LECTURE NOTES IN CIS300

YUZHE TANG

SPRING, 2018

SECTION 1: BASH

REFERENCES

- "Basic UNIX commands" [link]
- "Bash Guide for Beginners" [link]
- "Advanced Bash-Scripting Guide" [link]

GETTING STARTED

Access Shell terminal in your computer

- Option 1: Web terminal
  - [http://www.webminal.org/terminal/]
- Option 2: Setting up Ubuntu through VirtualBox
  - TA will talk about this.

LECTURE 2: FILES & DIRECTORIES

DIRECTORIES

- List files and directories: `ls`
  - `ls ~`, `ls .`, `ls`
  - `ls /`
  - `ls -al`
- Enter a directory: `cd`
  - `cd`, `cd ~`, `cd ..`
  - `cd /`
- Print the current pathname: `pwd`
- Create a directory: `mkdir`
  - `mkdir dir_a`

BASIC FILE MANAGEMENT

- Create a file: `touch`
  - `touch file_a`
- Move a file (change file name): `mv`
  - `mv file_a file_b`
- Copy a file: `cp`
  - `cp file_a file_b`
- Remove a file: `rm`
  - `rm file_a`

BASIC FILE MANAGEMENT (2)

- Show the content of a file: `cat`, `more`
  - `cat file_a`
  - `more file_a`: use q to quit, / to search
  - Write text to a file: `echo >>`
    - `echo "Alice Bob" >> file_a`
    - `echo "Alice" >> file_b`,
      `echo "Alice" >> file_c`
- Show the count of lines/words/chars a file: `wc`
  - `wc file_a`
- Show difference between files: `diff`
  - `diff file_a file_b`

EXERCISE 2.1

1. Run command `ls -a /`. Copy and paste (C&P) the printout on BB.

2. Run command `cat file_b`. C&P printout on BB.

3. Create a directory `dir_b` under `dir_a` and enter it. C&P the commands on BB.

4. Create a text file named `file_d.txt` and put there the following string: `Charlie is a student`. Run `cat file_d.txt`.
   - C&P the list of commands and their printout on BB

LECTURE 3: FILE PERMISSION

REFERENCES

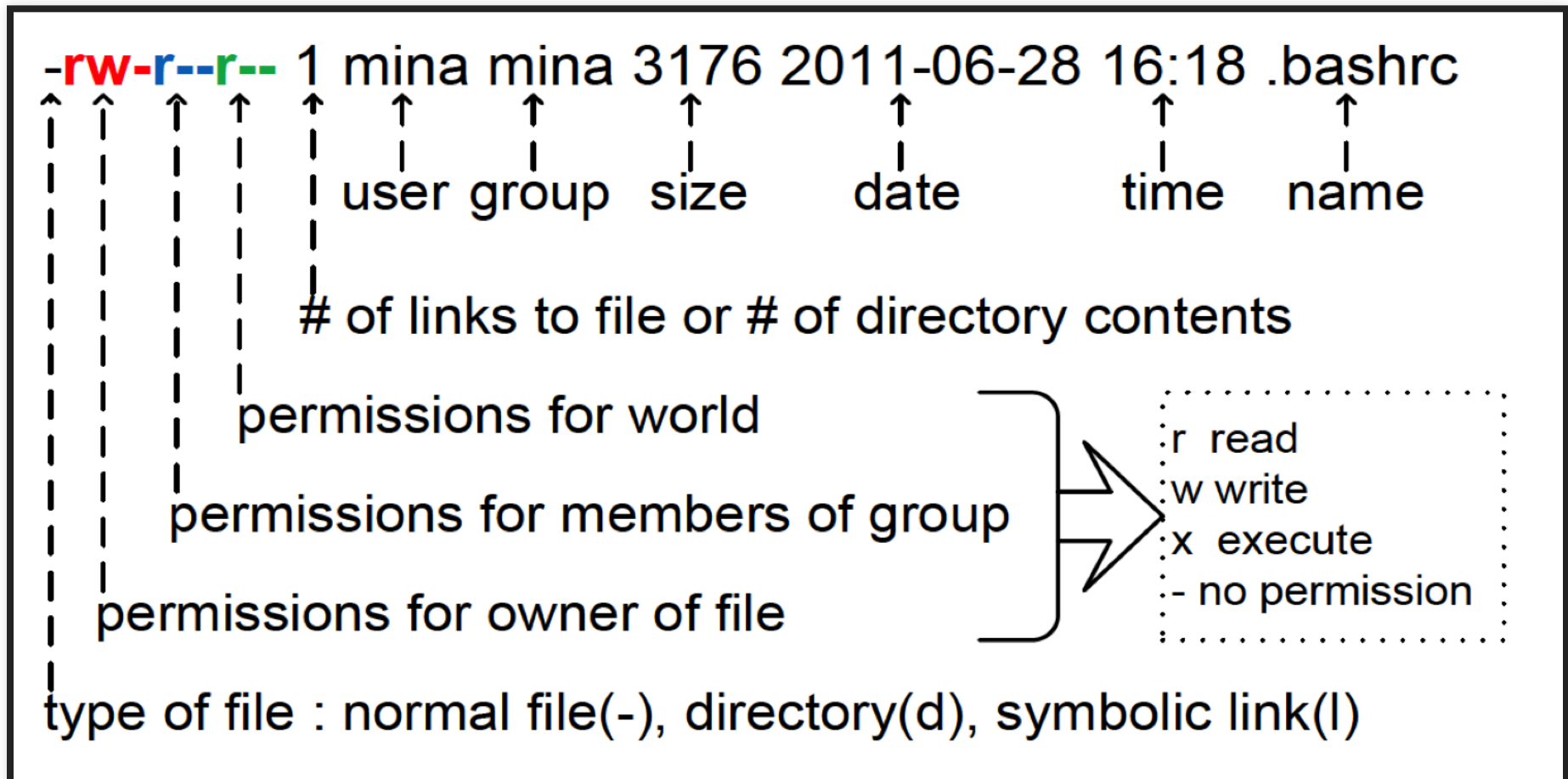- Understanding Linux file permissions [link]

BASIC CONCEPT

- file permission: access right, or file mode
  - permission controls the ability of a *user* to take *actions* on a *file*
  - user: owner, group, all users
    - group: group of users and files.
  - type: read, write, execute

VIEWING PERMISSION

```
ls -l
```

- owner and group
- permissions
  - users: owner (u), group (g), others (o), all users (a)
  - type: read (r), write (w), execute (x)

-rw-r--r-- 1 mina mina 3176 2011-06-28 16:18 .bashrc

user group size date time name

# of links to file or # of directory contents

permissions for world

permissions for members of group

permissions for owner of file

r read
w write
x execute
- no permission

type of file : normal file(-), directory(d), symbolic link(l)

ls -al

CHANGING PERMISSION

- chmod: change mode
  - add +:
    - `chmod a+wx file_a`: add write/execute permission to all users
    - `chmod g+r file_a`: add read permission to group users
  - assign/copy =:
    - `chmod g=rw file_a`: assign read/write permission to group
    - `chmod g=u file_a`: copy owner permission to group permission

CHANGING PERMISSION (2)

Options	Definitions
u	Owner
g	Group
o	Other
a	All (same as ugo)
x	Execute
w	Write
r	Read
+	Add permission
-	Remove permission
=	Set permission

CHANGING PERMISSION: NUMERIC MODE (3)

- `chmod 777 file_a;chmod a+rwx file_a`
  - `chmod 666 file_a;chmod a=rw file_a`
  - `chmod 000 file_a;chmod a-rwx file_a`
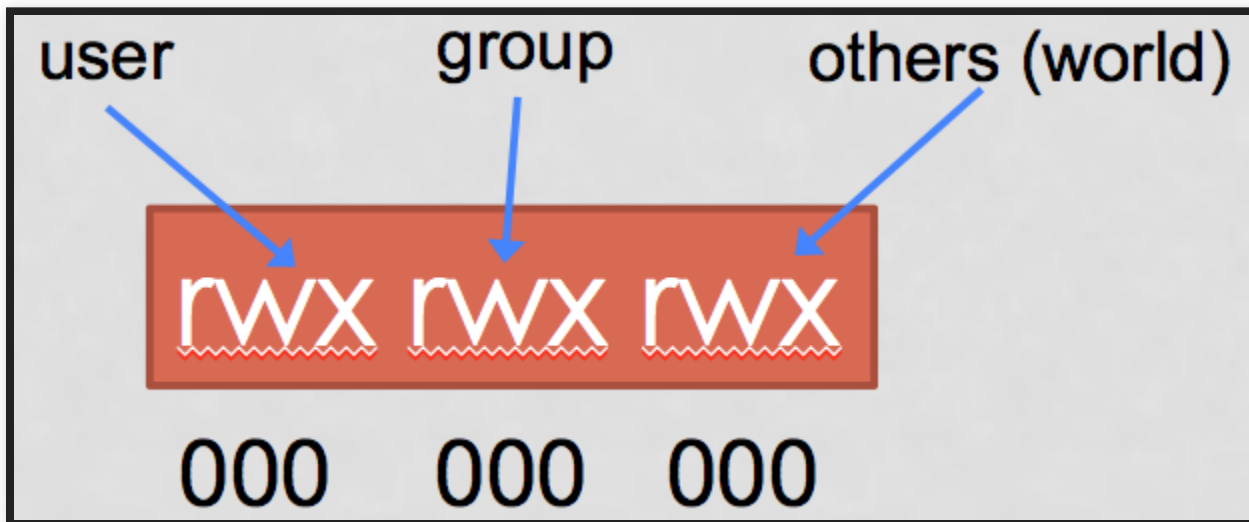
CHANGE OWNERSHIP

- chown owner:group filename
  - chown user1:staff file_a

EXERCISE 3.1

1. Run command `chmod o-r file_a; cat file_a`. C&P the printout on BB.
2. Design the command to make a file read-only to group. C&P your command on BB.
3. Design the command to make a file read-only to all users. C&P your command on BB.
4. Convert the following two commands to numeric mode: `chmod a-rwx file_a; chmod o+x file_a`. C&P your command on BB.

LECTURE 4: TEXT EDITING

- `gedit`: text editor with GUI
  - `gedit filename &`
- `vim`: text editor in terminal
- other editors: `emacs`, etc.

VIM

- basic movement: `h,j,k,l`
  - word movement: `w,e,b`
    - `word`, `end`, `begin`
  - number powered movement: `5w`
- find character in current line: `f`
  - `fq`: find char `q` in current line
  - `2fn`: find the second char `n` in current line
  - find word under cursor: * (next) and # (previous)
  - go to matching parentheses: `%`

VIM (2)

- begin and end of line: `0` and `$`
- go to line: `g`
  - first line: `gg`
  - last line: `G`
  - 10th line: `10gg`
- search: `/keyword` with `n` and `N`

VIM (3)

- modes: normal and insert
- from normal to insert: `i`, `o`, `R`
  - backward: `esc`
- editing in normal mode
  - copy/yank : `v+y`
  - and paste: `p`
  - cut: `v+x`
  - delete: `v+d`
  - undo/redo: `u,R`
- save/exit a file
  - write: `:w`
  - quit: `:q`

DEMO/PRACTICE

- Install vim on your VM: `sudo apt-get install vim`
- Or use online Vim: http://www.openvim.com/

EXERCISE

1. Write down the action sequence that searches String `Alice` in a file opened in vim

2. Open `file_a.txt` using vim. Insert your name in the file, save it, and close the file. C&P the actions you used.

3. What do the following action sequence do? (You can test it in vim)

   - `v+p`
   - `6e`
   - `j`

LECTURE 5: SHELL SCRIPTING (1)

INTRODUCTION: SHELL

- Linux Shell
  - Linux shell is a *program* that interpretes user *commands* from users and execute them by interacting with Linux OS kernel.
  - Different distributions:
    - `sh`: Bourne shell
    - `bash`: Bourne Again shell, superset of `sh`
    - others: `ksh`, `csh`

INTRODUCTION: SHELL SCRIPTING

- Two ways to put shell commands
  1. terminal: run commands one by one
  2. scripting: put shell commands in a file
- Why learn shell scripting/programming?
  - automate administrative tasks, save your efforts!
  - e.g. automatic software update, file backup, resource monitoring

SHELL SCRIPT BASICS: SHA-BANG

- Your first script
  - `#!` sha-bang is a two-byte magic number
  - basically says it's an executable shell script
- To execute a script `script.sh`:
  - `./script.sh`
  - `source script.sh`

SHELL SCRIPT BASICS: LANGUAGE

- Script is a group of commands:
  1. `#!/bin/bash echo 'hello world';`
- Variable:
  - Reference a variable: $a
  - untyped: integer, char, etc.
  2. `#!/bin/bash a=1;b=2;a=$a+$b;echo $a;`
  3. `#!/bin/bash a=1;b=2; a=$((a+b));echo $a;`
- If/else
  3. `if [ $a -gt $b ];then echo 'a larger than b`
    - `fi` declares the end of if/else clause

EXERCISE

1. Put commands
   `a=1;b=2;c=$a;a=$b;b=$c;echo $a,$b;` in a script,
   execute it, and put the printout to BB.
2. Write a script to initialize variables a, b, c with 1,2,3, and
   print their sum.
3. Write a script to swap the names of two files, file1 and file2.
   For example if input file1 contains Alice and file2 contains
   Bob at the beginning, after the execution, file1 should
   contain Bob and file2 should contain Alice.

COMMENTING

- # is used to comment in bash

LECTURE 6: SHELL SCRIPTING (2)

SHELL INITIALIZATION

- `~/.bashrc` (bash run commands)
  - script run everytime you open a terminal
  - sample:

    `export PS1='\[\033[0;35m\]\W\[\033[00m\]>`
- `~/.bash_profile`
  - run everytime you login
- user configuration files (not system-wide)

SHELL VARIABLES

- Global versus local
  - Global: environment variables
  - `env` to list all enivornment variables system-wide.
- naming convention:
  - shell var name include char and digit
  - name does not start with digit: `1x=5`

SHELL VARIABLES: LIFE CYCLE

life cycle	local var	global var
define & init	`set lvar='5'; lvar2=6`	`export gvar=7`
reference	`echo $lvar`	`echo $gvar`
destroy	`unset lvar`	

RESERVED AND SPECIAL VARIABLES

- Bash reserved variables:
  - `echo $HOME`
  - `echo $PATH`
  - `echo $PS1; $PS2`
  - `echo $BASH; $BASH_VERSION`
- Bash special parameters:
  - `echo $?`: exit status of the last command executed
  - `echo $0`: name of shell
  - `echo $1,$2`: 1th,2nd shell parameter
  - `echo $*,$#`: all positional parameters and the number of these parameters

PASSING PARAMETERS

- Demo:
  - `#!/bin/bash echo $1; echo $2; echo $#;`
- Exercise:
  1. `#!/bin/bash a=$1; b=$2; echo $((a*b));;` try this script and tell what it does?
  2. write a script to get 3 integers from the command-line and prints their product.
     - what happens if you do not pass the 3 required integers when executing the bash script?

LECTURE 7: SHELL SCRIPTING (3)

QUOTES

SHELL EXPANSION

SHELL DEBUG BY `SET -XVF`

- `set -x` print command traces
  - `bash script.sh`
  - `bash -x script.sh`
  - debug on parts: `set +x; w; set +x`
- `set -v`: prints input lines
- `set -f`: disable file name generation
  - `set -f; ls *; set +f`

ALIAS