CS 35L

Linux Basics

Slide Set 1.2

Lab 7 – Spring 2020

What is ‘ls’?

- What happens when we type ‘ls’ in the shell?
- The shell *executes* programs (executables)
- We can pass the absolute path of the executable file:
  - $: /bin/ls
- We can pass the name of the executable and let shell search for ‘ls’ at certain specific locations.

Environment Variables

- Environment: Area that contains the variables that the **shell** uses to define system properties.
  - Environment Variables: Variables that are defined for the current shell and are inherited by any child shells or processes (processes have access to these variables).
  - Shell Variables: Variables that affect the current shell.

Creating Shell Variables

- To create a shell variable (no spaces between equal sign):
  - [user@host]$: a=1
  - [user@host]$: echo $a → 1

  - [user@host]$: a=joe
  - [user@host]$: echo $a → joe

  - [user@host]$: a="${a} Johnson"
  - [user@host]$: echo $a → Joe Johnson

- Note, we use curly braces {} when "expanding" a variable inside a string.

The PATH Variable

- Is an environment variable that specifies which directories the *shell* should search in for executables.

- To get the value of PATH:
  - [user@host]$: echo $PATH

    /path/to/directory/1:/path/to/directory/2:(and so on)

- When we type ‘ls’, the shell searches for the ‘ls’ executable in **/path/to/directory/1** then **/path/to/directory/2** and so on…

- If the executable is not found in any directory in PATH, the shell returns an error.

Editing the PATH Variable

- We can prepend directories to the PATH variable (for the current shell session):
  - [user@host]$: export PATH="/usr/local/cs/bin:$PATH"
  - Here, we prepended **/usr/local/cs/bin** to the old value of PATH.
- To prepended a directory permanently:
  - Add **export PATH="/usr/local/cs/bin:$PATH"** to your **~/.bash_profile** or the **~/.profile** file.
  - Create the file if it doesn't exist.

The Export Command

- **export** is a command that exports variables of the current shell to the environments of all child processes.
- The *shell* is a process, and when we call ‘ls’, a child ‘ls’ process is spawned by the *shell*.
- When we assign a variable **a=1** in the current *shell*, the variable **a** is local in the *shell*’s memory space and thus will not exist in any spawned processes.
- Using **export**, we can pass the variable **a** to any spawned processes.

File System vs Directory Structure

- A file system is a mechanism that organizes physical data on a disk and provides an interface for their manipulation.
  - In Unix file systems, every file has metadata that is stored in an entity called an **inode**.
  - Every file (and thus directory) has a unique **inode number**.
  - Metadata includes: owner, permissions, size, number of links, …
  - All inodes are found in the **inode table**. A lightweight version is stored in memory while the full table is stored on disk.
  - To view inode number: **ls -i**

- A directory structure is a mapping from file representations (file names) to inodes.
  - More than one entry in the directory structure can map to a single inode (many-to-one mapping).
  - The files and directories we see in the shell is the directory structure.

Directory Entry

- A Unix directory is a list of association structures (mappings of file names to inode numbers).

- For example, in the EXT2 file system, a directory is a list of entries of the following format:
  - **Inode** – the inode number for this directory entry
  - **Name Length** – the length of this directory entry in bytes
  - **Name** – the name of the directory entry

- Example:
  - Inode: 3047639937
  - Name Length: 15
  - Name: /home/joe/file1

- Reference: http://www.tldp.org/LDP/tlk/fs/filesystem.html#tth_sEc9.1.4

Soft/Symbolic Links vs. Hard Links

- Soft/Symbolic Link files: These are pointers to files.
  - Similar to “shortcuts”
  - If original file is deleted the soft link will no longer work properly (dangling link)
  - It can link to a directory
  - It links across file systems

$ ln –s [original filename] [link name]

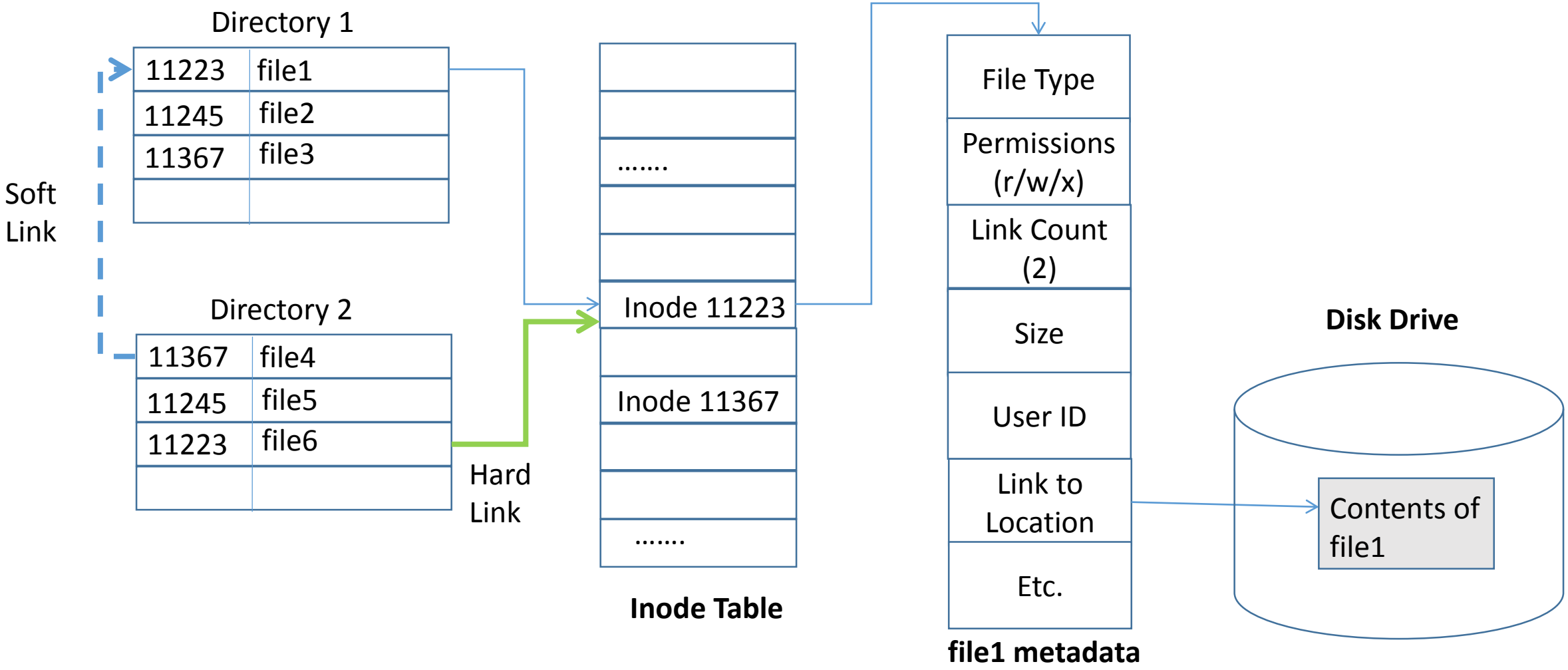
$ ln –s ~/myfiles/file1 file4
Soft Link: file4 ---> ~/myfiles/file1

- Hard Link
  - Points to the actual data
  - If original file is removed the link will still point to the actual data
  - Cannot link to a directory
  - Does not work across files systems

$ ln [original filename] [link name]

$ln ~/myfiles/file1 file6
Hard Link: file6 ---> actual file1 data

Inodes – Hard Links – Soft/Symbolic Links



More on Links

- Link count of an inode tracks the number of directory entries mapped to the inode.
- Symbolic links do not increase the link count and have their own inode.
- Hard links have less indirection than soft links when trying to locate a file/inode, thus are faster.
- Cannot hard link across file systems:
  - If **/some/directory** is in a different file system than **/another/directory**, then we cannot create a hard link between **/some/directory/proposed_link** to **/another/directory/source_file**

More on Links

- Unix file systems have to be a directed acyclic graph (DAG), so directories cannot be hard linked.

- For example:
  - If **/directory1/** had a hard link to itself as a subdirectory: **/directory1/directory2 → /directory1/**
  - Then **cd** to **directory2** will take us to **directory1**, which has **directory2** as a subdirectory, which **cd** to **directory2** will take us to **directory1** and so on…

- Cycles are bad!

More on Links

- Since a symbolic link only stores the path to a file (a string), it can link across file systems.

- Symbolic links can create cycles (not a DAG).

Directory Structure vs File System

- Directory Structure
  - Only one directory structure in an operating system with a single root /
  - Two entries in the directory structure can refer to the same file in the file system.
  - Symbolic links can create cycles, so the graph structure is not acyclic.
- File System
  - There can be multiple file systems. E.g. each disk drive can have a separate file system, maybe even with different partition formatting.
  - Every entity has a unique inode.
  - Forms a directed acyclic graph.

Retrieving files from the web

- wget [URL]

- DESCRIPTION: GNU Wget is a free utility for non-interactive download of files from the Web. It supports HTTP, HTTPS, and FTP protocols, as well as retrieval through HTTP proxies.

Emacs – The Display Editor

- “Advanced”: Does much more than simple insertion/deletion of text.
- “Self-documenting”: Lots of help commands.
  - Tutorial (C-h t) ← Very Helpful!
  - Manual (C-h r)
- “Customizable”: Users can customize font, color, … (without programming)
- “Extensible”: Redefine and create new commands (programming in LISP)
- “Real-time”: Edits are displayed on screen as they occur

Emacs – The Display Editor

- Navigating within a file:
  - Move up/down/left/right: C-p, C-n, C-b, C-f (can also use arrow keys)
  - Move to the beginning/end of a line: C-a, C-e
  - Move to the first/last line of the text: M-<, M-> (use shift for <, >)
  - Move to a specific line number: M-g g [line number]
- Search and replace within a file:
  - C-M s: Regular expression search forward
  - C-M r: Regular expression search backward
  - M-%: search and replace (usage: M-% [From-String] Enter [To-String] (y/n)…)
- Deleting (killing or cutting)
  - Delete the character before cursor: Backspace
  - Kill from cursor position to end of line: C-k
  - Kill to the end of the current sentence: M-k
  - Kill up to and including a specific character in a word: M-z (a.k.a zap-to-kill)

Note:
C: Ctrl
M: Alt or Meta
SPC: Space

Emacs – The Display Editor

- Copy/cut and paste in a file:
  - Begin: C-SPC (to set mark)
  - Move the cursor up or down to highlight the desired text
  - End: C-w (cut), M-w (copy), C-y (paste)
- Undo: C-/ or C-x u
- Cancel a partially written command: C-g

Emacs – Buffers

- A buffer is an object that holds text you can edit.
- Multiple buffers can be opened at the same time.

- Add a new file to a buffer: C-x C-f (also used to create a new file)
- Save current buffer to file: C-x C-s
- Save current buffer to a specified file name: C-x C-w [filename]
- List all buffers: C-x C-b
- Change current buffer: C-x b
- Display only one buffer (the one you are currently editing): C-x 1
- Switch to other buffer: C-x o

Emacs – Dired

- Dired: Shorthand for Directory Edit
- Creates an Emacs buffer listing the contents of a directory

- Can remove, rename, copy files, …
- Can navigate the file system (switching to different directories, …)

- **dired: C-x d**
- “q” to close the dired buffer

Emacs – Other important commands

- Emacs as a shell:
  - M-! <command>
  - M-x shell (interactive shell)

- Emacs as IDE:
  - M-x compile, then specify command to compile
  - (Tip for later HW: gcc hello.c –o hello)
  - (Run the executable by running the shell command “./hello”)

- Running LISP code:
  - M-x emacs-lisp-mode

Emacs Reference Card

- https://www.gnu.org/software/emacs/refcards/pdf/refcard.pdf

Org mode

- Org mode for:
  - Keeping notes
  - Maintaining TODO lists
  - Planning projects
  - Authoring documents with a fast and effective plain-text system

- http://pragmaticemacs.com/emacs/org-mode-basics-structuring-your-notes/

Recording Key Strokes in Emacs

https://www.thegeekstuff.com/2010/07/emacs-macro-tutorial-how-to-record-and-play/