CS100: Software Tools & Technologies Lab I

Linux Commands

Vishwesh Jatala

Assistant Professor

Department of EECS

Indian Institute of Technology Bhilai

vishwesh@iitbhilai.ac.in

2022-23 W

Working with Files: Reading

- Often we only want to see what is in a file without opening it for editing.
- cat <filename>
  - Prints the contents of the file to the terminal window
- more <filename>
  - allows you to scroll through the file
- less <filename>
  - Lets you scroll up and down by pages or lines

Working with Files: Reading

- Reading beginning of a file (maybe read a header) or the end of a file (see the last few lines of a log).
- head -[numlines] <filename>
  - Prints the first numlines of the file
- tail -[numlines] <filename>
  - Prints the last numlines of the file
- Example: tail /var/log/Xorg.0.log
  - Prints the last ten lines of the log file.

Working with Files: Editing (VIM Editor)

- Vim is a powerful lightweight text editor.
- The name "Vim" is an acronym for "Vi IMproved"
  - vi is an older text editor
- Vim allows you to perform text editing tasks much faster than most other text editors!
  - Though it does have a learning curve

Why VIM Editor?

- Allows you to performs tasks quickly is because it works in modes.
- Without modes:
  - menus (with a mouse or keyboard), or
  - use complex/long command shortcut keys involving the control key (ctrl)
  - the alt key (alt)
- Vim uses modes to speed up

Do all these with just keyboard stikes!

VIM Modes

- **Editing (normal) mode:**
  - Launching pad to issue commands or go into other modes
  - Allows you to view the text but not edit it
  - Vim starts in normal mode
  - You can jump to normal mode by pressing the Escape (Esc) key on your keyboard

VIM Modes

- **Insert mode:**
  - Used to type text into the buffer (file)
  - This probably what you're used to from your text editor
  - You get to the insert mode by pressing the i key on your keyboard
- **Visual mode:**
  - Used to highlight text and perform operations on selected text
  - You get to visual mode from normal mode by pressing the v key on your keyboard

VIM Commands (Basic)

- **Not possible to teach all (Explore!)**

- Entering normal mode
  - Press <ESC>

- Entering insert mode
  - <i>

- Entering visual mode
  - <v>

VIM Commands (Basic)

- Save file
  - :w
- Exit
  - :q
- Quit without saving
  - :q!

File Permissions

- Linux was designed to allow multiple people to use the same machine at once.
- How about security?
- Access to files depends on permissions

File Permissions

- Each file is assigned to a single user and a single group (usually written user:group).
- For example Alice's files belong to alice:users, and roots files belong to root:root.
- Needs root privilege to change file ownership — a regular user can't take ownership of someone else's files and can't pass ownership of their files to another user or a group they don't belong to.
- To see what groups you belong to type groups.

File Permissions

- We can use ls -l to tell us about ownership and permissions of files
- ls -l - lists files and directories in the long format
- Example:
  - -rw-r--r-- 1 myuser mygroup 3775 2009-08-17 15:52 index.html

Permissions Format

- Example:
  - -rw-r--r-- 1 myuser mygroup 3775 2009-08-17 15:52 index.html
- rwxrwxrwx
  - User's Permissions
  - Group's Permissions
  - Other's permissions
- R = Read, W = Write, X = Execute
- Directory Permissions begin with a d instead of a -
- What permissions would -rw-rw-r-- mean?

Changing Permissions

- chmod <mode> <file>
- Mode: [0-7][0-7][0-7] (for user, group, others)
- Think of r, w, and x as binary variables:
  - 1 ON
  - 0 OFF

$$r \times 2^2 + w \times 2^1 + x \times 2^0$$

- **Examples:**
- chmod 755 : rwxr-xr-x
- chmod 600 : rw-------
- chmod 777 : rwxrwxrwx

Changing Permissions Recursively

- Most commands have a recursive option. This is used to act on every file in every subdirectory of the target
- Usually -R option

Special Characters

- Special characters
  - **$ * < > & ? { } [ ]**
- The shell interprets them in a special way unless we escape (\$) or place them in quotes "$".
- When we first invoke a command, the shell first translates it from a string of characters to a Linux command that it understands.

Special Characters

- * matches any string, including the null string (i.e. 0 or more characters).

Input	Matched	Not Matched
`Lec*`	`Lecture1.pdf Lec.avi`	`ALecBaldwin/`
`L*ure*`	`Lecture2.pdf Lectures/`	`sure.txt`
`*.tex`	`Lecture1.tex Presentation.tex`	`tex/`

Special Characters

- ? matches a single character

Input	Matched	Not Matched
`Lecture?.pdf`	`Lecture1.pdf Lecture2.pdf`	`Lecture11.pdf`
`ca?`	`cat can cap`	`ca cake`

Special Characters

- [...] matches any character inside the square brackets
  - Use a dash to indicate a range of characters
  - Can put commas between characters/ranges

Input	Matched	Not Matched
`[SL]ec*`	`Lecture Section`	`Vector.tex`
`Day[1-4].pdf`	`Day1.pdf Day2.pdf`	`Day5.pdf`
`[A-Z,a-z][0-9].mp3`	`A9.mp3 z4.mp3`	`Bz2.mp3 9a.mp3`

Special Characters

- Brace Expansion: {...,...} matches any phrase inside the comma-separated brackets

Input	Matched
`{Hello,Goodbye}\ World`	`Hello World Goodbye World`

Special Characters

- Any combination is also fine!

Input	Matched	Not Matched
`*i[a-z]e*`	`gift_ideas profile.doc`	`DriVer.exe`
`[bf][ao][ro].mp?`	`bar.mp3 foo.mpg`	`foo.mpeg`

References

- Miscellaneous resources from internet
- Lecture notes from https://www.cs.cornell.edu/courses/cs2043/2014sp/

Thank you!