

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](#)]

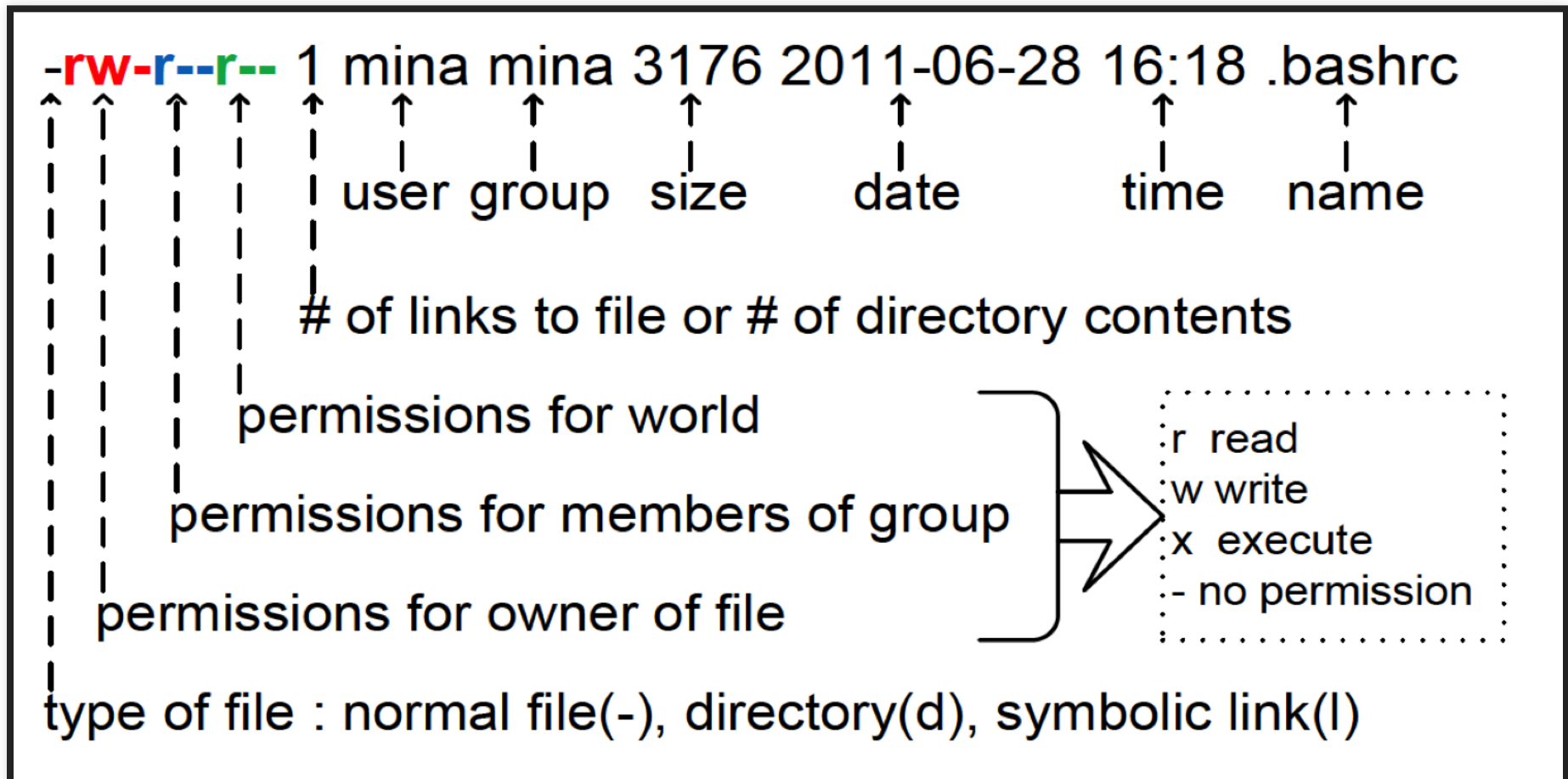
BASIC CONCEPT

- 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)



ls -al

CHANGING PERMISSION

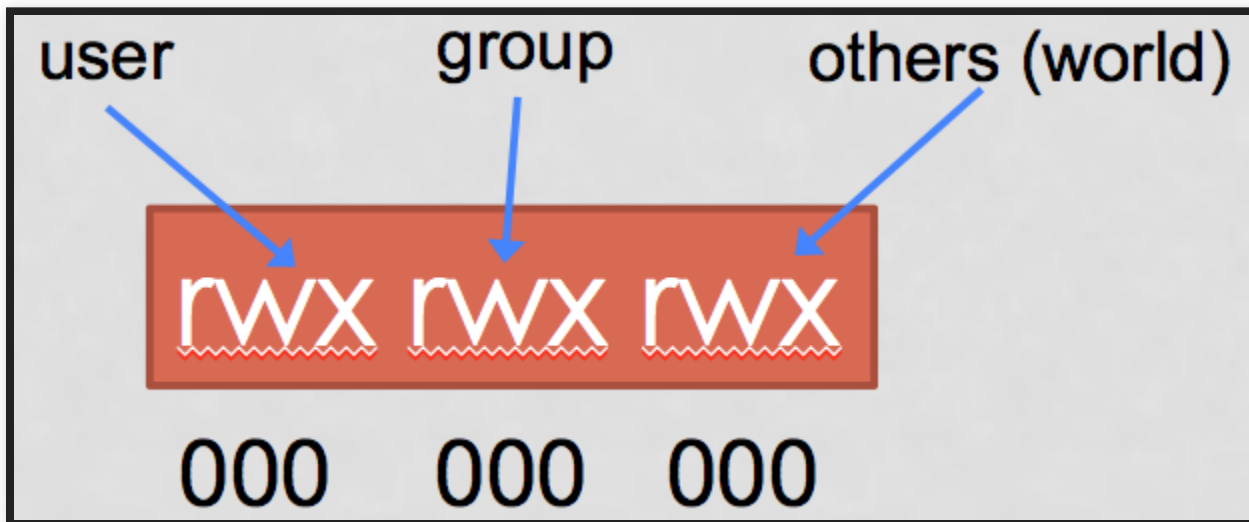
- `chmod`: change mode
 - `add +`:
 - `chmod a+wx file_a`: add write/execute permission to all users
 - `chmod g+r file_a`: add read permission to group users
 - `assign/copy =`:
 - `chmod g=rw file_a`: assign read/write permission to group
 - `chmod g=u file_a`: copy owner permission to group permission

CHANGING PERMISSION (2)

Options	Definitions
u	Owner
g	Group
o	Other
a	All (same as <u>ugo</u>)
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. Conver 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 parenthese: %

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/redu: `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 PROGRAMMING

INTRODUCTION

- Why learn shell programming?
 - automate administrative tasks, save your efforts!
 - e.g. automatic software update, file backup, resource monitoring
- Script: tie shell commands in a file
- Execute script `script.sh`:
 - `./script.sh`
 - `source script.sh`

BASICS: SHELL LANGUAGE

- `#!` sha-bang is a two-byte magic number that designates a file to be executed by a shell
 - basically says it's an executable shell script
- Language syntax: if/else, variable
- demo:
 1. `#!/bin/bash echo 'hello world';`
 2. `#!/bin/bash a=1;b=2;a=$((a+b));echo $a;`
 3. `#!/bin/bash a=1;b=2; a=$((a+b));echo $a;`
 4. `if [$a -gt $b];then echo 'a larger than b';`

- exercise:

1. * try `a=1 ; b=2 ; c=$a ; a=$b ; b=$c ; echo $a , $b ;`,
and put the output to the **blackboard**.
2. write a script to initialize variables `a`, `b`, `c` 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.

PASSING ARGUMENTS

- 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?

COMMENTING

- `#` is used to comment in bash