ULI101

Week 02

Week Overview

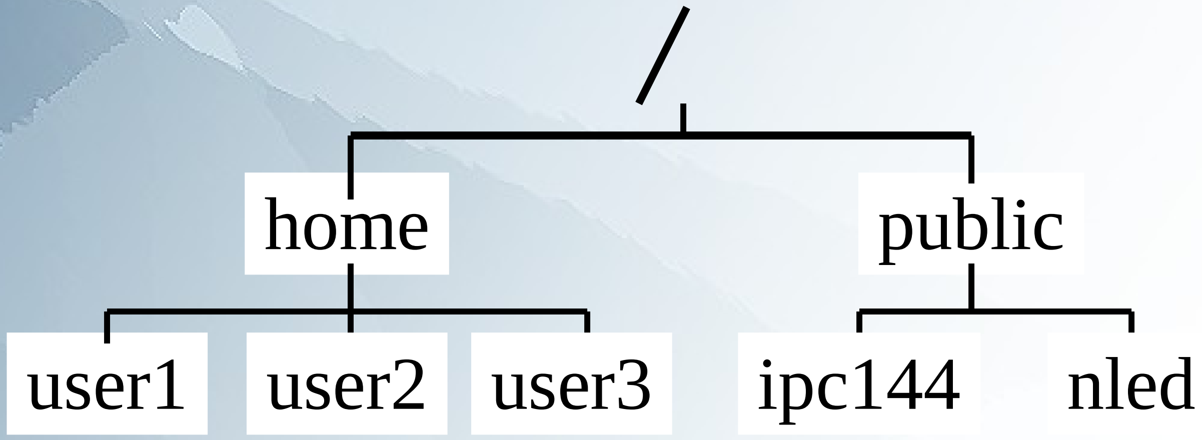
- Unix file system
- File types and file naming
- Basic file system commands: pwd,cd,ls,mkdir,rmdir,mv,cp,rm
- man pages
- Text editing
- Common file utilities: cat,more,less,touch,file,find

Unix File System

- The Unix/Linux file system is hierarchical, similar to other operating systems today
  - Files are organized in directories
  - Directories may contain sub-directories
- What is different (from Windows anyway) is that there are no drive letters (such as C:, or D:)
  - All files and directories appear under a single root, even if multiple storage devices are used
- Learning command-line navigation of the file system is essential for efficient system usage

Hierarchical File System

- In the Linux (Unix) OS, the "root directory" / is the starting directory, and other "child directories", "grandchild directories", etc. are created
- The hierarchical structure resembles an "upside-down tree". There is actually a command called tree that can display a "tree diagram"!



Typical Unix/Linux Directories

• /	Root directory (ancestor to all directories).
• /home	Used to store users’ home directories.
• /bin	Common system binaries (commands).
• /usr/bin	Common utilities (commands) for users.
• /usr/sbin	Common utilities for user administration.
• /etc	General System Admin. Files (eg passwd)
• /var	Dynamic files (log files)
• /tmp, /var/tmp	Temporary files for programs
• /dev	Device files (terminals, printers, etc.)

Home directory

- Every user when receiving an account has a “home” directory created
- This is where you keep your personal files
- ~ represents your home
  - You can use the ~ symbol in the pathnames
- A cd command without any parameter will get you directly to your home directory
- Remember to keep your files private

Types of Files

- On a Unix/Linux file system a “file” can be anything
  - To an average computer user a file is a text document, video, music, photo etc.
- A directory is really an index file, containing references to file locations on the physical disc and other related information
- Devices such as the terminal or printer are also files
  - You will learn more details about this later in the course
- Any file (or directory) name starting with a period is considered to be a hidden file

Types of Files

- You can use the ls –l command to determine the type of file.

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

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

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: you can use the **–d** option with the **detailed listing command** to get information for just the directoryfile. eg. **ls -ld /home/myacct**

Hidden Files

- A file is hidden if its name starts with a .
  For example: .profile
- ls –a will show all files including hidden
- . and .. directories are hidden
  - ls –A will show "Almost" all files – not including . and ..
- Why make files hidden?
  - To clean up directories
  - To hide backups
  - To protect important files from accidental deletion
- Remember: directories are really files, you can hide them as well

Working With The File System

- Be very careful when working with files on the command line, as there is no undo command or a Trash/Recycling Bin
  - A single command can wipe out your entire account
  - Changes are instant and permanent
- Make backups of important files, preferably outside of your account – USB storage is a good option
- You will learn later additional ways to control file access through file permissions which will help you prevent accidental file damage or deletion

Basic Commands

pwd

- Used to display the user's present working directory. A user may need to where they are located on the computer system in order to build directories, copy files, etc…

cd *directorypath*

- Used to change to a directory. Entering the cd command without the directory path will change to the user's home directory.

Basic Commands

ls

- Used to display the contents of a directory (eg. regular files or sub-directories). By default, the ls command displays non-hidden filenames only.
- The following are common options associated with the ls command:
- **-a** short display of hidden & non-hidden files
- **-l** detailed display of files (excl. hidden files)
- **-F** displays / after directory, * after executable file
- Options can be combined, for example: ls -la (or ls -l -a)

Basic Commands

mkdir *directorypath*

- Used to create a subdirectory with a directory. Multiple arguments can be used to create many subdirectories. The option –p allows for parent directories to be created.

rmdir *directorypath*

- Used to remove only empty directories (i.e. directories that contain no subdirectories or regular files). A user cannot remove a directory from within the directory location itself.

Basic Commands

mv *sourcepath destinationpath*

- Used to move a file from one location to another and/or rename the file. The mv command can be used to move directories as well as files. The –i option asks for confirmation if the destination filename already exists.

cp *sourcepath destinationpath*

- Used to copy a file from one location to another. The cp command can be used to backup important files.
- The –i option asks for confirmation if the destination filename already exists.
- The -r option allows copying of directories and their contents

Basic Commands

rm *filepath*

- Used to remove a regular file.

rm -r *filepath*

- Used to recursively remove a directory and it's contents. Recursive means to descend to lower levels, which in this case, indicates that subdirectories and it contents are also removed.

  Note: it is a good idea to include the –i option to confirm deletion of subdirectories and its contents!

Basic Commands

cat *filepath*

- To join files (i.e. to con**cat**enate files). For example, **cat file1 file2 file3** will display the contents of file1 and file2 and file3 on the screen at the same time.
- To display the contents of small files (files longer than the screen will scroll to the end). For example, issuing the command **cat .bash_profile** in your home directory would display the contents of your setup file.

more *filepath*

- Used to display the contents of large regular files one screen at a time. The user can navigate throughout the file by pressing keys such as:

spacebar	`Move to next screen`
b	`Move to previous screen`
enter	`Move to next line`
/car	`Search for pattern "car"`
q	`Exit to shell`

less *filepath*

- Works like **more** command, but contains more navigation features.

Basic Commands

touch *path*

- Used to update the date and time of existing files.
- The touch command is also used to create empty files. You will be using the touch command to create empty files when you practice the file management on-line tutorial

file *path*

- Determines a file type
- Useful when a particular file has no file extension or the extension is unknown/incorrect

The find Command

- The find command allows searching for files by file name, size as well as file attributes recursively throughout the file system
  - An optional action can be performed on matches
- Examples:
  - Search for a file named bob:
    - `find / -name bob`
  - Delete empty files belonging to user alice:
    - `find / -user alice -empty -delete`
  - Find all files modified less than 5 minutes ago:
    - `find / -mmin -5`
  - Find large files:
    - `find . -size +100M`

File Naming

- Unix/Linux is case sensitive!
- Adopt a consistent file naming scheme – this will help you find your files later
- Make your file and directory names meaningful
- Avoid non alphanumeric characters, as they have a special meaning to the system and will make your work more difficult
- Avoid using spaces in file names – consider periods, hyphens and underscores instead
- Feel free to use file name extensions to describe the file purpose

Getting Help with Commands

A comprehensive online manual for common UNIX/Linux commands exists on your server

The online manual is a command called man

Command Structure:

```
man [options] command
```

Options:

-k provides short (one-line) explanation relating to the commands matching the character string. This can be used if user doesn't know name of command.
eg. `man -k calendar`

Text Editing

- Editing text files is an everyday activity for both users as well as administrators on a Unix and Linux system
  - System configuration files
  - Scripts and programs
  - Documentation
  - Web pages
  - ...
- As the GUI may not always be available, knowing command-line text editors is a very valuable skill
- Please note that although both Unix/Linux and Windows use ASCII to encode text files, there are small differences that may cause problems (particularly with scripts) when copying files between different systems
  - If needed, use the unix2dos and dos2unix utilities to convert between the two systems

Text Editing

- A specific system may have many editors available and as you work with one for a while you will probably pick a favourite one
- A traditional fall-back is the vi editor, as it is most likely to be present on all Unix-like systems, especially when installed with a minimum software complement
  - Consider knowing vi as one of the “badges” of a competent Unix/Linux user
- Vi has a relatively steep learning curve and is not user friendly, but it offers nice advanced features which will be introduced later in the course

vi (Visual) Editor

vi is a powerful, interactive, visually-oriented text editor

Features:

- Efficient editing by using keystrokes instead of mouse.
- Use of regular expressions
- Possibility to recover files after accidental loss of connection
- Features for programmers (eg. line numbering, auto-indent, etc…)

Although you may prefer to use other editors (such as nano or nled), knowing vi is very useful, as this is one editor that is present on all Unix-like systems

Starting vi Session

There are two ways to start an editing session with vi:

- Enter **vi filename** (recommended since filename has already been assigned and changes will be saved to filename by entering **ZZ** while in vi).
- Enter **vi** (filename is not assigned, therefore user has to type **<ESC> :w filename <ENTER>** in order to save file.

Modes

- There are three operational modes while using the vi editor:
  - Command Mode (default mode when starting)
    - User presses letter for a command – for example, input text, delete text, append text, etc.
  - Input Mode
    - Input Mode allows user to enter or edit text. Press ESC to return to command mode.
  - Last-line Mode
    - Pressing colon ":" opens a prompt to enter letter or word commands. More complex operations such as search and replace can be performed.

Moving in Command Mode

- You can move around to text in the screen by using the following keys:
  - **h (left), j (down), k (up), and l (right).**
  - **w (right one word to special character),**
  - **W (right one word including special characters)**
  - **b (left one word to special character),**
  - **B (left one word including special characters)**
  - **0 (zero) (beginning of line), $ (end of line)**
- You may be able to move around by using the arrow keys (depends on version of vi).
- For more advanced editing, you can return to Command Mode and use appropriate editing commands.

Getting into Input Mode

While in command mode, you can issue the following commands to input text:

i – insert to left of cursor
o – insert line below current line
a - append to right of cursor
r - replace character

I – insert at beginning of line
0 – insert line above current line
A - append at end of current line
R – overwrite text

Common Editing Commands

x – Delete single character

d – Delete

c – Change

y – Yank (copy)

p – paste below cursor, P – paste above cursor

u – undo previous edit

. – repeat previous edit

Editing commands can be preceded with a number, for example:

3x = delete the next three characters

2u = undo the last two edits

12dd = delete 12 lines

Searching

- Search for text (in command mode)

– /pattern	Search forward for pattern
– ?pattern	Search backwards for patter
– n	Display next match
– //	Repeat previous search

Saving Edited File

- Work performed during vi session is stored in a Work Buffer (temporary storage) until the user saves their work.
- When saving, changes in the work buffer are placed in a new file if creating a new file, or changes in work buffer modify existing (previously created) file.
- To save your vi session, you must make sure you are in command mode by pressing <ESC>
- To save your changes and exit, enter ZZ (i.e. two capital z's). You could perform the same operation in last line mode by :x
- You can also save without exiting by entering :w

Saving Edited File

- To save your vi session, you must make sure you are in command mode by pressing <ESC>
- To save your changes and exit, enter ZZ (i.e. two capital z's). You could perform the same operation in last line mode by :x
- You can also save without exiting by entering :w

Aborting Editing Session

- If you make a mistake in your editing session (that undo cannot solve), you can abort your session without modifying the contents of your file (dump the work buffer)

- To abort the current editing session, press `<ESC> :q! <ENTER>`