Lecture 1

Introduction to Linux/Unix environment

1

1

Unix

- brief history:
  - Multics (1964) for mainframes
  - Unix (1969)
  - K&R
  - Linus Torvalds and Linux (1992)
- key Unix ideas:
  - written in a high-level language (C)
  - virtual memory
  - hierarchical file system; "everything" is a file
  - lots of small programs that work together to solve (giải quyết) larger problems
  - security, users, access, and groups
  - human-readable documentation included

2

2

Linux

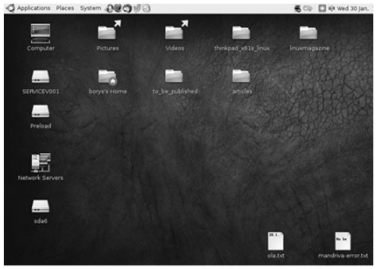
- **Linux**: A kernel for a Unix-like operating system.
  - Commonly (thường) seen/used today in servers, mobile/embedded devices, ...
- **GNU**: A "free software" implementation of many Unix-like tools
  - many GNU tools are distributed with the Linux kernel
- **distribution**: A pre-packaged (được đóng gói sẵn)set of Linux software.
  - examples: Ubuntu, Fedora, CentOS
- key features of Linux:
  - **open source software**: source can be downloaded
  - free to use
  - constantly being improved/updated by the community

3

3

Linux Desktop

- X-windows
- window managers
- desktop environments
  - Gnome
  - KDE
- How can I try out Linux?
  - CSE Virtual machine
  - CSE basement labs
  - Shared server



4

4

Things you can do in Linux

- Load the course web site in a browser
- Install and play games
- Play MP3s
- Edit photos
- IM, Skype

5

5

Shell

- **shell**: An interactive program that uses user input to manage the execution of other programs.
  - A command processor, typically runs in a text window.
  - User types commands, the shell runs the commands
  - Several different shell programs exist:
    - `bash` : the default shell program on most Linux/Unix systems
    - We will use bash
    - Other shells: Bourne, csh, tsch
- Why should I learn to use a shell when GUIs exist?

6

6

Why use a shell?

- Why should I learn to use a shell when GUIs exist?
  - Faster
  - Work remotely
  - Programmable
  - Customizable (tùy chỉnh)
  - Repeatable

7

7

Shell commands

command	description
`exit`	logs out of the shell
`ls`	lists files in a directory
`pwd`	**p**rint the current **w**orking **d**irectory
`cd`	**c**hanges the working **d**irectory
`man`	brings up the manual for a command

```
$ pwd
/homes/iws/rea
$ cd CSE391
$ ls
file1.txt file2.txt
$ ls -l
-rw-r--r-- 1 rea    fac_cs 0 2017-03-29 17:45 file1.txt
-rw-r--r-- 1 rea    fac_cs 0 2017-03-29 17:45 file2.txt
$ cd ..
$ man ls
$ exit
```

8

8

Relative directories

directory	description
`.`	the directory you are in ("working directory")
`..`	the parent of the working directory (`../..` is grandparent, etc.)
`~`	your **home** directory (on many systems, this is `/home/`***username***)
`~`***username***	***username***'s **home** directory
`~/Desktop`	your desktop

9

9

Directory commands

command	description
`ls`	list files in a directory
`pwd`	**p**rint the current **w**orking **d**irectory
`cd`	**c**hanges the working **d**irectory
`mkdir`	create a new directory
`rmdir`	delete a directory (must be empty)

- some commands (`cd`, `exit`) are part of the shell ("builtins")
- others (`ls`, `mkdir`) are separate programs the shell runs

10

10

Shell commands

- many accept **arguments** or **parameters**
  - example: `cp` (copy) accepts a source and destination file path
- a program uses 3 streams of information:
  - stdin, stdout, stderr (standard in, out, error)

- **input**: comes from user's keyboard
- **output**: goes to console
- **errors** can also be printed (by default, sent to console like output)
- parameters vs. input
  - *parameters*: before Enter is pressed; sent in by shell
  - *input*: after Enter is pressed; sent in by user

11

11

Command-line arguments

- most options are a `-` followed by a letter such as `-c`
  - some are longer words preceded (đi trước) by two `-` signs, such as `--count`
- options can be combined: `ls -l -a -r` can be `ls -lar`
- many programs accept a `--help` or `-help` option to give more information about that command (in addition to `man` pages)
  - or if you run the program with no arguments, it may print help info
- for many commands that accept a file name argument, if you omit (bỏ sót/quên) the parameter, it will read from standard input (your keyboard)

12

12

Shell/system commands

command	description
`man or info`	get help on a command
`clear`	clears out the output from the console
`exit`	exits and logs out of the shell

command	description
`date`	output the system date
`cal`	output a text calendar
`uname`	print information about the current system

- "man pages" are a very important way to learn new commands

```
man ls
man man
```

13

13

File commands

command	description
`cp`	copy a file
`mv`	move or rename a file
`rm`	delete a file
`touch`	create a new empty file, or update its last-modified time stamp

- caution: the above commands do not prompt for confirmation
  - easy to overwrite/delete a file; this setting can be overridden (how?)
- *Exercise* : Given several albums of `.mp3` files all in one folder, move them into separate folders by artist.
- *Exercise* : Modify a `.java file` to make it seem as though you finished writing it on Dec 28 at 4:56am.

14

14

Exercise Solutions

- caution: the cp, rm, mv commands do not prompt for confirmation
  - easy to overwrite/delete a file; this setting can be overridden (how?)
    - Use "-i" with the command, "interactive" to prompt before overwrite
- *Exercise* : Given several albums of `.mp3` files all in one folder, move them into separate folders by artist.
  - **`mkdir U2`**
  - **`mkdir PSY`**
  - **`mkdir JustinBieber`**
  - **`mv GangnamStyle.mp3 PSY/`**
  - **`mv Pride.mp3 U2/`**
- *Exercise* : Modify a `.java file` to make it seem as though you finished writing it on Dec 28 at 4:56am.
  - **`touch -t "201812280456" Hello.java`**

15

15

Basic Emacs Commands

- C- = control key M- = meta/alt key
- read a file into Emacs: C-x C-f
- save a file back to disk: C-x C-s
- exit Emacs permanently: C-x C-c
- search forward: C-s search backward: C-r
- scroll to next screen: C-v scroll to previous screen: M-v
- Undo: C-x u

entity to move over	backward	forward
character	C-b	C-f
word	M-b	M-f
line	C-p	C-n
go to line beginning/end	C-a	C-e
go to buffer beginning/end	M-<	M->

16

16

Basic Vim Commands

- :w Write the current file
- :wq Write the current file and exit.
- :q! Quit without writing
- To change into insert mode: i or a
  - Use escape to exit
- search forward /, repeat the search backwards: N
- Basic movement:
  - h l k j character left, right; line up, down (also arrow keys)
  - b w word/token left, right
  - ge e end of word/token left, right
  - 0 $ jump to first/last character on the line
- x delete
- u undo

17

17