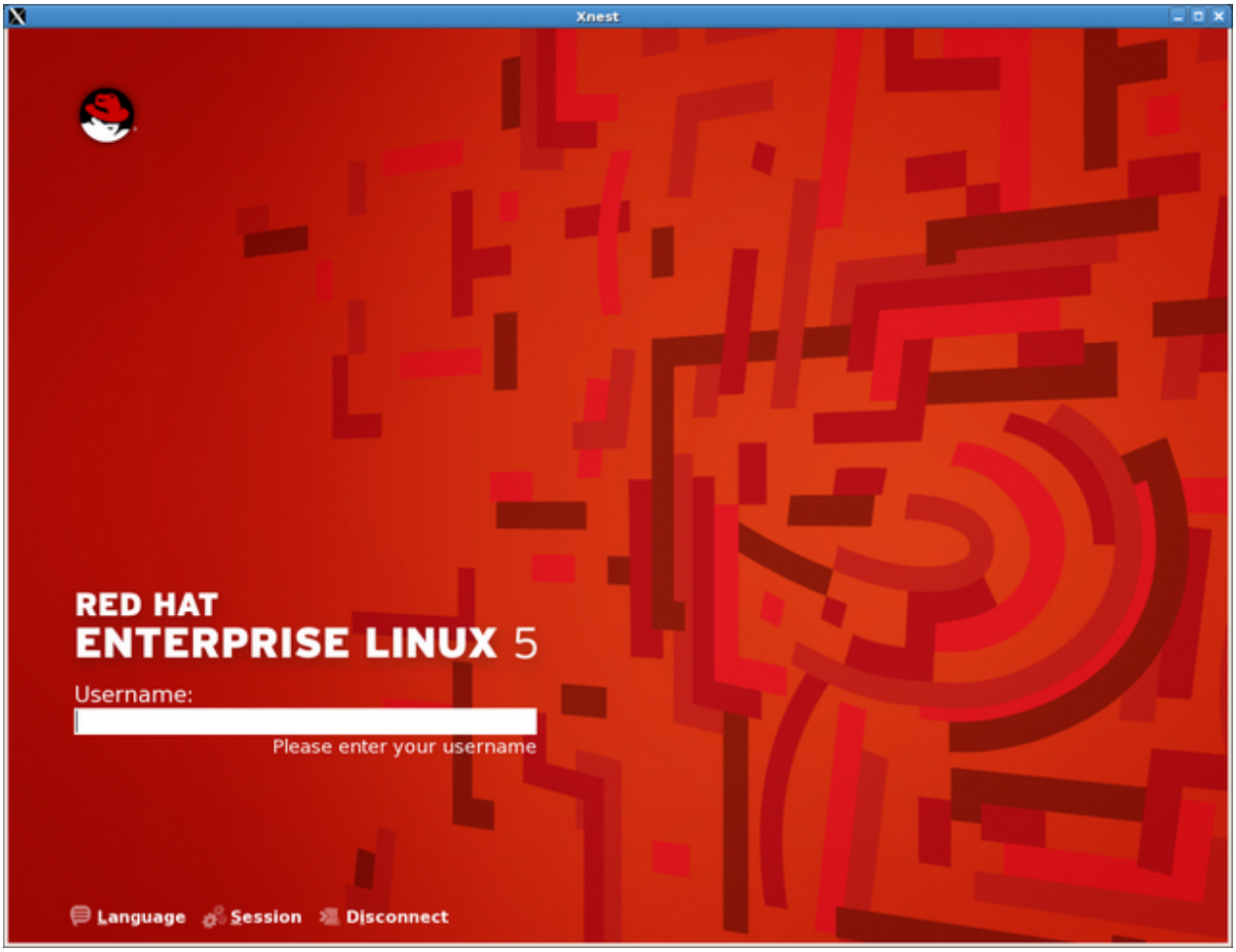
2.1.1. Logging In

Library and IT sent you a username and password via postal mail for logging into the Linux systems. Please remember that usernames and passwords are case sensitive so you must type them exactly as they appeared in your login letter.

To login:

1. Enter your username
2. Press the **Enter** key
3. Enter your password
4. Press the **Enter** key

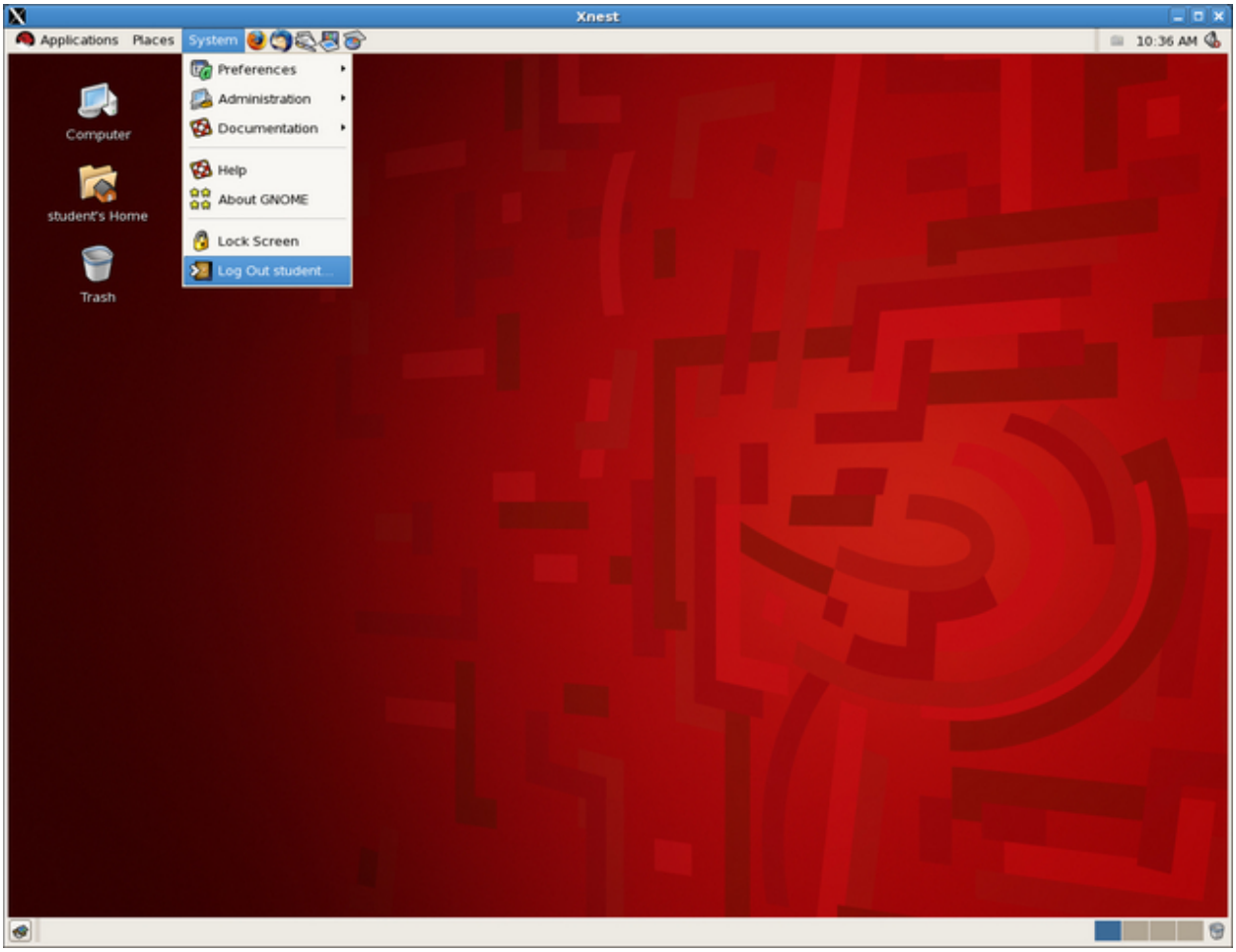
Figure 1. Graphical Login Screen



2.1.2. Logging Out

To log out of GNOME, go to the **System Menu** menu and select **Log Out username...**.

Figure 2. Logging Out

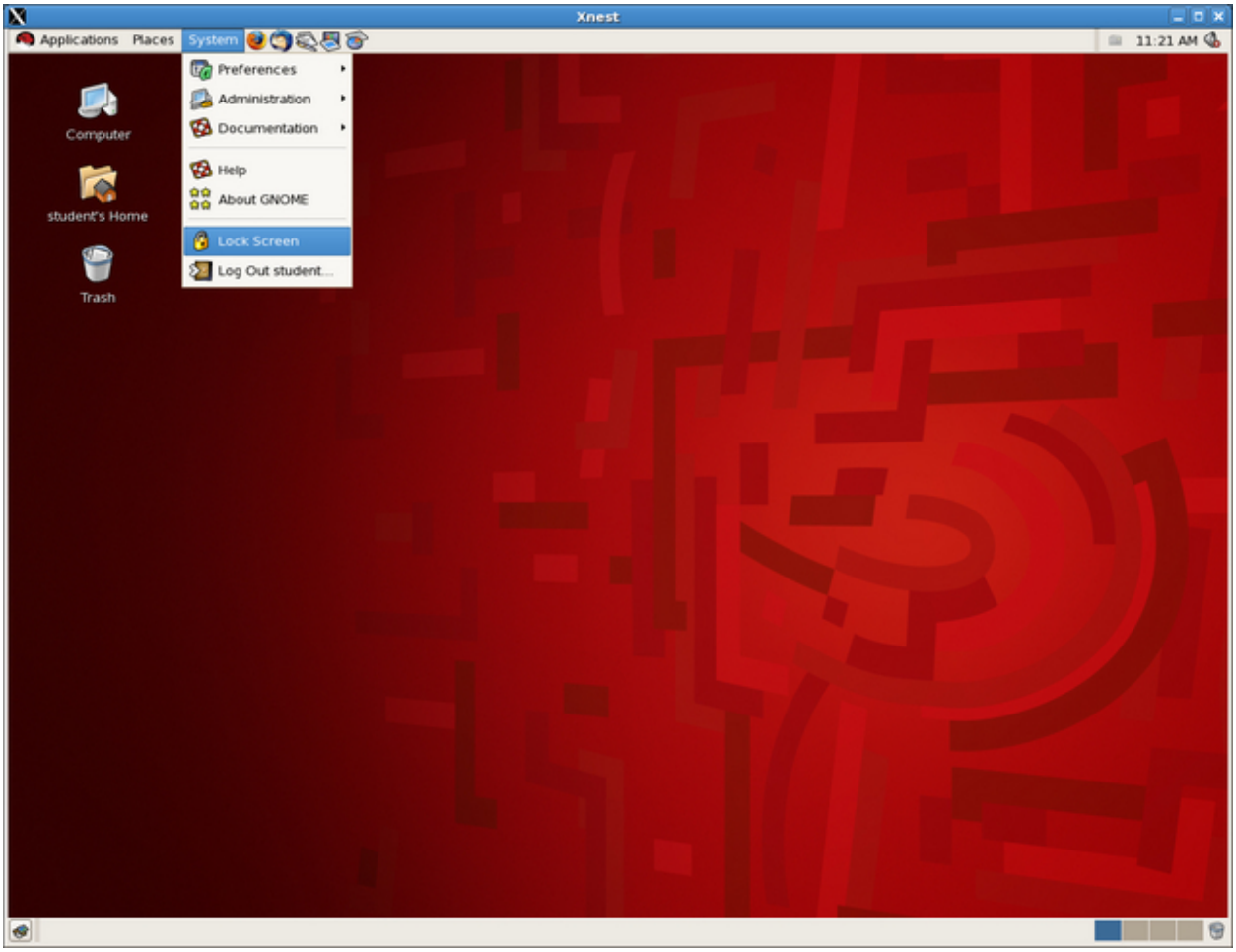


2.1.3. Locking the Screen

You may find that you wish to walk away from your computer for a few minutes but don't want to go to the trouble of logging out. Locking your screen is an effective way to securely put your session on hold while you're away from your computer. Please note that it is extremely important to lock your screen when you walk away from your computer. This prevents people from accessing your files by requiring you to enter your password when you move the mouse or touch the keyboard the next time you use the computer. **However, please be warned that if you leave your screen locked for more than 15 minutes, you may be logged out so the machine can be used by someone else. If you plan on being away longer than 15 minutes, please log out.**

To lock your screen, go the **System Menu** menu and select **Lock Screen**.

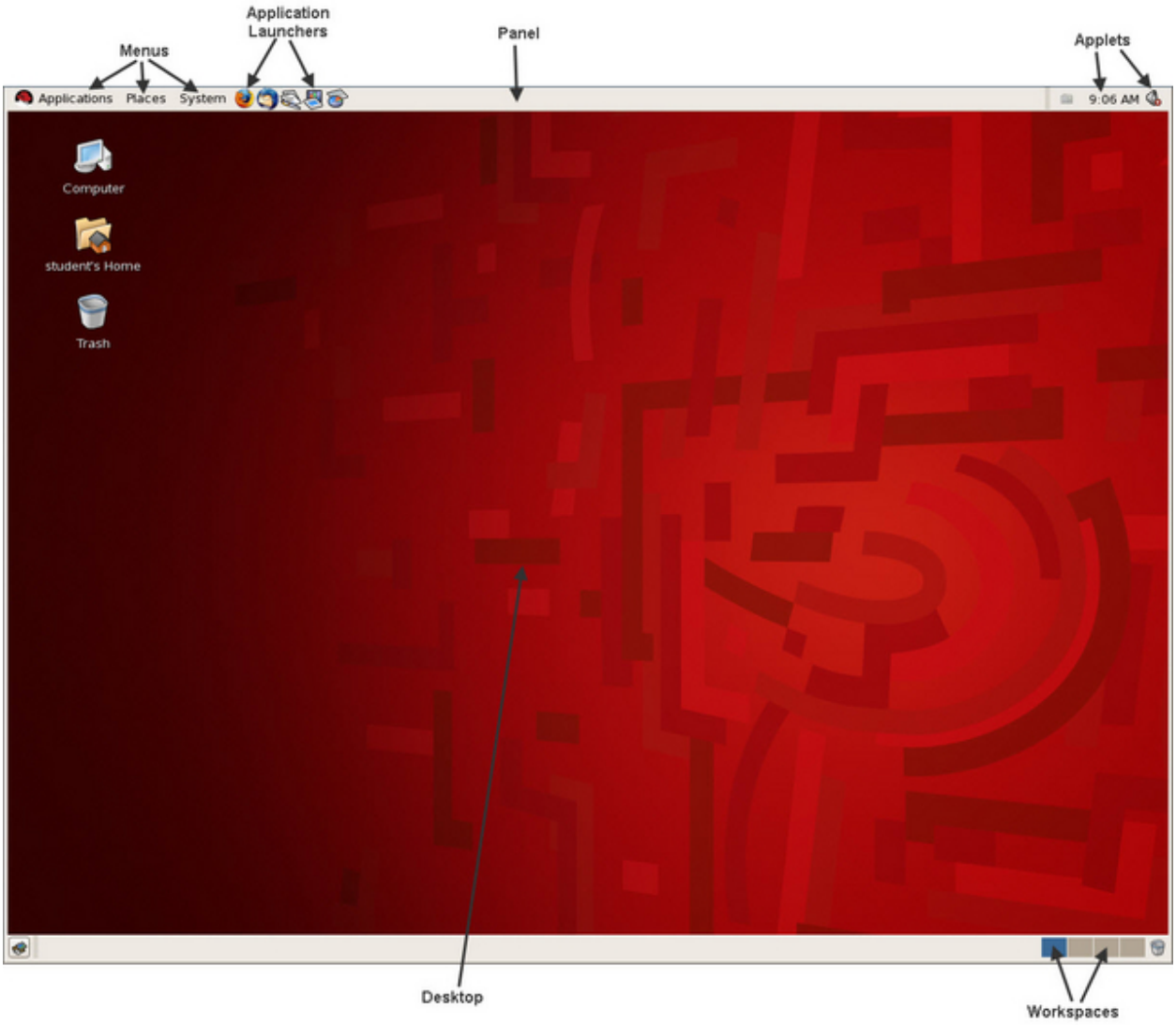
Figure 3. Locking the Screen



2.2. Introduction to the Desktop

When you start a desktop session for the first time, you should see a default session, with two panels and various icons on both the panels and desktop.

Figure 4. The Default Desktop



We will provide a brief introduction to the major components of the GNOME Desktop in this section and then elaborate on some of the various components in future sections.

The major components of the GNOME Desktop are as follows:

Panels

- Panels are areas in the GNOME Desktop from which you can access all of your system applications and menus. Panels are very configurable. A particularly important panel is the top edge panel. The top edge panel includes the Menu Bar. The Menu Bar contains three special menus, as follows:
  - Applications menu: Contains all applications and configuration tools. This menu also includes the file browser and the help browser.
  - Places menu: Contains various locations on the system to which you might want to navigate.
  - System menu: Contains various commands for performing functions like setting user preferences, locking the screen and logging out of the system.

Menus

- You can access all GNOME Desktop functions through menus. You can use the Applications menu to access many of the standard applications, commands, and configuration options. You can access the Applications menu from the Main Menu and from the Menu Bar applet. You can add the Main Menu and the Menu Bar applet to your panels. The Menu Bar applet also contains a Places and a System menu. The Places menu contains various locations on the system to which you might want to navigate. The System menu provides command for performing functions like setting user preferences, locking the screen and logging out of the system.

Windows

- You can display many windows at the same time. You can run different applications in each window. The window manager provides frames and buttons for windows. The window manager enables you to perform standard actions such as move, close, and resize windows.

Workspaces

- You can subdivide the GNOME Desktop into separate workspaces. A workspace is a discrete area in which you can work. You can specify the number of workspaces in the GNOME Desktop. You can switch to a different workspace, but you can only display one workspace at a time.

Nautilus file manager

- The Nautilus file manager provides an integrated access point to your files and applications. You can manage the contents of folders in the file manager and open the files in the appropriate applications.

Desktop

- The desktop is behind all of the other components on the desktop. The desktop is an active component of the user interface. You can place objects on the desktop to access your files and directories quickly, or to start applications that you use often. You can also right-click on the desktop to open a menu.

Preferences

- The GNOME Desktop contains dedicated preference tools. Each tool controls a particular part of the behavior of the GNOME Desktop. To start a preference tool, choose **System -> Preferences**. Choose the item that you want to configure from the sub-menus.

The components of the GNOME Desktop are inter-operable. Usually you can perform the same action in several different ways. For example, you can start applications from panels, from menus, or from the desktop. This chapter provides a useful quick guide to how to work with the GNOME Desktop.

To read more information about the GNOME Desktop Environment, please refer to the official GNOME documentation.

2.3. Panels

When you start a session for the first time, the desktop should contain two panels, one along the top and one along the bottom. You can customize your panels as needed. We'll discuss the following actions with panels:

You can perform the following actions with panels:

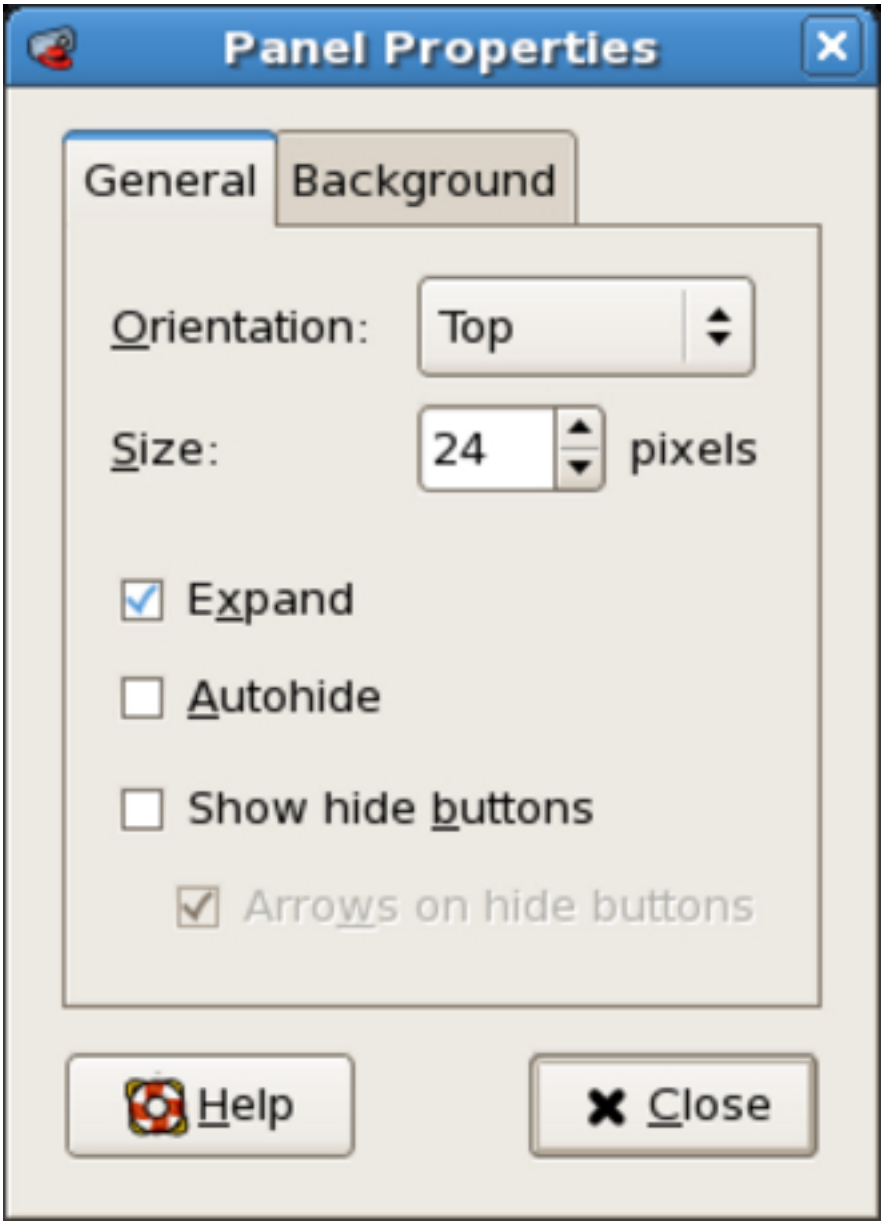
- Modifying a panel's properties
- Adding objects to panels
- Manipulating existing panel objects

Though less common, you can also add, delete or hide panels. Please refer to the official GNOME documentation for information on these tasks.

2.3.1. Modifying a Panel's Properties

To modify a panel's properties, right-click on an vacant space on the panel and select Properties. A dialog box similar to the following should appear:

Figure 5. Panel Properties

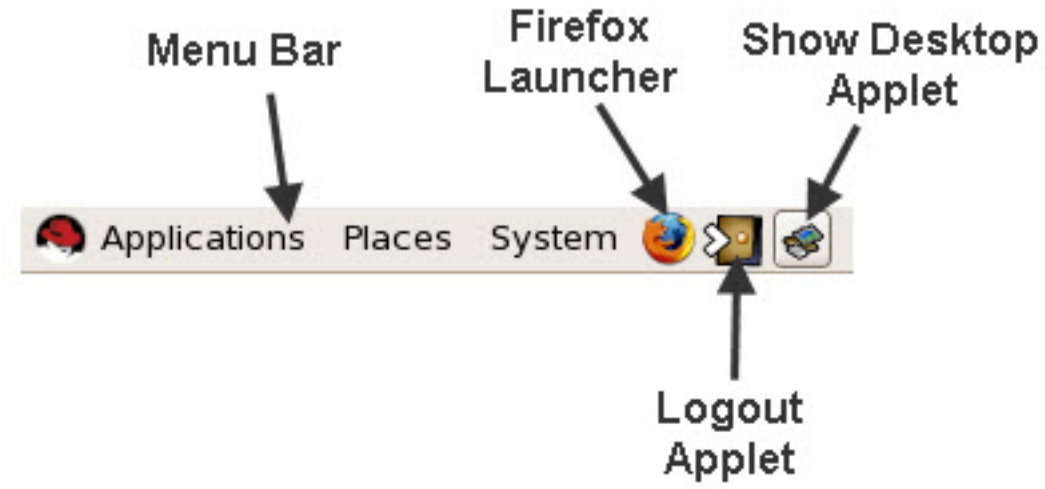


Within this dialog box, you can change the location, size, and/or color of the panel, among other various options.

2.3.2. Adding Object to Panels

A panel can hold several types of objects. The panel in the figure "A Panel With Various Objects" contains some of the types of objects you can add.

Figure 6. A Panel With Various Panel Objects



The two most common types of objects you'll want to add are:

- Applets

  Applets are small, interactive applications that reside within a panel, for example *Show Desktop Button* in "A Panel With Various Objects". The following applets appear in your panels by default:

  - *Menu Bar*: Provides access to the Applications menu and the Actions menu. You can use the menus to access your applications, preference tools, and other tasks.
  - *Clock*: Displays the date and time.
  - *Window Selector*: Lists all your open windows. To give focus to a window, click on the window selector icon at the extreme right of the top edge panel, then choose the window.
  - *Window List*: Displays a button for each window that is open. You can click on a window list button to minimize and restore windows. By default, Window List appears in the edge panel at the bottom of the screen.
  - *Workspace Switcher*: Displays a visual representation of your workspaces. You can use Workspace Switcher to switch between workspaces. By default, Workspace Switcher appears in the edge panel at the bottom-right portion of the screen.
  - *Show Desktop*: When clicked, this button hides all windows to show you the desktop.

  To add an applet to a panel, right-click on a vacant space on the panel, then choose **Add to Panel...**. Scroll through the list of available applets, select the one you want, and click **Add**.

- Launchers

  A *launcher* starts a particular application, executes a command, or opens a file. The Firefox icon in "A Panel With Various Objects" is a launcher for the *Firefox* application. A launcher can reside in a panel or in a menu. Click on the launcher to perform the action that is associated with the launcher.

  You can create your own launchers for applications. For example, you can create a launcher for a word processor application that you use frequently, and place the launcher in a panel for convenient access. To add a new launcher to a panel, right-click on a vacant space on the panel, then choose **Add to Panel -> Application Launcher** to

select one of the application launchers that already exists in the *Application* menu or choose **Add to Panel -> Custom Application Launcher** to define a new application that you'd like to launch.

2.3.3. To Manipulate Panel Objects

You can manipulate panel objects in the following ways:

- Move objects within a panel, or to another panel.

  You can move any object to another location in the panel. You can also move an object from one panel to another panel. Use the middle mouse button to click and drag the panel object to the new location.

- Remove objects from a panel.

  Right-click on the object and select **Remove From Panel**.

2.4. Menus

You can access many GNOME Desktop functions through menus. You can access your menus from the following GNOME Desktop components:

Main Bar

- The Menu Bar contains Applications and Actions menus and is in the upper-left corner of your screen by default. You can use the Applications menu and the Actions menu to access almost all of the standard applications, commands, and configuration options. To add the Main Menu to a panel, right-click on a vacant space on the panel, then choose **Add to Panel**. In the Add to the panel dialog, select **Menu Bar** and click **Add**.

Menu Menu

- The Main Menu contains the Applications menu, and various other functions. You can use the Applications menu to access almost all of the standard applications, commands, and configuration options. To add the Main Menu to a panel, right-click on a vacant space on the panel, then choose **Add to Panel**. In the Add to the panel dialog, select **Main Menu** and click **Add**.

2.5. Windows in the Desktop

You can display many windows at the same time on your desktop. Each window has a frame. The window frame contains active control elements that you can use to work with the window.

2.5.1. Types of Windows

The desktop features the following types of window:

- Application windows

  When you run an application, a frame usually borders the window. The top edge of the application window contains a titlebar. The titlebar contains buttons that you can use to work with the window. The buttons in an application window frame enable you to perform actions such as open the **Window Menu**, or close the window. The **Window Menu** provides commands that you can perform on the window.

- Dialog windows

  Dialog windows are associated with interactive processes. A dialog window consists of the window frame, and a single interactive pane that provides information and controls for the user. This manual refers to the interactive part of a dialog window as a dialog. The frame of a dialog window contains buttons that enable you to open the **Window Menu** , or to close the dialog window.

2.5.2. To Manipulate Windows

You use the frame of an application window or dialog window to perform various actions with the window. Most of the control elements are located on the top edge of the window frame. Top Edge of Frame for a Typical Application Window shows the top edge of a frame for a typical application window.

Figure 7. Top Edge of Frame for a Typical Application Window

The active control elements of the window frame are as follows:

Control Element	Description
Window Menu button	Click on the **Window Menu** button to perform actions such as moving, resizing, or moving the window to a different workspace.
Titlebar	You can use the titlebar to move the window by clicking and dragging it or you can shade the window by double-clicking it.
Minimize button	Click on the **Minimize** button to minimize the window.
Maximize button	You can use the **Maximize** button to maximize and restore the window. To maximize a window click on the **Maximize** button. To restore the window click on the **Maximize** button again.
Close Window button	Click on the **Close Window** button to close the window.
Border	Right-click on the border to open the **Window Menu**.

To change the size of windows grab the border (i.e. outside edge) of the window, but not the titlebar. Drag the border until the window is the size that you require.

2.5.2.1. To Give Focus to a Window

A window that has focus can receive input from the mouse and the keyboard. Only one window can have focus at a time. The window that has focus has a different appearance than other windows.

You can use the following elements to give focus to a window:

Element	Action
Mouse	Click on the window, if the window is visible.
Shortcut keys	Use shortcut keys to switch between the windows that are open. To give focus to a window, release the keys. The default shortcut keys to switch between windows are **Alt + Tab**.
Window List	Click on the button that represents the window in **Window List**.

2.6. Workspaces

You can display many windows at the same time on your desktop. Your windows are displayed in subdivisions of your desktop that are called workspaces. A workspace is a discrete area on the desktop in which you can work.

Every workspace on the desktop contains the same desktop background, the same panels, and the same menus. However, you can run different applications, and open different windows in each workspace. You can display only one workspace at a time on your desktop but you can have windows open in other workspaces.

Workspaces enable you to organize the desktop when you run many applications at the same time. When your current workspace becomes crowded with windows, you can move your work to another workspace. You can also switch to another workspace then start more applications.

Workspaces are displayed in the *Workspace Switcher* applet. In Workspaces Displayed in Workspace Switcher, the *Workspace Switcher* contains four workspaces. The first and second workspaces contain open windows. The last two workspaces do not contain currently active windows.

Figure 8. Workspaces Displayed in Workspace Switcher

2.6.1. To Switch Between Workspaces

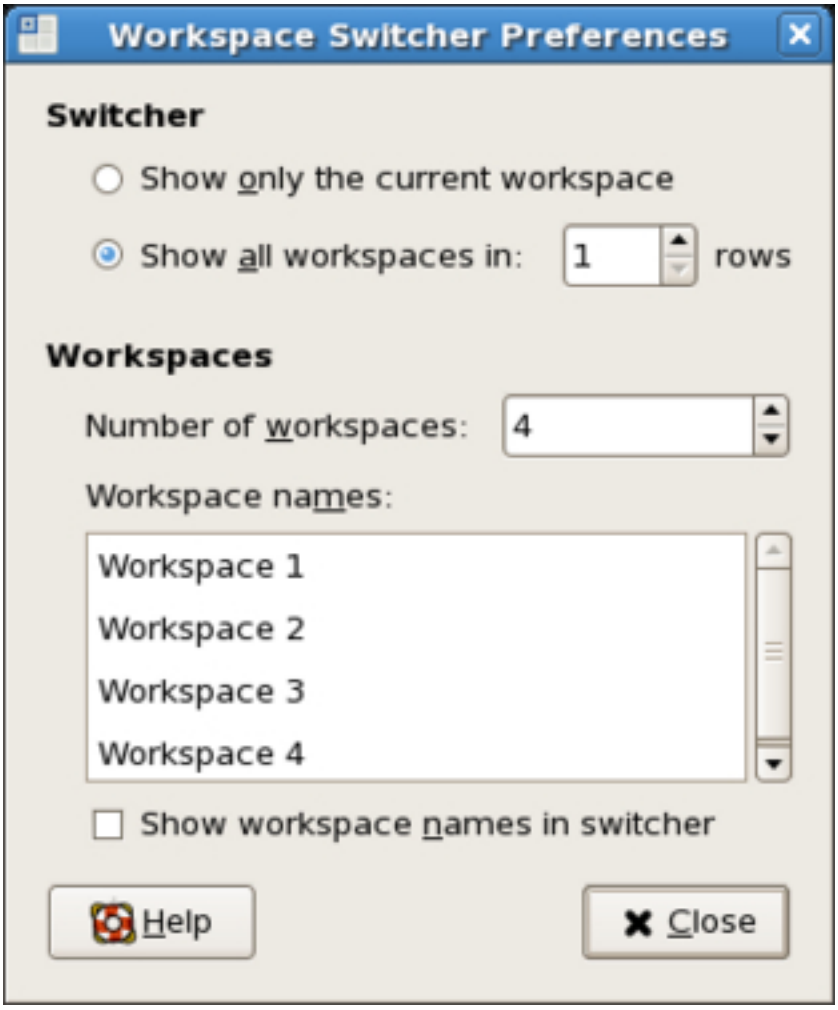
You can switch between workspaces in the following ways:

- In *Workspace Switcher*, click on the workspace where you want to work.
- Press **Ctrl + Alt + right** arrow to switch to the workspace on the right of the current workspace.
- Press **Ctrl + Alt + left** arrow to switch to the workspace on the left of the current workspace.

2.6.2. To Add Workspaces

To add workspaces to your desktop, right-click on the *Workspace Switcher* applet, then choose **Preferences**. The **Workspace Switcher Preferences** dialog is displayed. Use the **Number of workspaces** spin box to specify the number of workspaces that you require.

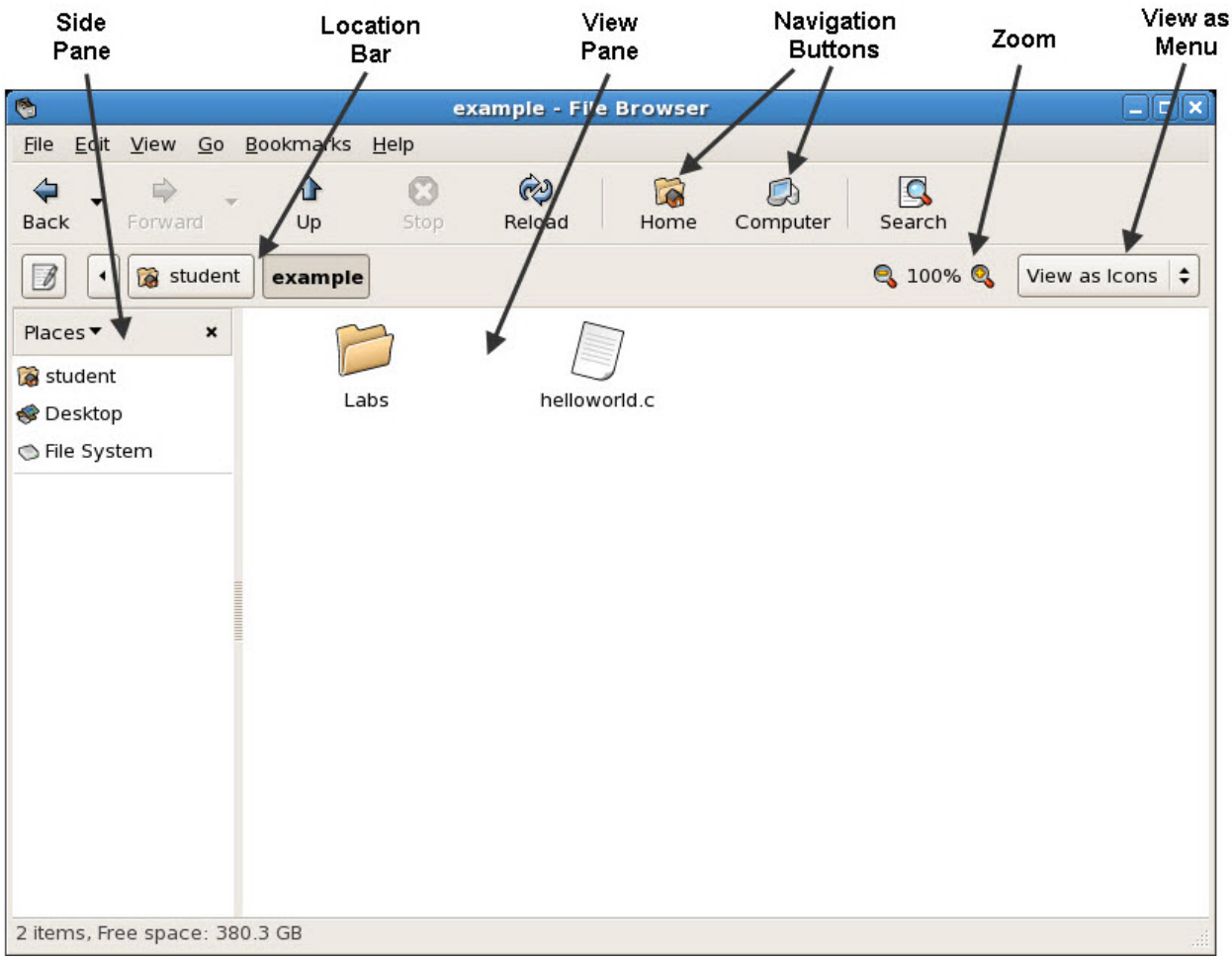
Figure 9. Workspace Switcher Preferences



2.7. Nautilus File Manager

The **Nautilus** file manager allows you to graphically manage the files and folders in your account. To open a **Nautilus** window, choose **Places -> Home Folder**. The following figure shows an example **Nautilus** window that displays the contents of a folder.

Figure 10. Nautilus File Manager



A Nautilus window contains the following components:

- Navigation Buttons

  The navigation buttons allow you to jump around to various locations in your account. The following describes the functions of each:

  - Back - Navigates back one folder in your history.
  - Forward - Navigates forward one folder in your history.
  - Up - Navigates one folder up the directory tree.
  - Stop - Stops the loading of a folder.
  - Reload - Refreshes the view of your current folder.
  - Home - Navigates to your account's home directory.
  - Computer - Navigates to a place where you can view the entire file system or access external drives (e.g. a flash drive that you've inserted).
  - Search - Search for files within that folder.

- Location Bar

  Displays the location of folder you're currently viewing. You can edit this entry by clicking on the notepad icon at the left.

- Zoom

  Allows you to zoom in and out of the current folder.

- "View as" Menu

  Allows you to view the current folder as a group of icons or as a list.

- Side pane

  Typically enables you to browse to other common locations you may want to visit.

- View pane

  Displays the contents of files and folders. The view pane is on the right side of the window.

2.7.1. To Open Files From the Nautilus File Manager

To navigate to the folder where the file that you want to open resides, double-click on the folder icons in the view pane. When the file that you want to open is displayed, double-click on the file icon to open the file.

2.7.2. To Move Files Between Folders

You can move files between folders by opening two or more Nautilus windows. Open a different folder in each window, then drag the files from one window to the other.

2.8. Desktop

The desktop is an active component of the user interface. You can use the desktop to perform the following actions:

- Start your applications, and open your files and folders.

  You can add desktop objects for convenient access to the files, folders, and applications that you use frequently. For example, you add a launcher for an application that you use often.

- Open the Desktop menu.

  Right-click on the desktop to open the Desktop menu. You can use the Desktop menu to perform actions on the desktop.

The file manager manages the desktop.

2.8.1. To Open Desktop Objects

To open an object from the desktop, double-click on the object.

2.8.2. To Add Objects to the Desktop

You can add desktop objects for convenient access to files, folders, and applications that you use frequently. You can add objects to your desktop in the following ways:

- Use the Desktop menu (by right-clicking on the desktop) to add a launcher to the desktop.
- Drag an object from a file manager window to the desktop. For example, you can create a symbolic link to a file that you use often, then drag the link to your desktop. The icon for the link is moved to the desktop. To open the file, double-click on the icon. You can also drag files and folders to the desktop.
- Drag an application launcher from a menu to the desktop. For example, you can open a menu that contains a launcher for an application that you use often, then drag the launcher to your desktop.

2.9. GNOME Preferences

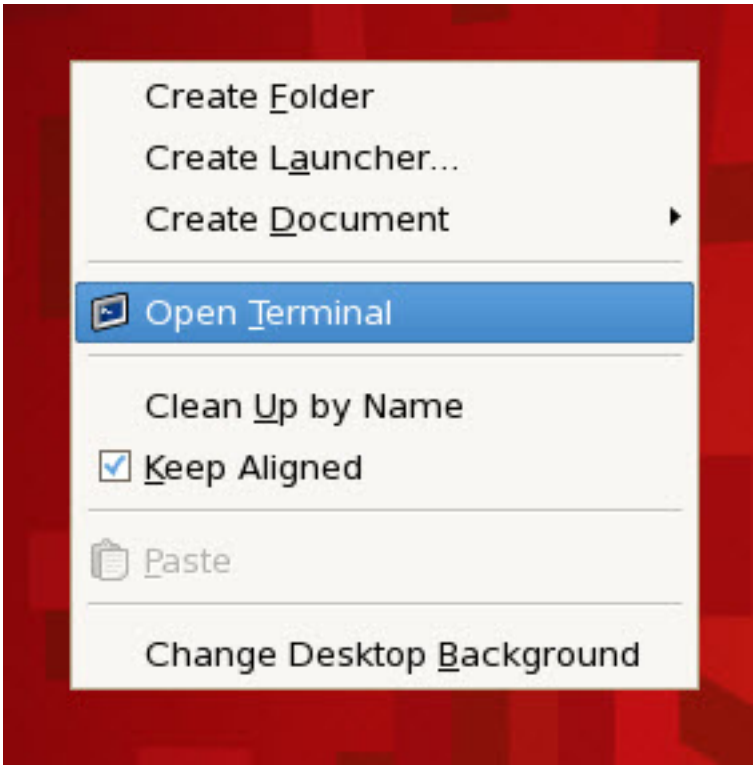
You can use desktop preference tools to configure many features of the desktop. Each tool controls a particular part of the behavior of the desktop. For example, you can use a preference tool to select a theme for your desktop. A *theme* is a group of coordinated settings that specify the visual appearance of a part of your interface. You can open your desktop preference tools by going to **System -> Preferences** and choosing the item that you require from the submenu.

3. Using the Command Line

3.1. Opening a Terminal Window

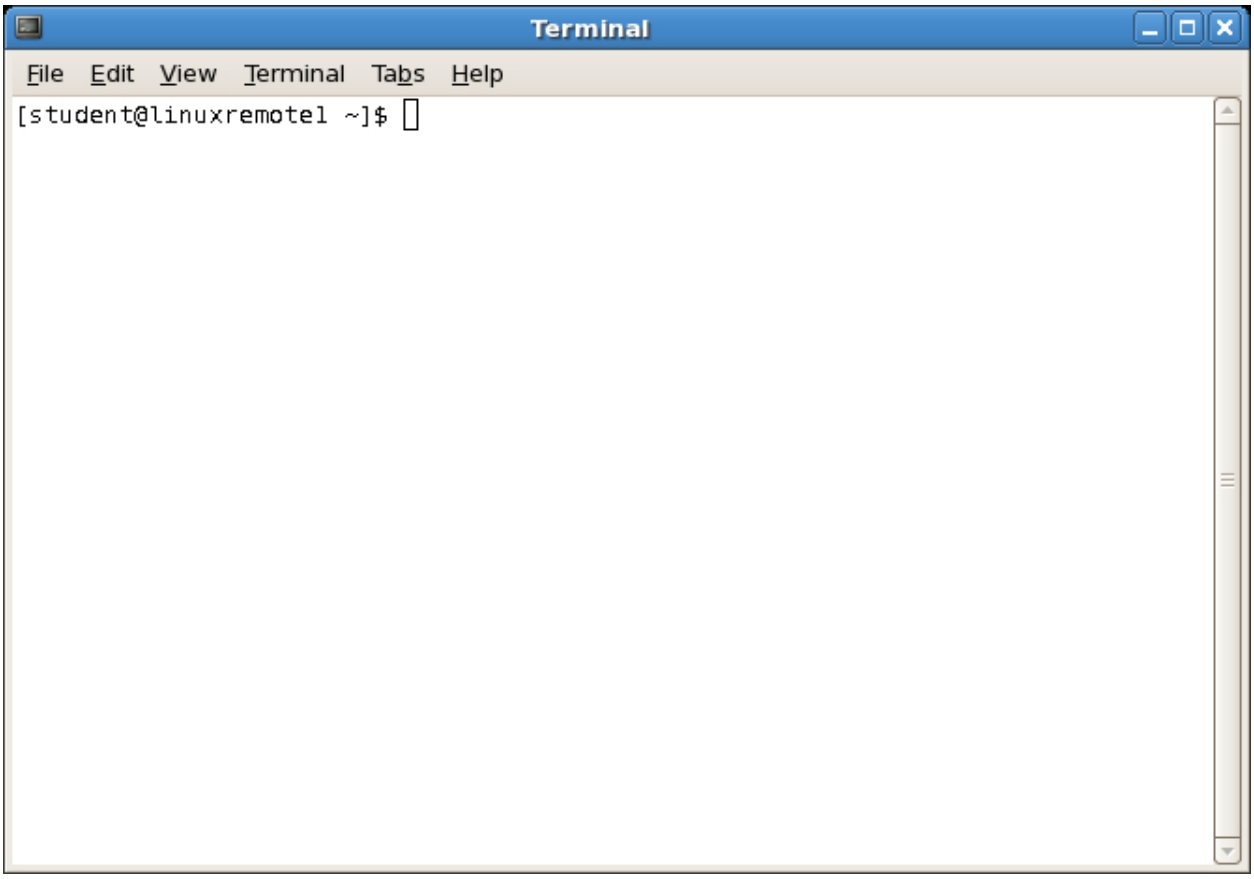
To begin using the command line while sitting at a Linux desktop/lab system, you need to open a terminal window. To so so, right-click on the desktop and select Open Terminal

Figure 11. Opening a Terminal Window



You should then be presented with a terminal window similar to the following

Figure 12. Terminal Window



3.2. Interacting with Linux from the Command Line

When Linux is waiting for the user to enter a command, it displays what we call the a prompt. The prompt is a sequence of characters on the video display (possibly a shell window) which is really a request from the operating system to the user to enter another command. A typical Linux prompt could be

```
[student@hostname ~]$ _
```

When you look at your screen, you will see the name student replaced by your username and hostname replaced by the name of the machine (or host) you are using. The value after the hostname is the directory that you're currently in. The tilda (~) means that you are in your home directory. When a prompt appears on the screen you will also see an insertion point one space past the % sign. When you type a command this insertion point is where the characters will appear (the '_' represents the insertion point. For example:

```
[student@hostname ~]$ date_
```

To issue a **date** command, you press the **Enter** key. If that were done in the previous example, you would see on your screen something similar to the following:

```
[student@hostname ~]$ date
Mon Jul 26 09:58:44 EDT 2004
[student@hostname ~]$
```

Notice that the next prompt appears automatically when the last command has completed.

In Linux, it is important to remember that the commands are usually in lower case letters. Therefore, ‘date’ works but ‘DATE’ does not! We say that Linux is case sensitive. Try both ‘date’ and ‘DATE’.

```
[student@hostname ~]$ DATE
DATE: Command not found.
[student@hostname ~]$
```

Some commands have options that are specified by a minus sign followed by characters. For example, ‘date’ may have the ‘-u’ option that displays universal time or Greenwich Mean Time.

```
[student@hostname ~]$ date -u
```

If you spell the command name correctly but supply an improper option, an improper argument or the wrong number of arguments, many of the commands will not give you an error message but only display a brief description of usage. The command assumes you mistyped something and need some assistance. Try an incorrect option for **date** and you will see an example of a usage message:

```
[student@hostname ~]$ date -p
date: illegal operation --p
usage: date [-u] mmddHHMM[[cc]y]
date [-u] [+format]
date -a [-]sss[.fff]
```

If you ever receive a ‘usage’ message, you typed something wrong. In a usage description, the square brackets ‘[ ]’ means that part is optional. Obviously from **date**'s usage message, it allows several other features.

3.3. Changing your password

The first time you log in you may want to change your password from the one given you by the computer center to one of your own choosing. A word of warning, it is important not only to choose a password which you can remember, but one which another user will not be able to guess. Your password is a security device that ensures only you have access to your files. To safeguard against unauthorized use of your account, be sure that you (and only you!) know your password. Your password should contain at least 6 characters and should not be based on a word contained in a dictionary. An easy way to meet this guideline is to choose two small words, join them together, and replace one or more of the characters with a digit. For example, you might join the words 'box' and 'door' and replace the o's with 3's (e.g. b3xd33r).

To change your password, type the **passwd** at the Linux prompt and press **Enter**.

When you press the **Enter** key, you will be asked to type in your old password. After entering your old password, you will be asked to enter the new password; after entering the new password, you will be asked to verify it by entering it again. If you don't type the same password the second time or if you incorrectly typed your old password, the system will print an appropriate error message and your password will not be changed. If this happens, just start over by typing **passwd** again.

```
[student@hostname ~]$ passwd
Changing password for user student.
Enter login(LDAP) password:
New UNIX password:
Retype new UNIX password:
passwd: password successfully changed for student
```

Don't forget your password! Don't tell anyone else your password!

3.4. Files and File Commands

Data is stored in a file. A file can contain a program (a list of instructions for the computer to follow), data, or the text of a paper. A file is like a piece of paper on which you can put lots of characters, in any order. One of the primary functions of an operating system is to maintain user files, making sure that each user has access to their own files.

Files can be created in several ways. You can create the file, say, using an editor or programs can create files. For example, the C++ compiler creates files.

3.4.1. File Names

So that you can tell your files apart, you give a name to each one you create. When you create files, you should give them names that are meaningful to you. File names can contain most printable character, but to avoid pitfalls, you should use only letters, digits, dashes, periods and underscores. The following are legal file names: lab1, LAB3, Lab1, Alongname, read.me, program1.cc, program1.java

Remember that capital letters are different from lower case ones in Linux. Therefore, Lab1 is a *different* file name from lab1.

3.4.2. What are Directories?

It is obvious that with so many people using the Linux workstations, there are a huge number of files on the systems. One needs a way to organize all of them as well as keep peoples' files separate. Linux does this with *directories* and *sub-directories* (directories inside of directories).

When you are given a Linux account, your login is associated with a directory. When you login you are placed in this directory which is called your ***home directory*** . Your home directory is a file that contains a list of all of your files (if any) and a list of all your sub-directories (if any). When you are in a directory, say your home directory, and you create a file, it is placed in that directory.

So that the Linux system knows which files are yours, it associates your computer account with your home directory. The files in your home directory and sub-directories are yours; no one else can create, change or remove them except you (or someone who has *superuser* privilege like the System Administrator).

Sub-directories are used to organize your files much like file folders are used to organize a heap of documents. For example, you might place all the files for your labs in a sub-directory called ‘labs’ and your programs in another one called ‘progs’.

3.4.3. File Suffixes

Some commands in Linux require an identifying part of the file name called a *suffix*. This suffix is part of the file name and contains a period followed by a single character or several characters. For example, the C++ compiler requires the `.cc` suffix in the file name. Therefore, `prog1.cc` is a file name that would contain a C++ program. Some popular suffixes follow:

```
.c for a C program
.cc for a C++ program
.java for a Java program
.h for a header file
.o object file (created by compiler)
.tex for a TeX document
```

3.4.4. Using Wildcard Characters in Filenames

A *wildcard character* is a symbol that you can use with many Linux commands to specify more than one file. In Linux, the wildcard character is the ‘*’ and it matches zero or more characters in that place of the filename argument. Therefore, `b*.java` matches all files starting with **b** and ending with the suffix **.java**. You could list all the Java program files in a directory starting with b by typing:

```
[student@hostname ~]$ ls b*.java
```

You can use more than one wildcard in a filename argument as shown below:

```
[student@hostname ~]$ ls do*at*
```

That would list the following files: doat, doat21, doormat, dogcatcher and documentation.

3.4.5. Copying Files

One way to create a file is to copy an existing file. The Linux command to copy a file is **cp.** After the command you type the file to be copied and a new file name. Copy two files into your directory by doing the following:

```
[student@hostname ~]$ cp ~csci203/examples/humpty myhumpty
[student@hostname ~]$ cp ~csci203/examples/jack_theory .
```

In the first line we copied a file called `humpty` in the sub-directory `examples` under csci203's home directory (~username is a user's home directory and csci203 is a username) and placed the contents of the file into a newly created file called `myhumpty`. The second line is the same except that we were lazy and typed a period that tells Linux to use the same name. You made no changes to csci203's existing files.

The long string of directory/sub-directory/filename is an example of a *pathname* that is how you specify any file on the system. Many commands take pathnames as possible arguments (see the section entitled "Using Pathnames").

The command **cp** may use a wildcard character in the filename argument. You could have copied *all* of the files in my **examples** directory by doing the following:

```
[student@hostname ~]$ cp ~csci203/examples/*  .
```

Here the period is important to give each file the same name.

3.4.6. Listing Files in Your Directory

Now try listing your directory by using the command ls and the two files should be there.

```
[student@hostname ~]$ ls
myhumpty      jack_theory
```

Try ls with the -a option.

```
[student@hostname ~]$ ls -a
```

You should see several file names which begin with a period, for example a `.cshrc` file. These *dot* files control how your system looks when you login and were placed there by the Systems Administrator. If you remove or alter these files you may experience many problems and may find you can't even login! If this should ever happen see your instructor immediately. Later, when we are more advanced, we may alter some of these files. Try **ls** with the **-l** option to list the files in extended form.

```
[student@hostname ~]$ ls -l
```

You can combine the two options by typing -al.

```
[student@hostname ~]$ ls -al
```

You can also list a filename or a filename argument with a wildcard. For example, you could list all the files starting with 'lab' by:

```
[student@hostname ~]$ ls -l lab*
```

3.4.7. Displaying the Information in a File

The command **more** followed by the filename displays the contents of that file. Try **more** on `myhumpty` and `jack_theory`. Hit the spacebar to display the next window full. While using **more** on `jack_theory` press an **h** and read the on-line help of the commands available in **more**. Hit the spacebar to get out of the help display. To go back a page, type "**b**." To quit type "**q**." Using the command **cat** can also complete this operation. However, the latter command scrolls through the entire document without page pausing. Note this command would *not* be desirable for displaying long documents.

```
[student@hostname ~]$ more jack_theory
[student@hostname ~]$ cat jack_theory
```

Like most commands that take a filename argument, **more** allows a wildcard character (the '*'). Therefore, we

could be lazy and type:

```
[student@hostname ~]$ more j*
[student@hostname ~]$ cat j*
```

That will do **more** on all files in the current directory beginning with "**j**." In this case, there is only one: `jack_theory`.

3.4.8. Removing Files

Many times we need to clean up our directories and we desire to remove some files. The command **rm** removes or deletes the file specified by the filename.

Copy your file `myhumpty` to several new file names, say `junk` and `baloney`. Then remove `junk` and `baloney` from your directory. Using **ls**, check that they were removed.

```
[student@hostname ~]$ rm junk baloney
[student@hostname ~]$ ls
```

To avoid inadvertently deleting the wrong file, type carefully when you use **rm**. The command **rm** allows a wild-card but you should be very careful when you use it or you will remove files you wanted to keep.

3.5. Using Directories

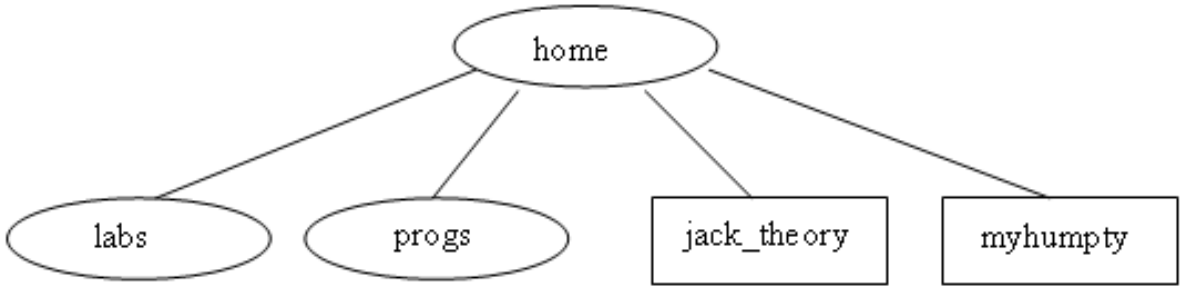
After becoming comfortable with Linux, most users use many directories to organize their files. For example, you could make a new directory for each new programming assignment.

3.5.1. Making a New Directory

To create a new directory, use the command **mkdir** which stands for *make directory*. The conventions for names of directories is the same as the names of files, i.e., you should use letters, digits, dashes, periods and underscores. From your home directory make two directories (also called sub-directories) named `labs` and `progs`.

```
[student@hostname ~]$ mkdir labs
[student@hostname ~]$ mkdir progs
```

Check using **ls -l** to see that they were created. The '**d**' in the first column means they are directories.



3.5.2. Moving Between Directories

The above diagram shows the structure of the three directories and the two files we have made so far (we are *not* showing the dot files). To change the current working directory to a new one, use the **cd** command. To change to the `labs` directory type:

```
[student@hostname ~]$ cd labs
```

Try an **ls** command now. You will see no files as you haven't created any in this directory. To return to your home directory, you use the **cd** command and no file name.

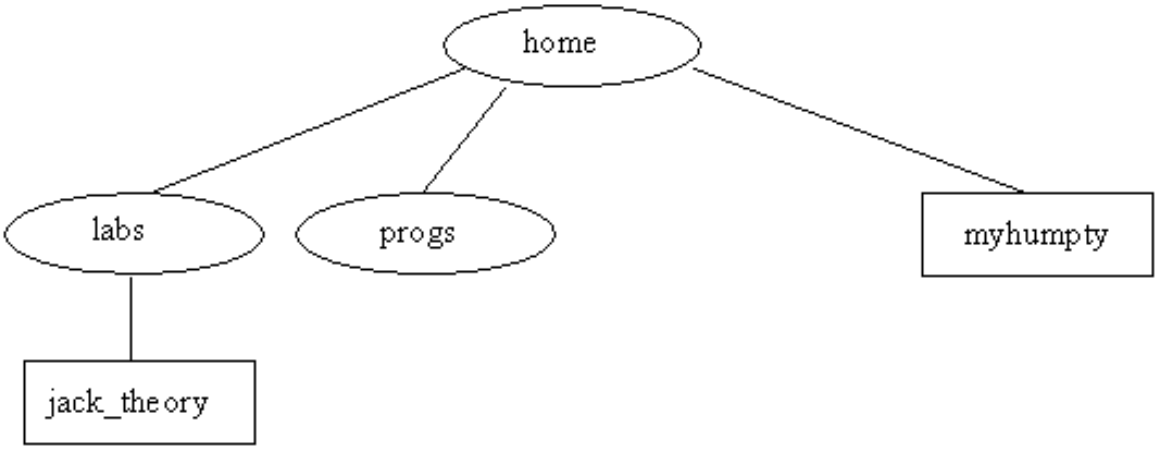
```
[student@hostname ~/labs]$ cd
```

3.5.3. Moving or Renaming a File

When you *copy* a file you end up with two files - the original file and the new one. Sometimes you may wish to *move* a file to another directory or keep it in the same directory but with another name. The command **mv** (for move) does both.

Move to your home directory if you have not done so. Move the file `jack_theory` from your current directory to the `labs` directory by typing the following:

```
[student@hostname ~]$ mv jack_theory labs
```



Using the **cd** command, change your working directory to `labs` and check that `jack_theory` is there.

3.5.4. What is My Current Working Directory?

You can always tell what is your current working directory by using the **pwd** command (print working directory). **pwd** is a very useful and important unix command.

```
[student@hostname ~]$ pwd
```

Notice that **pwd** displays the whole pathname with something like:

```
/home/accounts/student/initial_letter/username/labs
```

Change your current working directory to your home directory and try **pwd** there.

3.5.5. Using Pathnames

Most, if not all, the Linux commands that use a filename can also take a pathname. A *pathname* is a string of directory names separated by '/'s where the last entry is a file name. The full pathname specifies where a file resides. You can see the full pathname when you type a **pwd** command. However, you don't need to use the whole pathname most of the time. There are shortcuts that we will describe below. If you are in your working directory (in this case your home directory) and want to display the contents of `jack_theory` in the sub-directory `labs`, you could use **more** with a pathname.

```
[student@hostname ~]$ more labs/jack_theory
```

From your current working directory you can go 'down' the directory structure by using the directory name, a '/' and file name. You can go 'up' the directory structure by using '..'s. For example, change your working directory to `labs` and type the following:

```
[student@hostname ~]$ more ../myhumpty
```

We could also have used "~" to specify your home directory because `myhumpty` is in your home directory.

```
[student@hostname ~]$ more ~/myhumpty
```

You can have as many layers of directories as you want. Make a new directory called `lab1` under `labs`.

```
[student@hostname ~]$ cd  ~/labs
[student@hostname ~/labs]$ mkdir lab1
```

As another example of using pathnames, we will copy a file in the home directory to the new `lab1` directory two different ways. First, while in the home directory we will issue the **cp** command to make a copy of `myhumpty` called `humpty1` and place it in the directory `lab1`.

```
[student@hostname ~]$ cd
[student@hostname ~]$ cp myhumpty labs/lab1/humpty1
```

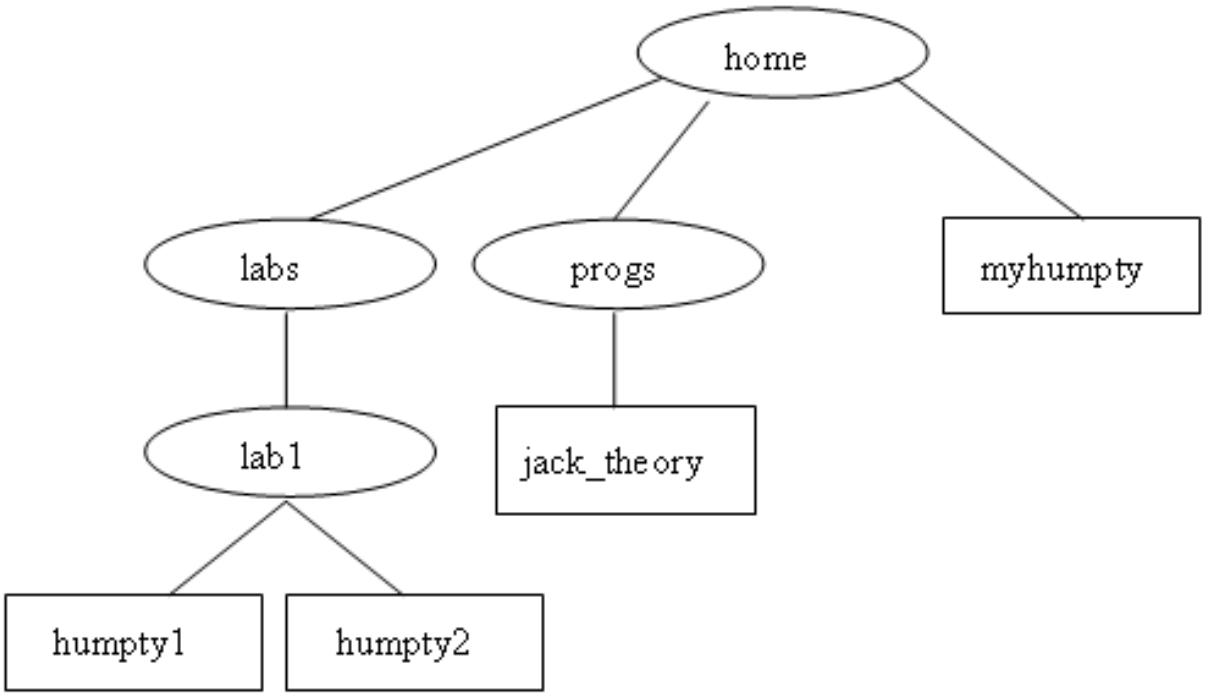
Now we move to the `lab1` directory and copy the file from there and name the new copy `humpty2`.

```
[student@hostname ~]$ cd
[student@hostname ~]$ cd labs/lab1
[student@hostname ~/labs/lab1]$ cp  ~/myhumpty  humpty2
```

Move to the `lab1` directory and make sure the two humpties are there.

For a final example, we will move `jack_theory` from `labs` to `progs` with the same name. Assuming we are in the `labs` directory, we go 'up' one directory with '..' and 'down' another by specifying the directory name.

```
[student@hostname ~]$ cd ~/labs
[student@hostname ~/labs]$ mv jack_theory ../progs/.
```



Above are the directories and files we have now. It is important that you understand how to move around in directories and how to specify the correct pathname. If you are still confused in any way, go over this section again carefully studying the commands and using the diagrams.

3.6. Reading the Online Manual Pages

Linux has manual pages stored on-line for most commands. The command **man** followed by the command name

displays these reference manual pages.

To read the manual pages on the command **man**, type the following:

```
[student@hostname ~]$ man man
```

Since the command **man** uses **more** to display the text, you use **more**'s commands to get around, i.e., the space bar for the next page, **b** to go back a page, and **q** to quit. The manual pages are not written for the beginning user. Therefore, the beginner needs to learn how to extract useful information by selectively skipping over sections and ignoring large portions. Try reading some other man pages, say for **ls** and **cd**.

If you don't know the command you want but want to find an appropriate one, you can try a keyword search of man. Using the **-k** option and the keyword does this. Say you want to copy a file but can't remember the command. Trying '**man copy**' gives the message "No manual entry for copy" that means there is no command spelled "copy". We could try the following keyword search:

```
[student@hostname ~]$ man -k copy
```

And find lots of commands that do copying including **cp** that is what we wanted.

3.7. Printing Files

To print a text file (a file which you can edit), you use the **lp** command where you specify the printer and the name of the file. The default printer is whatever printer is in the lab you're currently using. To print the Java source file called `myprog1.java` you type the following:

```
[student@hostname ~]$ lp myprog1.java
```

3.7.1. Printer Names

An option **-d** and the name of the printer specify the different printers. The names of the printers are the following:

```
brki167-lp1 laser printer in Breakiron 167 (specifically for Jr/Sr CS Majors).
brki164-lp1 laser printer in Breakiron 164.
dana213-lp1 laser printer in Dana 213.
```

For example, to print a letter to Mom on the laser printer in Breakiron 164, you would type the following:

```
[student@hostname ~]$ lp –dbrki164-lp1 letter.mom
```

The command **lp** only prints what's in the file. It does not include the date or even the filename! One uses the **pr** command to add the date, time, name of file and page number at the top of each page. Normally one uses the following for printing programs and output which is to be handed in for a class.

```
[student@hostname ~]$ pr myprog1.cc | lp
```

In the above, **pr** formats the C++ file `myprog1.cc` and pipes (read about pipes in Section 2.8.6 on "Pipes and Filters.") the formatted output to **lp** to be printed. You may choose the alternate **two columns** output format for your submission. You would type the following to submit your homework:

```
[student@hostname ~]$ a2ps –Pdana213-lp1 myprog1.cc
```

3.7.2. Checking a Printer Queue

Since several people may request to print a file on a printer at the same time, your request is placed in a queue. You can display the contents of a printer's queue with the **lpstat** command where you specify the printer name by using the **–o** (lowercase letter o) option. To check the queue of the laser printer in room 164 Breakiron, you would type.

```
[student@hostname ~]$ lpstat –o brki164-lp1
```

3.7.3. Removing Requests from a Printer Queue

At times you find you may need to remove a request from a printer's queue. Perhaps, you discover that you printed the wrong file and don't want to waste paper. First, perform a **lpstat** command and notice the job ID of the print request to be removed. Then issue a **cancel** command specifying the job ID. Below we remove job ID brki164-lp1-20 from the line printer queue.

```
[student@hostname ~]$ cancel brki164-lp1-20
```

On rare occasions, a print request will hang the printer. If there are print requests in the queue and the printer has not printed for many minutes, perhaps the first (top) job in the queue is problematic. After checking that the printer is turned on, has paper, not jammed, is on-line, etc., then remove the first job from the queue with **cancel**. You can remove only your own printer requests. If the first job is not yours then you need to find that person or report the problem to the System Administrator.

3.8. Other Unix Commands

This section covers some Linux commands that are important but less frequently used.

3.8.1. Grep - Searching a File for a String

To search for a word, a phrase or any string of characters in a file use the **grep** command. You specify the string of characters then the file as shown.

```
[student@hostname ~]$ grep Humpty humpty
Humpty Dumpty sat on a wall.
Now Humpty's unscrambled and good as new.
```

Two useful options for **grep** are **-i** that ignores the case of letters and **-n** that displays the line numbers at the front of the line. Here is the output after one asks **grep** to find all occurrences of '**and**' including both upper and lower case letters and displaying the line numbers.

```
[student@hostname ~]$ grep -i -n and jack_theory
37:  This is Chaotic Confusion and Bluff
46:  This is the Cybernetsics and Stuff
47:  That covered Chaotic Confusion and Bluff
49:  And thickened the Erudite Verbal Haze
57:  To make with the Cybernetics and Stuff
58:  To cover Chaotic Confusion and Bluff
60:  And thickened the Erudite Verbal Haze
69:  That made with the Cybernetics and Stuff
72:  And, shredding the Erudite Verbal Haze
77:  And Demolished the Theory Jack built
```

As with many Linux commands, **grep** can take a wildcard for the filename. (see the section entitled "Using Wildcard Characters in Filenames.") Here we search for 'to' in all the files of the directory.

```
[student@hostname ~]$ grep -i -n to *
humpty:3:The King set the Tme Machine back to two.
jack_theory:56:  This is the Button to Start ther Machine
jack_theory:57:  To make with the Cybernetics and Stuff
jack_theory:58:  To cover Chaotic Confusion and Bluff
jack_theory:68:  Who pushed the Button to Start the Machine
```

Notice that it displayed the filename before the line number.

3.8.2. How Much Disk Space Am I Using?

All student accounts are limited in the amount of disk space (called **disk quota**) that they may use. This is typically fifty megabytes. If you bump up against this limit, you won't be able to create new files. If you exceed your quota, you must first delete some files. If you find you can't delete any other data (because you need it), then contact the System Administrator about increasing your quota.

To see what your disk usage and quota are use the **quota** command.

```
[student@hostname ~]$ quota -v
```

```
Disk quotas for user student (uid 15341):
        Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
medusa.eg.bucknell.edu:/vol/vol1/home
                          57956   92160  102400           3035  4294967295 4294967295
```

The **blocks** column shows your current disk usage (in kilobytes) and the first **limit** column shows your disk quota (in kilobytes).

Doing an **du -sh *** (disk usage) in a directory will show the size of each direcotry and files.

```
   4K   helloworld.c
   4K   Labs
```

The **-h** option tells du to print the size in a human readable format. **K** means kilobytes (about 1,000 bytes), **M** means megabytes (about 1,000,000 bytes), and **G** means gigabytes (about 1,000,000,000 bytes).

3.8.3. Changing Permissions on a File

When you do an **ls -l** command, the current permissions are displayed for each file. The first 10 characters are the permission fields for that file. A '-' in column one means a plain file. A 'd' in first column means a directory.

```
-rw-r--r--   1 student    student        245 Aug 14 14:16 humpty
-rw-r--r--   1 student    student       2147 Aug 14 14:16 jack_theory
```

The next nine characters are grouped into three characters each (**rwx, r-x and r-**) for the permissions for *owner*, *group* and *world* respectively. The owner of the file is the login name that created it. In the above, both files are owned by **csci203**. The three characters for the owner's permission are '**rwx**' which means the owner can **read**, can **write** and can **execute** the file. The System Administrator determines which login names are in a group. In this case, others in **cs's** group can read and execute the files but can't write or change them. The world is all others on the system. Anyone can read the files but can't changed them or execute them.

For a directory, execute permission is interpreted to mean permission to search the directory. The permissions and file modes are summarized as follows:

```
r or '4'  the file is readable;
w or '2'  the file is writable;
x or '1'  the file is executable;
-         the indicated permission is not granted.
```

The **chmod** command allows the owner to change the permissions on a file. The form of chmod is:

```
chmod <who><op><permission> filename

where <who> is a combination of:
u       User&apos;s permissions (owner).
g       Group permissions.
o       Others (world).
a       All.

<op> is one of:
+       To add the permission.
-       To remove the permission.

<permission> is any combination of:
r       Read.
w       Write.
x       Execute
```

A few examples, will help you to understand how to use **chmod**. To set the file `humpty` so no one but the owner can read, write or execute it, we would type the following:

```
[student@hostname ~]$ chmod go-rwx humpty
```

Note: no spaces are allowed in '**go-rwx**'. Doing an **ls -l humpty** shows the following permissions.

```
-rwx------  1 student    student       161 Jul 30 16:30 humpty
```

To set **humpty**'s permissions back to what they were before, we need to do two **chmod** commands. One for the group and another for others.

```
[student@hostname ~]$ chmod  g+rx  humpty
[student@hostname ~]$ chmod  o+r  humpty
[student@hostname ~]$ ls -l humpty
-rwxr-xr--  1 student   student        161 Jul 30 16:30 humpty
```

If you want to hide all your programs so only you can read them, you could use a wildcard. The below changes all the files in the directory so only you, the owner, can read, write and execute them.

```
[student@hostname ~]$ chmod  go-rwx *
```

Or you could have changed the permissions on the directory of where the files reside.

For instance, here are a few of the common ones:

```
644 is rw-r--r--
     the owner can read and write the file or directory, everybody else
     can only read it.

755 is rwxr-xr-x
     the owner can read, write and execute the file, everybody else can
     read or execute it.  For a directory, this mode is equivalent to 644.

711 is rwx--x--x
     the owner can read, write and execute the file, everybody else can
     only execute it.
```

3.9. Some Useful Features of Linux

3.9.1. Using History of Your Commands

Most users find themselves issuing the same command repeatedly during a work session. Since Linux commands can be lengthy and subsequently tedious to type out, there are shortcuts to repeating a command. For these shortcuts, Linux provides a history feature, which records the last several commands (typically the last 40) for each shell window (a window with hostname % prompt).

To display the last so many commands, type **history**.

```
[student@hostname ~]$ history
```

If you have an alias set for the history command, you should be able to type just ‘h’.

```
[student@hostname ~]$ h
```

Was ‘**h**’ defined? If not, you can add the alias later when we cover ‘dot’ files and command aliases in the next *two* sections.

When you display the history, you will see an associated number with the commands you typed before. To redo a command, type “!” followed by the number (No spaces between “!” and number!) For example to redo the command with 23.

```
[student@hostname ~]$ !23
```

To do the last command, you type “**!!**”. You can redo the last command starting with a string by typing “**!**” followed by the string. For example to repeat the last use of **more**, you would type:

```
[student@hostname ~]$ !m
```

3.9.2. Changing Dot Files

When you issue the **ls -a** command in your home directory, you should see a series of dot files or filenames that start with a period. Below are some common dot files and their purposes. You may display them with **more** to see what is in them. Be sure you know what you are doing before you make any changes!!

.cshrc	The .cshrc file is executed every time you start a new shell. This is where to place your own command aliases. See next section.
.gnome, .gnome2, .gconf	These dot directories sets features for GNOME.
.login	This file is executed whenever you login.
.emacs	This file is used by Emacs to set up features.

3.9.3. Creating Your Own Command Aliases

It is very easy to customize your frequently used Linux commands by adding aliases to your `.cshrc` file. Using 'more' display your `.cshrc` file in your home directory. Notice that there are some aliases already existed. For example that you can type 'lo' for 'logout' and 'm' for 'more.' If you want to add your own aliases just edit this `.cshrc` file and you may use the aliases already there as models. Here are some examples:

```
alias cls clear
alias pinebox 'ssh pine.bucknell.edu'
```

3.9.4. Redirecting Input and Output

Redirecting the input to and the output (I/O) from a command is easy in Linux. To send the output of any command or program to a file instead of the screen, use >.

For example, to place a list of the files in the current directory in a file called 'myfiles,' you would type the following:

```
[student@hostname ~]$ ls > myfiles
```

Again for any command or program, to read input from a file instead of the keyboard, use <. For example, to input from a file called `data` for a program called `myprog` type:

```
[student@hostname ~]$ myprog > data
```

The symbol >> operates much as > does, except that it means, "add to the end of the file." Redirecting input from a file and output to a file is especially useful with writing programs from other computer languages.

3.9.5. Background Processes

Many commands, e.g., **emacs**, don't finish after a few seconds and, therefore, tie up the window in which you typed the command. Linux allows you to easily place the command (or program) in the *background* and run by itself. Then it is no longer associated with the window where you typed the command. This allows you to continue using the window's screen and keyboard. This is called *submitting a background process*. Processes are just instances of programs in execution. Any command (or program) can be submitted as a background process by adding "**&**" on the end of it. For example, to sort a large file and redirect the output to the file sort-output and not tie up your window while doing it, one would type the following:

```
[student@hostname ~]$ sort large-file > sort-output &;
```

Some commands when run under GNOME, e.g. emacs, create a special window to run in. If run in the background, the window stays until you quit or kill it.

To see a list of your current processes including both *foreground* and *background* processes, type the following:

```
[student@hostname ~]$ ps -ef
```

The '-ef' are options to the **ps** command. Do a **man ps** to read about all the options.

In desperation, you can kill a runaway or hung up process. Use the **ps -ef** command to find the process-id. Then type

the following:

```
[student@hostname ~]$ kill process-id
```

3.9.6. Pipes and Filters

Pipes and *filters* are some of the niftiest features of Linux. A Linux command's output may be redirected directly to another command's input by the pipe symbol |. For example, to list a directory with too many files to display in the window at one time, one can pipe the output of ls into ‘more’ by the following:

```
[student@hostname ~]$ ls -l | more
```

This is similar to redirecting the output of **ls** to a temporary file then displaying the contents of the file using ‘**more**’ as follows:

```
[student@hostname ~]$ ls -l > temp
[student@hostname ~]$ more temp
[student@hostname ~]$ rm temp
```

The use of the pipe symbol | is a lot handier than using a temporary file especially since we tend to forget to remove the temporary file. Pipes are really handy as shown by the below examples.

To count the number of users on system, one could type:

```
[student@hostname ~]$ who | wc -l
```

where the **-l** (that's minus el) is an option to **wc** (word count) to only display the number of lines.

To look for a particular user “csci203” when you do a **ps** command, type the following:

```
[student@hostname ~]$ ps -ef | grep csci203 | more
```

In the previous example, we would say that **grep** is a *filter* as it filters or selects only portions of the output. The command **grep** displays each line in which it finds the string (see the section entitled “Grep - Searching a File For a String”).

Assume you have written a C++ program that displays the names (last name first) of the members of your organization but they are not in alphabetical order. To sort the names and then print them, you could pipe the output of the C++ program to **sort**, then pipe its output to **pr** to format it and then pipe its output to the printer.

```
[student@hostname ~]$ names | sort | pr | lpr
```

However, if you do this, make sure the amount of output to the printer is reasonable.

3.10. A Brief Summary of some UNIX commands

Files and Directories Commands:

ls	- list files in current directory.
ls -a	- list dot files in current directory.
ls -l	- list files in current directory in full form.
pwd	- print working (current) directory.
mkdir name	- make new directory.
cd pathname	- change to directory.
cd	- change directory to home directory.
rmdir name	- remove directory.
cat file	- display a file; concatenate several files.
more file	- display file on screen pausing at each screenful. Press

	space to continue. Press h for help.
cp file1 file2	- copy file1 to file2.
mv file1 file2	- move file1 to file2 (rename).
mv file dir	- move file to directory dir.
rm file	- remove file (delete).
du	- disk space used (do in home directory).
quota -v	- displays your disk quota or allotment.
chmod <who><op><permission> file	- change permissions on a file.

Useful Commands:

a2ps -Pprinter file	- prints file in two column format.
lp –dlobby file	- prints file on laser printer in Dana lobby.
lp –dbrki167-lp1 file	- prints file on laser printer in room 167 Breakiron.
lp –ddana213-lp1 file	- prints file on laser printer in room 213 Dana.
lp –dbrki164-lp1 file	- prints file on laser printer outside room 164 Breakiron.
lpstat –o printer	- list printer queues.
cancel job-id	- remove job from printer queue (get job-id from lpstat).
man command	- displays manual page to screen (on-line help).
man -k keyword	- displays one line of commands which contain keyword.
logout	- to log off of UNIX.
script	- to capture a work session in a file.
exit	- to end a script session. To end a rlogin session.
who	- displays who is on your machine.
ssh hostname	- remote login to another computer (exit to get off).
date	- displays time and date.
wc file	- word count of file.
cal	- prints a calendar.
spell file	- displays misspellings of the file
sort file	- displays file sorted alphabetically.
grep 'string' file	- displays all lines in file containing the string.
history	- history of commands you have typed (!number to redo).
ps	- display your processes running on your machine.
ps -ef	- display all processes running on your machine.
kill process-id	- kill a runaway process (get process-id with ps).
passwd	- change your password.

4. Remotely Accessing the Linux Servers

4.1. Using the Linux servers from a PC or a Macintosh

The Linux servers may be used from a personal computer, say in a lab or your dorm room, or from a Macintosh on campus.

4.1.1. Using the Linux servers from a Public Macintosh

You can access Linux servers from a Mac by using **Secure Shell (SSH)**, a program that simply gives you a command line interface. You will not be able to use a mouse in that window, and you will not be able to run any application that requires a graphical environment.

1. Double click on the **Terminal** application, which is located in **Applications - Utilities**
2. Type the command **ssh username@linuxserver**, where username is your Linux username and linuxserver is an appropriate Linux server name (e.g., **linuxcomp1.eg.bucknell.edu**), in the terminal window.
3. You will then be prompted for your Linux password. You will login without a windowing environment. Therefore, you will only have one window and the mouse is inoperative. However, the arrow keys still work.
4. To Logout of the Linux server type **logout** as usual from hostname prompt.

4.1.2. Using the Linux servers from a public Windows PC

You can access Linux servers from a laboratory PC in two ways:

1. By using **Secure Shell (SSH)**, a program that simply accesses the Linux servers and gives you a command line only interface. You will not be able to use a mouse in that window, and you will not be able to run any application that requires a graphical environment.
2. By using **X-Win32** a program that simulates X Windows.

To use **SSH** from a laboratory PC:

1. From the **Start** menu, select **All Programs –> Internet Applications –> Secure Shell Client.**
2. In the SSH window that appears, click the **Quick Connect** button.
3. Type in a Linux server name and your Linux username in the appropriate fields; then, click the **Connect** button.
4. A dialog box *may* appear asking if you'd like to save the new host key in your local database. Simply click **Yes**.
5. Type in your password when prompted to do so and then click **OK**.
6. You will be logged in without a graphical environment. Therefore, you will only have one window and the mouse is inoperative. However, the arrow keys still work.
7. To Logout, type **logout** as usual from the prompt.

To use **X-Win32** from a laboratory PC:

1. Click on the **Start Menu** button at the bottom left corner and select **All Programs –> Internet Applications –> X-Win32**.
2. After launching X-Win32, a small **X** should appear in the lower right hand corner of your screen.

3. Click on the **X** and select one of the sessions, e.g. **Linuxcomp1 Terminal** or **Linuxcomp1 Graphical**. **Terminal** sessions will simply give you a command line interface and you can launch applications via the command line. **Graphical** sessions will present you with the interface that you receive when physically sitting at a Linux workstation.

4. Simply login with your Linux username and password.

5. To exit X-Win32, right-click on the **X** and select **Close**.

4.2. Transferring files to and from the Linux ystems

4.2.1. How to use FTP in Linux

The **lftp** command invokes the Internet standard File Transfer Protocol (FTP), which enables file transfers to and from a remote machine.

Type **lftp -u username servername**

```
[student@hostname ~]$ lftp -u student ftp.netspace.bucknell.edu
Password:
lftp student@ftp.netspace.bucknell.edu:~>
```

From the **lftp** prompt, you can type your necessary commands. The most common **lftp** commands are:

help - displays all the available ftp commands

quit - terminates the FTP session with the remote server and exit ftp

cd - changes the working directory on the remote machine to remote-directory

dir - prints a listing of the directory contents in the directory

get - retrieves the remote file and stores it on the local machine

put - puts a local file onto the remote machine

To get more detailed information about **lftp** from the man page, simply type **man lftp**.

```
[student@hostname ~]$ man lftp
```

4.2.2. Transferring files from a Mac to the Linux systems

There are two ways to transfer files between a Mac and the Linux systems: one is by using **FETCH**, and the other is by using **CIFS/SMB**. CIFS/SMB has a simpler interface which allows you to interact with your files just as you would with any file on your local Mac system.

4.2.2.1. Transferring files betweena Mac and the Linux systems by using CIFS/SMB

The procedure for using CIFS/SMB is as follows:

1. From the **Finder**, select **Go – Connect to Server…**

2. In the dialog box that appears, type in **smb://unixspace.bucknell.edu/linux-username.$** (where **username** is your actual Linux username) and click the **Connect** button.

3. Type in **BUCKNELL** in the **Workgroup/Domain** field and enter your **username** and **password** in the appropriate fields. NOTE: The password you need is your Windows network password, NOT your Linux password. There

is a trust relationship that allows you to use your Windows password, in this case, to access your Linux files.

4. Click the **OK** button.
5. After logging in, a volume icon will appear on your desktop labeled **linux-username.$**
6. You can now interact with this volume as you would with any other set of folder/files on your local machine.
7. To **exit**, drag either the **linux-username.$** icon icon to the **Trash Can/Eject Button**.

4.2.2.2. Transferring files between a Mac and the Linux systems by using FETCH

1. Open the **Applications** folder.
2. Double click on **Fetch**.
3. In the **New Connection**... window, type in a valid hostname (e.g. **linuxcomp1.eg.bucknell.edu** or **linux-comp2.eg.bucknell.edu**) for the Host.
4. On a line that says: **Username**: type your username.
5. On a line that says **Connect Using**: select **SFTP**.
6. On a line that says **Password**: type your password.
7. Click the **Connect** button.
8. To transfer files to/from the Linux systems, simply drag and drop files from your local machine to/from the Fetch window.
9. To quit **Fetch**, select **Fetch – Quit Fetch**.

4.2.3. Transfering Files from a Public PC to the Linux Systems

There are two options to transfer files between a laboratory Windows PC and the Linux systems. The first option is to use the **Secure File Transfer Client (SFTP)** from **Internet Applications**; the second choice is to map a network drive to your Linux space.

4.2.3.1. Transferring files between a PC and the Linux system by using SFTP

1. Go to the **Start** Menu and select **All Programs -> Internet Applications –> Secure File Transfer Client**.
2. In the window that appears, click the **Quick Connect** button.
3. Type in a Linux server name and your Linux username in the appropriate fields; then, click the **Connect** button.
4. A dialog box *may* appear asking if you'd like to save the new host key in your local database. Simply click **Yes**.
5. Type in your password when prompted to do so and then click **OK**.
6. Your Linux files should then appear in the right window. You can now drag files to the local machine from this window.
7. Simply exit **SFTP** by closing the window when you're finished.

4.2.3.2. Transferring files between a PC and the Linux systems by using a mapped network drive

1. On the desktop, right-click on **My Computer** and select **Map Network Drive**…
2. In the dialog box that appears, select a free drive letter (usually the default is fine) in the **Drive** field.
3. In the **Folder** field, type **\\unixspace\linux-username.$** and click the **Finish** button.
4. A window containing your Linux folders/files should appear, and you can now interact with this volume as you would with any other set of folder/files on your local machine.

5. Data Backup & Restore

5.1. Manual Backups (Archiving Data)

At some point you may wish to archive some files and/or folders so that you can save these to some sort of longterm storage (e.g. CD-R, another hard drive, etc.). You can use the **tar** command to create a tar archive (basically a "bundle" of files and folders) and then compress the resulting tar file.

To create this compressed archive, do something like the following:

```
[student@hostname ~]$ tar -zcf archive.tar.gz filename folder
```

The **c** option tells gtar to *create* the archive. The **f** option specifies that the filename will be the next argument, and the **z** option tells **gtar** to compress (gzip) the tar archive. All other filenames and/or folders specified after the archive name will be added to this compressed archive. So essentially this will create a file named `archive.tar.gz` containing the file `filename` and the `folder`.

5.2. Automated Backups

There are two types of automatic backups performed on your Linux data: snapshots and tape. Snapshots are an on-line, read-only copy of your recent data. Tape backups are also performed to add another layer of data protection. The primary differences between the two is that snapshots only provide access to recent changes (within the last week) but you can do your own restores. Tape backups are kept for months, but only the system administrators can restore this data for you.

5.2.1. Snapshots

5.2.1.1. Understanding Snapshots

Our Network File System (NFS) file server, **unixspace**, provides file system backup called **snapshots**. A snapshot is an on-line, read-only copy of the entire file system. Snapshots can be accessed by users as read-only files to recover previous versions of files (for example, files that have been accidentally changed or deleted). This feature allows users to restore their own files without the system administrator. To ensure system security, files within a snapshot carry the same permissions as the original files.

There is a "*magic*" directory named `.snapshot` which does not normally show up in a directory listing under each user's home directory. There are also few sub-directories under `.snapshot` based on the snapshot schedule. The command to view the snapshots with access times is **ls -lu** (Note: **ls -al** will not show the correct date information).

```
[student@hostname ~]$ ls .snapshot
08132003-1200pm  08132003-1200am  08122003-1200am  08092003-1200am
08132003-0800am  08122003-0800pm  08112003-1200am  08082003-1200am
08132003-0400am  08122003-0400pm  08102003-1200am  08072003-1200am
```

or

```
[student@hostname ~]$ cd .snapshot
[student@hostname ~/.snapshot]$ ls -lu
total 2400
drwx--x--x   81 ecst  eg         200704 Aug  7 00:00 08072003-1200am
drwx--x--x   81 ecst  eg         200704 Aug  8 00:00 08082003-1200am
drwx--x--x   81 ecst  eg         200704 Aug  9 00:00 08092003-1200am
drwx--x--x   81 ecst  eg         200704 Aug 10 00:00 08102003-1200am
drwx--x--x   81 ecst  eg         200704 Aug 11 00:00 08112003-1200am
drwx--x--x   82 ecst  eg         200704 Aug 12 16:00 08122003-0400pm
drwx--x--x   82 ecst  eg         200704 Aug 12 20:00 08122003-0800pm
drwx--x--x   82 ecst  eg         200704 Aug 12 00:00 08122003-1200am
drwx--x--x   82 ecst  eg         200704 Aug 13 04:00 08132003-0400am
drwx--x--x   82 ecst  eg         200704 Aug 13 08:00 08132003-0800am
drwx--x--x   82 ecst  eg         200704 Aug 13 00:00 08132003-1200am
drwx--x--x   81 ecst  eg         200704 Aug 13 12:00 08132003-1200pm
```

The current snapshot schedule is:

- Seven snapshots at midnight for the most recent 7 days
- Six snapshots (one every four hours) for the past 24 hours—12:00am, 04:00am, 08:00am, 12:00pm, 04:00pm, 08:00pm

5.2.1.2. Restoring From Snapshots

Files in the `.snapshot` directory are **read-only**, hence they may only be copied (through **cp**). Also due to sharing of disk space, it is wise to rename the file name to avoid overwriting the same disk space. The following is the procedure to restore a file named *software.info* in Files directory:

```
[ecst@hostname ~/Files]$ pwd
/home/accounts/facultystaffstaff/e/ecst/Files
[ecst@hostname ~/Files]$ mv software.info soft
[ecst@hostname ~/Files]$ cd
[ecst@hostname ~]$ cd .snapshot
[ecst@hostname ~/.snapshot]$ ls -lu
total 2400
drwx--x--x   81 ecst  eg         200704 Aug  7 00:00 08072003-1200am
drwx--x--x   81 ecst  eg         200704 Aug  8 00:00 08082003-1200am
drwx--x--x   81 ecst  eg         200704 Aug  9 00:00 08092003-1200am
drwx--x--x   81 ecst  eg         200704 Aug 10 00:00 08102003-1200am
drwx--x--x   81 ecst  eg         200704 Aug 11 00:00 08112003-1200am
drwx--x--x   82 ecst  eg         200704 Aug 12 16:00 08122003-0400pm
drwx--x--x   82 ecst  eg         200704 Aug 12 20:00 08122003-0800pm
drwx--x--x   82 ecst  eg         200704 Aug 12 00:00 08122003-1200am
drwx--x--x   82 ecst  eg         200704 Aug 13 04:00 08132003-0400am
drwx--x--x   82 ecst  eg         200704 Aug 13 08:00 08132003-0800am
drwx--x--x   82 ecst  eg         200704 Aug 13 00:00 08132003-1200am
drwx--x--x   81 ecst  eg         200704 Aug 13 12:00 08132003-1200pm
[ecst@hostname ~/.snapshot]$ cd 08132003-0400am
[ecst@hostname ~/.snapshot/08132003-0400am]$ cd Files
[ecst@hostname ~/.snapshot/08132003-0400am/Files]$ pwd
/home/accounts/facultystaffstaff/e/ecst/.snapshot/Files
[ecst@hostname ~/.snapshot/08132003-0400am/Files]$ cp software.info ~/Files
[ecst@hostname ~]$ cd
[ecst@hostname ~/Files]$ cd Files
[ecst@hostname ~/Files]$ ls -al so*
-rw-------   1 ecst staff    2412 Jan 29 11:20 soft
-rw-------   1 ecst staff    2412 Jan 29 16:19 software.info
```

5.2.2. Tape Backups

Two types of backups are performed on our Linux data. *Daily* tape backups are performed each day and are retained for one month. *Archival* tape backups are performed once a month and are retained for one year. Please contact the system administrators if you need data restored from tape.

6. The Emacs Editor

6.1. What is Emacs?

Emacs is a powerful text editor that is available on the Linux systems. It has many advantages over other text editors in that it can provide helpful formatting, such as indenting and syntax highlighting.

6.2. Starting Emacs to Edit a File

Before you start the next section, you should copy the file ‘stories’ into your directory. Type the following to copy the file.

```
[student@hostname ~]$ cp ~csci203/examples/stories stories
```

To run Emacs, type at a **hostname %** prompt the command emacs followed by a file name and ‘&’. To create the new file named myfile type the following:

```
[student@hostname ~]$ emacs myfile &
```

The & sign tells Linux that you wish to run this program in the “background”. Basically, what this does is give you the host-name % prompt back to allow you to issue others commands in the window.

Since you probably don't have a file called ‘myfile’ in your directory, Emacs creates a new file. You should see ‘(New file)’ at the bottom of the window. Notice the black bar near the bottom. It should tell you the name of the file, the time and other information. And the black rectangle in the upper left hand corner of the Emacs window. This ‘cursor’ moves along as you type. It is important for you to distinguish between the mouse pointer and the cursor. The pointer is used to move between windows and to select items to click on. The cursor is where you type text. Each Emacs and host-name % window has a cursor. The cursor is a black rectangle when the window is active and white when not active.

Now you are ready to edit the file. With the mouse pointer in the large area of the window, start typing. Don't worry about making mistakes. We will fix them up in a few moments. Type three or four lines worth of text. Use the four arrow keys to move the cursor around and insert text. Make corrections by deleting the character to the left of the cursor by using the *Backspace* key. You can also use the left mouse button to position the cursor where the pointer is.

6.3. Notation for Emacs' Key Sequences

Emacs uses special sequences beginning with the **Control** and **Esc** keys. The notation **C-f** means to hold the **Control** key down and press the **f** key. The notation **M-f** means to press the **Esc** key once and then **f**. Do not hold the **Esc** key down as that repeats the key.

6.4. Exiting From Emacs

To exit Emacs, from the menu select **File – Exit Emacs** (or type **C-x C-c**) and Emacs will ask you if you need to save any files.

6.5. Saving the Contents of the File

What you have typed for the file `myfile` has not been saved. To save the file, select **File – Save (Current Buffer)** (or type **C-x C-s**). You should see a message appearing at the very bottom of the window saying Emacs ‘wrote’ the file. This small area is where Emacs displays messages or requests input. As you are developing a program or typing in a data file, you should save it every few minutes.

6.6. Reading A File

You can read (find or load) a file after you are in Emacs by selecting **File – Open File…** (or typing **C-x C-f**). In the

bottom small area, you should see find file: `~/` where you type in the pathname of the file. For now, you should type in `stories` after the `/` and a file with stories should be loaded in the window for editing (If the stories file is missing, you didn't copy it. Exit Emacs and copy the file into your directory as described in an earlier section). Since you will be experimenting and modifying your own copy of the file, feel free to modify it as you wish. The stories file is too large to fit on the screen all at once. Notice the black bar on the bottom, it says 'top'. This means you are at the top of the file. At other times by displaying a percentage, it indicates how far you are through the file. When you are at the end, it says 'bot' for bottom. If all the file can be displayed, then it says 'all.'

6.7. Moving the Cursor and Moving Text Within the Screen

To scroll forward and backward, use **Page Up** and **Page Down** respectively and notice that the percentage changes on the bar across the bottom. Below are some cursor commands to try on the stories file. Practice them all several times before going on.

Table 1. Cursor Movement and Screen Motion

M-f	Forward one word
M-b	Back one word
C-e	End of line
C-a	Beginning of line
M-<	Beginning of file
M->	End of file
C-v	Scroll forward a page
M-v	Scroll back a page

6.8. Cutting and Pasting a Region of Text

To move a block of text, you drag the cursor from the beginning of the region to the end of the region and cut the text with **Edit –> Cut** (or type **C-w**). Now you move the cursor to the place where you want and *paste* with **C-y**.

To copy or duplicate a region of text is similar except instead of cut we *copy* with **Edit –> Copy** (or type **M-w**). Move a whole story from one part of the file to another. Copy or duplicate a saying or two. The cut operation can be used to delete a region as well. Remove from the file any stories you don't like! Here is a summary of the killing and deleting operations to practice on the stories file.

Table 2. Killing and deleting

DEL	Delete character to left
M-DEL	Kill word to left
C-k	Kill from cursor to end of line
C-SPACE	Mark start of region
C-w	Cut region (Kill region)
M-w	Copy region
C-y	Paste region (Yank back last thing killed)
M-y	Yank back thing before last thing killed (do after a C-y). Repeat to retrieve older text killed.

Notice the distinction between ‘deleted’ and killed.’ Deleted text can not be ‘yanked’ back or retrieved while killed text can.

6.9. Recovering From Errors

You just typed an **Emacs** command and it didn't do as you wanted. Perhaps, you deleted a large, but the wrong, section of text!! While in **Emacs** you can undo it by selecting **Edit – Undo** (or type **C-_**). You can continue to use this option to slowly undo your other changes.

Another cause of frustration is when you type in a wrong **Emacs** command but haven't pressed the **Return** key yet. You have great dread that this unknown command will wreak havoc on your file. Just type **C-g** to cancel a partially completed command.

6.10. Using Auto-save's Backup Copy of File

If you do not save your file for several minutes, **Emacs** automatically saves the file (**auto-save** feature) in a file name the same as yours with a ‘#’ before and after. In case of disaster, e.g. the electrical power goes out, you can retrieve the latest auto-save copy by copying that file to your file. Inside **Emacs** type **M-x recover-file** then type in the name of your file you want to recover. Then you can save with **C-x C-s** to put the recovered text into the file itself.

6.11. Searching for Character Strings

Emacs has a character searching feature that is a joy! It is called an incremental search as it finds the next match from where the cursor is as you type in each character! Select **Edit –> Search –> Search**… (or type **C-s**) and slowly type a common word such as ‘**and**’ to see this in action.

To find further occurrences of the string, type **C-s** repeatedly. When you get to the end of the file, **C-s** will wrap around to the beginning and continue searching. To get out of search mode type **C-g**. You can always search for the last thing searched by hitting **C-s C-s**. Select –> **Search –> Search Backwards**… (or **C-r**) does the same thing except the search goes backwards in the file.

6.12. Splitting the Screen and Multiple Files

Many times you would like to look at two different portions of a file but they both are not displayed at the same time. **Emacs** lets you split the window into two parts horizontally by selecting **File –> Split Window** (or typing **C-x 2**). Now both parts can be scrolled and operated on. For example, you may cut and paste from one part to the other. You move between the split windows by clicking in either pane with the mouse or by typing **C-x o**. You remove all split windows other than the window with the cursor by selecting **File – Unsplit Windows** (or typing **C-x 1**).

A powerful technique is to use the split windows on two different files. Just move to the other part and read (find) a file, say `myfile`. Again, you can scroll the two files and cut and paste between them.

Here is a summary of the split and multiple window commands.

Table 3. Split Windows and Multiple Files

C-x 2	Split window in two horizontally
C-x 0	Delete this window
C-x 1	Delete all other windows
C-x o	Switch cursor to other window

6.13. On-line Help and On-line Emacs Tutorial

There is on-line help in **Emacs** which you will probably find a little advanced. You receive the help by **C-h**. If you want some more practice with the **Emacs** commands, there is a tutorial which you begin by selecting **Help –> Emacs Tutorial** (or by typing **C-h t**).

6.14. Summary of Some Emacs Commands

On the following page(s) is a summary of many of the **Emacs** commands mentioned in the previous sections.

C-f means hold **Control** key down and press **f**.

M-a means press **Esc** key then press **a**.

Table 4. Emacs Command Summary

Cursor movement:	
C-f	Forward one space.
C-b	Back one space.
C-n	Next line.
C-p	Previous line.
C-a	Beginning of line.
C-e	End of line.
M-f	Forward one word.
M-b	Back one word.
M-<	Beginning of file.
M->	End of file.
Screen Motion:	
C-v	Scroll forward a page.
M-v	Scroll back a page.
Error Recovery:	
C-g	Abort partially typed in command.
C-_	Undo an unwanted change.
Killing and deleting:	
DEL	Delete character to left.
M-DEL	Delete word to left.
C-k	Kill from cursor to end of line.
C-SPACE	Mark start of region.
C-w	Cut region (Kill region).
M-w	Copy region.
C-y	Paste region (Yank back last thing killed).
M-y	Yank back thing before last thing killed (do after a C-y). Repeat to retrieve old text killed.
Split Windows and Multiple Files:	
C-x 2	Split window in 2 horizontally.
C-x 0	Delete this window.

C-x 1	Delete all other windows.
C-x o	Switch cursor to other window
General Commands:	
C-x C-c	To quit Emacs.
C-h	For help.
C-x C-f	To find or read a file (used to create new ones too).
C-x C-s	To save a file.
C-x i	To insert contents of file.
C-x C-b	List buffers.
C-s	Incremental search for a string. C-s for next occurrence. ESC to stop search.
M-x repl s	To replace a string with another.
C-x C-g or (**M-x goto-line**)	To go to a line number.

7. Compiling & Running Programs

7.1. The C Language

Create a file with .c as suffix, e.g., mine.c and place in it the following C program.

```
/* this is a comment */
#include <stdio.h>    /* bring in I/O library */
main ( )
   {
     int i;
     i = 8;
     i = i + 1;
     printf("This is a C program. %d\n", i);
   }
```

To compile and link using GNU's C compiler, type:

```
[student@hostname ~]$ gcc mine.c -o mine
```

To run:

```
[student@hostname ~]$ mine
```

7.2. The C++ Language

Create a file with .cc as suffix, e.g., mine.cc and place in it the following C++ program.

```
// this is a comment
#include <iostream>    // bring in I/O library
using namespace std;
main ( )
   {
     int i;
     i = 8;
     i = i + 1;
     cout <<  "This is a C++ program. " << i << "\n";
   }
```

To compile and link using GNU's C++ compiler, type:

```
[student@hostname ~]$ g++ mine.cc -o mine
```

To run:

```
[student@hostname ~]$ mine
```

7.3. The Java Language

Create a file with .java as suffix, e.g., test.java and place in it the following Java program.

```
// this is a comment
class test {
    public static void main(String[] args) {
        int i;
        i = 8;
        i = i + 1;
        System.out.print("This is a Java program. ");
        System.out.print(i);
        System.out.println("");
    }
}
```

To compile using the Java compiler, type:

```
[student@hostname ~]$ javac test.java
```

To run:

```
[student@hostname ~]$ java test
```