BIF713

File and Directory Management

File System

- A **File System** is a structure used to organize programs and data on a computer's storage device

- Files are used to store various items. These concepts relates to both the **Linux** and **Windows** Operating Systems. Below are examples of some file types:

Regular files – Used to store text, images, executable programs, music, etc.

Device files – Used to represent hardware devices such as hard disks, terminals, printers, mouse, keyboard, etc… (stored in the **/dev** directory - for Linux)

Directory files – Used to store regular, device, and other directory files for better organization and quicker access

File System

You can use the **ls –l** command (or **dir** in Windows) to determine file information. For Example:

```
ls -l /dev/tty
crw-rw-rw-    1 root  root  5, 0 2003-03-14 08:07 /dev/tty

ls -l monday.txt w1.c
-rw-r--r--    1 murray.saul users 214 2006-01-23 14:20 monday.txt
-rw-r--r--    1 murray.saul users 248 2005-10-12 13:36 w1.c

ls –ld uli101
drwxr-xr-x    2 murray.saul users 4096 2006-01-17 16:43 uli101
```

In Linux, you can determine file type from looking at first character in detailed listing:

- indicates a regular file

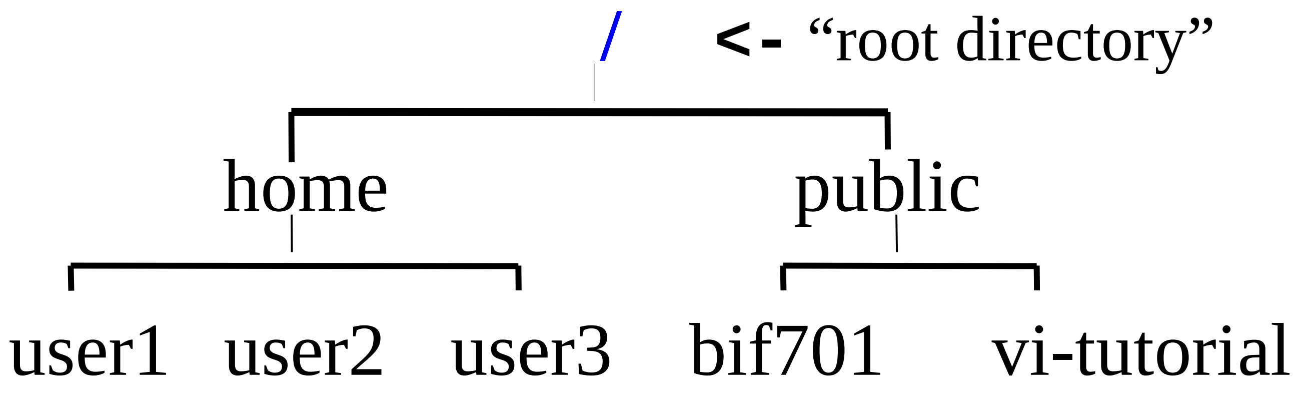
b or **c** indicates a device file

d indicates a directory file

Note: In MS Windows, the **dir** command provides information regarding type of file in the listing. For example <dir> represents a directory file....

Hierarchical File System

- In the Linux OS, the "root directory" **/** is the starting directory (in windows it starts with a disk drive letter like **c:\**). Other "child directories", "grandchild directories”, etc. are created originating from root dir.
- The hierarchical structure resembles an "upside-down tree“. There is actually a command called **tree** that can display a **tree diagram**!



Directory Structure in Linux

Directory Path	Description
/	Root directory (ancestor to all directories).
/home	Used to store users’ home directories.
/home/userid	User's actual home directory
/bin	Common system binaries (commands).
/usr/bin	Common utilities (commands) for users.
/usr/sbin	Common utilities for user administration.
/etc	General System administration files.
/var	Files that continually change (eg. log files)
/tmp, **/var/tmp**	Temporary files for programs
/dev	Device files (terminals, printers, etc

Directory Structure in Windows

Directory Path	Description
c:\ , a:\ , z:\	Root directory (for that disk drive).
c:\Users	Used to store various users' directories.
c:\Users\Username	User's actual home directory
c:\Windows	Common commands & programs.
c:\temp	Temporary files for programs
Username\Desktop	User's desktop.
Username\Documents	User's Documents (eg. Word, excel).
Username\Pictures	User's Pictures.

File Naming Rules

The following rules apply to naming regular files or directory files:

- Some Linux older file systems restrict filename size to 14 characters, most file systems allow for 255 characters (safest to select filename size of 14).
- Can use letters (upper & lower case), numbers, period , comma or underscore _ characters. Upper case is different than lower case.
- A period at beginning of filename hides file. MS Windows uses attributes to hide files and directories...
- Avoid spaces and other punctuation in filenames**. MS Windows has less restrictions than Linux in file naming rules...**

Pathnames

- A **pathname** is a listing of directories that will lead to a directory or a file.
- The concept of a pathname relates to every operating system including Unix, Linux, MS-DOS, MS-Windows, Apple-Macintosh, etc.! Newer versions of windows allows use of forward slash **/** as well as backslash \ when specifying file/directory pathnames...
- Examples:
  - Directory pathname:
    `/home/username/ics124/assignments`
  - File pathname:
    `/home/username/ops224/assignments/assn1.txt`

Absolute vs Relative Pathnames

Absolute Pathname

- A pathname that begins from root **/** (or in windows, for example **c:\**).
- The pathname begins with a slash
  eg. **/home/murray.saul/bif713**
  eg. **c:\Users\Userid\Desktop**

Relative Pathname

- A pathname that is "relative" to the location of the current or "working" directory.For example, if we are in our home directory, issuing the command `mkdir bif713` will create the bif713 directory from our home directory!

Relative Pathnames

Rules:

- A relative pathname does NOT begin with a slash. Many of these rules apply to <u>both</u> Linux and Windows operating systems.

- Following symbols can be used:

..	parent directory (up one directory level)
.	current directory

<u>WARNING</u>:
When using relative pathname, always make certain you know your present working directory!

Relative Pathnames

Examples:

- Change to another directory branch from parent directory: `cd ../ipc144`

- copy sample.c file from your professor's directory to your current directory:
  `cp /home/murray.saul/bif713/exam.txt  .`

  (Note: Copy command in Windows is: **copy**)

Relative-to-Home Pathnames

- You can specify a pathname as relative-to-home by using a tilde and slash at the start, e.g.,

  ~/bif713/notes.html

- The tilde ~ is replaced by your home directory (typically /home/*your.account*/) to make the pathname absolute.

- You can immediately place a username after the tilde to represent another user's home directory. For example:
  ~chris.tyler = /home/chris.tyler
  but ~ **=** **/home/your_home_dir**

- **Relative-to-Home pathnames <u>not</u> available in Windows.**

Which Type of Pathname to Use?

So far, we have been given many different types of pathnames that we can use for regular files and directories:

- **Absolute pathname** (starts with /)
- **Relative pathname** (doesn’t start with /)
- **Relative-to-home pathname** (starts with ~)

You can decide which pathname type to use to make it more convenient (eg relative – less typing or absolute – you don’t know what directory you are currently located in…)

Making Directories

Building directories is similar in approach to building a house

- Begins from a foundation (eg home directory).
- Need to build in proper order (add on addition to house in right location). Use a logical scheme.
- When building directories from different locations, must provide proper absolute or relative pathname!!

Planning Directories

Good directory organization requires planning:

- Group information together logically.
- Plan for the future: use dated directories where appropriate (**~/semester/2009**, or **~/semester/2010**)
- Too few directories = excessive number of files in each; too many directories = long pathnames.

Where do we want to build directory?

- We want to build a directory called **tmp** that branches-off of your home directory
- Verify which directory you are located (either look at directory from command prompt or issue the command **pwd**). The pwd command not available in Windows.
- Type **mkdir tmp** at the Unix prompt, followed by ENTER
- Always verify that directory has been created (e.g. use **ls** or **ls -ld** command). In Windows, use the **dir** command.

Creating Parent Directories

To create directory paths with parent directories that do not exist you can use the command

mkdir -p *pathname*

eg. mkdir -p mur/dir1

(This would create the parent directory mur and then the child directory dir1. The -p means "create all the directories in the Path").

Refer to **mkdir /?** for options in Windows...

Removing Directories

Removing directories is reverse order of building directories

- Issue command: **rmdir *directory-pathname***
- **rmdir** cannot remove directories containing files or other subdirectories.
- **rmdir** cannot remove directories that are anyone's current directory.
- Need to step back to at least parent directory to remove an empty directory.

Removing Sub-trees

- To remove a sub-tree (a directory and all of its contents including sub-directories) use **rm -r directory** (or **rm -R *directory***). The command **rm –rf** removes both subdirectories and files contained in a directory-path.
- The command **rm** is not available in Windows (See available option with **rmdir** command)
- **Caution! rm -r** can erase large numbers of files very quickly. Use with extreme care!

Additional File Management Commands

Operation	Linux	MS Windows
Make Directory	**mkdir**	**mkdir**
Change to Directory	**cd**	**cd**
remove Directory	**rm -r, rmdir**	**rmdir**
Delete File	**rm**	**delete**
Copy File	**cp**	**copy**
Move File	**mv**	**move**
Rename File	**mv**	**ren**
View text file	**cat, more, less**	**type**
List Contents	**ls, ls -l**	**dir**
View Current Directory	**pwd**	**N/A**

Additional Resources

- This slide-show provides the minimum amount of Linux and Windows concepts that could appear on a test or final exam.
- Here are some Related-Links for Interest Only:

General Definition of an OS File System:

http://en.wikipedia.org/wiki/File_system