CIS 18A
Introduction to Linux / Unix

Regular Files

De Anza College
Instructor: Clare Nguyen

Files

- Data on computers are stored in units called files.
- Files are identified by their name and their location in the system.
- Filename requirements for Linux:
  - can be any combination of text characters: letters, numbers, and some punctuation marks
  - case sensitive: uppercase and lowercase letters are different.
  - *cannot* contain the following characters:
    & ; | * + ? ~ ! $ ^ # / \ ' " ` [ ] ( ) { } < > and space
- Filename recommendations:
  - make filename descriptive, use mostly letters and numbers.
  - can be multiple words, separate words by _ or –
  - don't need extensions. Use filename extension if it helps make the filename descriptive.
  - most filenames should not start with ". " A period at the front of the filename indicates that it's a *hidden* file, which means the file will not show up when you do a default listing of files.

cat

- cat: (for con**cat**enate) displays on screen the content of a file.
- Common format: cat filename
- cat can also accept a file list made of multiple filenames, separated by space. If given a file list, cat will list the content of one file after another, in the order given on the command line.
  Example: cat fileA fileB fileC
- cat can also accept no argument. If there is no argument, cat will echo what you type on screen and stay in display mode. To get back to the shell prompt, use control-d

more and less

- more: displays to screen the content of a file, page by page.
- less: displays to screen the content of a file, page by page.
- less is newer than more. It is more efficient to use less when you are working with a large file.
- Common format: more filename or less filename
- more / less will display the first page of the file
  - To display the next page: space
  - To display the previous page: control-b or b
  - To quit: q
- more / less can also accept a file list. The files in the list will be displayed one by one, in the order on the command line.

ls (1 of 2)

- ls: (for **l**i**s**t) list the filename of all files in the directory.
- Common format: ls
  All non-hidden filenames in the current directory are displayed in alphabetical order.
- Common format: ls filename
  - If the file exists, the filename is echoed back on screen. If the file doesn't exist, an error message is displayed. This is a quick way to check whether a file exists or not.
  - ls will also accept a file list. The filenames in the list that exist will be listed, and an error message will appear for each filename that doesn't exist.
- Common format: ls –a
  (option a is for **a**ll) Shows all filenames in the directory, including hidden filenames.

ls (2 of 2)

- Common format: ls -l
  - (l for **l**ong) long listing of filenames. Each file and its attributes are listed on a separate line, in column format.
  - The order of the attributes are:
    - Mode: shows file type and access rights
    - Number of hard links
    - Owner ID
    - Group ID of owner
    - Size: in bytes
    - Last access time / date
    - Filename
- The arguments and options can be combined, such as:
  ls -l fileA or ls -al

touch

- Common format: touch filename
  - If filename doesn't exist, then a new, empty file with the given name is created.
  - If filename exists, the access time of the file will be updated to the current time.
- touch can also accept a file list. Each filename in the list will either be created or have its access time updated.
- touch is useful to quickly create new files.

cp

- cp: (for **co**p**y**) will copy the content of a source file to the destination file. The source file still exists after copying.
- Common format: cp source_file destination_file
  - The 2 arguments are required in the order shown.
  - If the source_file doesn't exist, cp will send to screen an error message.
  - If the destination_file doesn't exist, cp will create a new destination file which is a copy of the source_file.
  - If the destination_file exists, its content will be overwritten by the source_file.
- To have the system ask for confirmation before overwriting an existing file: cp -i source_file destination_file
  - the option i is for **i**nteractive.
  - answer y for overwriting, n for not overwriting. When not overwriting, there is no copying done.

mv

- mv: (for **m**o**v**e) will move the content of a source file to the destination file, and then *delete* the source file. This effectively renames the source file to the destination filename.
- Common format: mv source_file destination_file
  - The 2 arguments are required in the order shown.
  - If the source_file doesn't exist, mv will send to screen an error message.
  - If the destination_file doesn't exist, mv will create a new destination file.
  - If the destination_file exists, its content will be overwritten by the source_file.
- To have the system ask for confirmation before overwriting an existing file: mv -i source_file destination_file
  - the option i is for **i**nteractive.
  - answer y for overwriting, n for not overwriting the existing file.

rm

- rm: (for **r**e**m**ove) will delete a file.
- Common format: rm filename

  where filename is required
- rm will also accept a file list, where each of the file in the list will be deleted.
- To have the system ask for confirmation before deleting the file: rm –i filename
  - the i option is for **i**nteractive.
  - answer y for delete, n for not delete.

Wildcards or Filename Expansion (1 of 3)

- Some of the commands working with files accept a file list.
- One way to enter a file list is by typing each filename individually, separated by space. An easier way is by using wildcards.
- Wildcard characters are used in a filename, *in place of* or *in addition to* the characters in the filename.
- Each wildcard in a filename is interpreted by the shell to match a set of characters, rather than one single character. Thus the shell *expands* the wildcard character in a given filename to match any number of existing filenames.
- Wildcard characters:

  ? matches any one character

  * matches 0 or more of any character

  [character set] matches any one character in the character set

Wildcards or Filename Expansion (2 of 3)

- Examples of using ?
  - The name lab? will match any filename that has lab and exactly 1 more character.

    For example, it can match the actual filenames lab1, labA, or labs, but will not match lab12 (2 characters after lab), lab (no character after lab), or laboratory (too many characters after lab).
  - The name my???file will match any filename that starts with my, followed by any 3 characters, and ending with file.
    - The ? can be used one after another to specify a specific number of characters in the filename.
- Examples of using *
  - The name lab* will match any filename that has lab and 0 or any number of characters after. It can match lab, labA, lab12345, lab_report, laboratory.
  - The name my*file will match any filename that starts with my and ends with file, regardless of any number of characters in between. It will match names from myfile to my_extremely_long_and_obnoxiously_named_file
    - The * is *not* used one after another since one * is expanded to match as many characters as possible already.

Wildcards or Filename Expansion (3 of 3)

- Examples of using [character set]
  - The characters in between the [ ] means that there is a match if the filename contains *one* of the characters in the set.
  - The wildcard lab[123] will match the filename that has lab followed by the number 1 or 2 or 3. It can match the actual filenames lab1, lab2, or lab3 but will not match lab123 or lab12.
  - The characters in the set can be described with a short hand notation if they are part of a set of alphanumeric characters: [a-z] matches one lowercase letter of any kind, [A-F] matches one uppercase letter between A and F, [2-7] matches one digit that can be 2,3,4,5,6,or 7.
  - The name my[23][xy]file will match the filenames my2xfile, my2yfile, my3xfile, or my3yfile only.
    - The [character set] can be used one after another to specify different groups of allowable characters.
- Each of the 3 wildcards can be used separately or together in the filename. They can also be used by themselves or with specific text characters in the filename.

Next stop: Text Editing