Intermediate Linux Tutorial January 2013

Jimmy Li

Basic commands you should already know

- cd - change directory
- ls - list directory content
- mkdir - make directory
- cp - copy
- mv - move
- rm - remove file/directory
- pwd - present working directory
- ssh - accessing a computer remotely
- man - reading man pages

Shell

- Program that is used to interact with the operating system
- Takes textual commands as input, and performs actions based on commands
  - When you type into the black terminal window, you are using a shell
- Different linux shells
  - sh, bash, csh, tcsh
- Shell script
  - Text file containing commands
  - Interpreted by a shell program

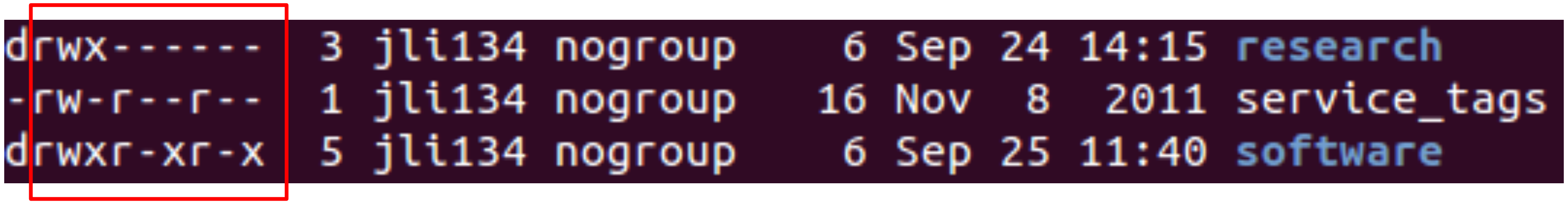
Bash (Borne Again Shell)

- Commands described on these slides are designed to be run under the bash shell
- If you are using another shell, like tcsh, syntax of commands may differ
- You can usually switch from another shell to bash by typing `bash` at the command line

Environment Variables

- Variables that describes the environment
- Use `echo` to print environment variables
- `echo $0` - print name of shell program you are using
- `echo $SHELL` - print path to shell program you are using
- Use export to set environment variable
- `export MYVAR=5` (no dollar sign!!!) - Set MYVAR to 5

Permission



```
drwx------  3 jli134 nogroup    6 Sep 24 14:15 research
-rw-r--r--  1 jli134 nogroup   16 Nov  8  2011 service_tags
drwxr-xr-x  5 jli134 nogroup    6 Sep 25 11:40 software
```

- View permissions with ls -l
- Permission indicated by 9 characters
  - First three characters indicate permissions of owner
  - Next three characters indicate permissions of group
  - Last three characters indicate permissions of everyone else
- r means readable, w means writable, x means executable

Change Permission

- Change permission with the chmod
- `chmod a+rx file` - give read and execute permissions to all
- u=owner, g=group, o=others, a=all
- r=readable, w=writable, x=executable
- + to add permissions, - to take away permissions, = to add permissions and take away unspecified permissions

Change Permission

- Fast way to use chmod
- Each of the three groups expressed as a binary number
- r-x can be expressed as 101, which is 5
- `chmod 755 file`
  - 755 corresponds to 111 101 101, which is rwxr-xr-x

How to run programs

- A user needs executable permission to run a program
- To run a program in the path, just type the program's name
  - Example: ls is a program. Because it is in the path, you can just type `ls` to run it
- The path is store in environment variable PATH
  - View your path with `echo $PATH`

How to run programs (cont'd)

- PATH is list of directories separated by colon
  - /usr/local/bin:/usr/lib/lightdm/lightdm:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
  - Note /bin is in the path. This is where `ls` resides
- Use `which` to see where a program is
  - `which ls`
- Use `whereis` to locate source and man files
  - `whereis ls`
- Adding new directory to PATH
  - `export PATH=/new/path:$PATH`
  - Do not do `export PATH=/new/path` because this will erase what was in your PATH

How to run programs (cont'd)

- To run program not in path, prepend `./` to the path to your program
- Example: Suppose you have navigated to the folder containing your program. You can then run your program with `./yourProgram`

Working with Text Files

- Reading text files
  - `cat file` - simply print content of file
  - `more file` - allows you to scroll through file
  - `less file` - allows you to scroll back and forth
  - `head file` - print the beginning of file
  - `tail file` - print the end of file
- Editing text files
  - `nano file` - simple text editor, easy to use
  - `vim file` - more powerful text editor, harder to use

grep

- Tool for searching through text
- Basic examples
  - `grep o myfile` - look for lines containing the letter o inside myfile
  - `grep o *` - look for lines containing the letter o inside all files in present working directory
- Regular expression pattern matching
  - `grep o$ *` - look for lines that end with o
  - `grep ^h *` - look for lines that begin with h
  - `grep ^h.*o$ *` - look for lines that begin with h and end with o

Piping, Redirection

- stdin (standard input stream)
- stdout (standard output stream)
- stderr (standard error stream)
- `P > F` - write stdout of program P into file F
- `P >> F` - append stdout of program P to file F
- `P 2> F` - write stderr of program P into file F
- `P1 | P2` - send stdout of program P1 to stdin of program P2

Using Piping and Redirect

- `ls > file` - write the output of ls to file
- `ls | grep Public` - search for Public in the output of ls
- `ps aux | grep nano` - search for a running process whose name contains 'nano'
- `du -ak . | sort -rn | less` - get disk usage in the current directory, sort this numerically in reverse order, then view the final result with less

Foreground and Background

- To start a program in background, append &
  - `nano &` - start nano and put it in the background
- control-z to put active program into background
  - Example: try starting nano with `nano` and then hitting control-z
- Type `jobs` to view processes that are in the background, and to see their job number
- Use `fg [job number]` to bring the corresponding process to the foreground

scp, rsync

- Tools for transferring files over network
- Format: scp source destination
- `scp file user@host:/path` - transfer file to remote machine
- `scp user@host:/path .` - transfer file from remote machine to present working directory on local machine
- rsync - can synchronize files and directories

scp example

- Transferring a file to school
    - scp myfile user@ubuntu.cs.mcgill.ca:/your/home
- `myfile` - file you want to send
- `user` - your username
- `ubuntu.cs.mcgill.ca` - this is the machine at school; you can also use mimi.cs.mcgill.ca
- `/your/home` - path on the remote machine where you want to store myfile; this should typically be somewhere inside your home directory. Use `pwd` to find out the path to your home

tar

- tar is an archiving utility
- A tar file is like a zip or rar file
- `tar cvzf mytar.tar.gz backup` - create an archive for the backup directory, and call it mytar.tar.gz
- `tar xvzf mytar.tar.gz` - extract the tar file mytar.tar.gz
- c=create, v=verbose, f=file, z=gzip, x=extract