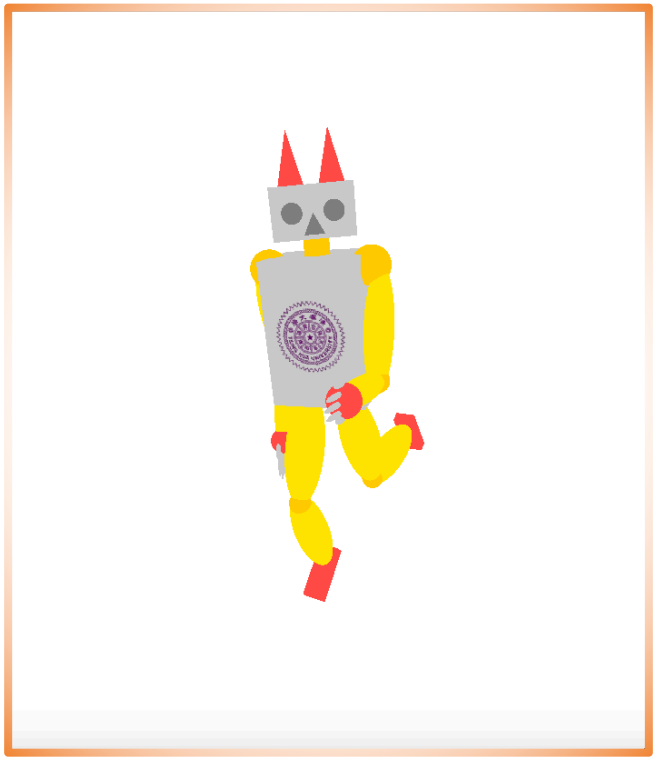
CS6135 VLSIPDA

Shell, Cygwin and Vim Tutorial



Yu-Hsiang Cheng

http://m105.nthu.edu.tw/~s105062901/Myweb/

References

- Learn X in Y minutes – X = Bash: https://learnxinyminutes.com/docs/zh-tw/bash-tw/
- Learn X in Y minutes – X = Vim: https://learnxinyminutes.com/docs/zh-cn/vim-cn/
- "Unix Guide with Vi Tutorial" by Professor Pai H. Chou (http://m105.nthu.edu.tw/~s105062901/Myweb/teaching/pdf/unix.pdf)

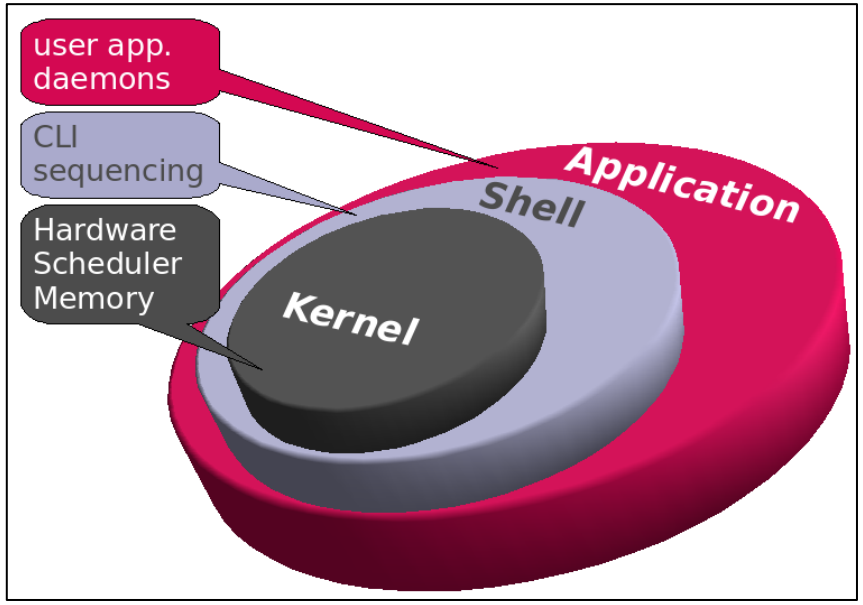
Outline

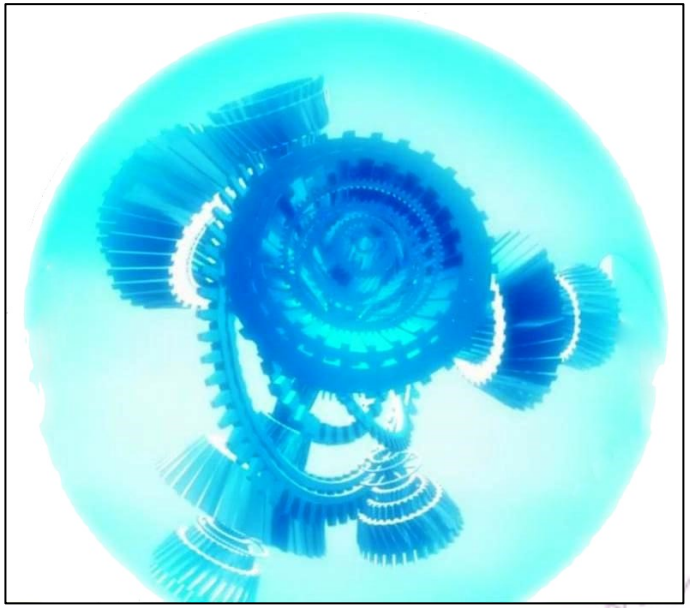
- Shell
- Cygwin
- Vim

Outline

- Shell
- Cygwin
- Vim

What is a Computer (1/2)



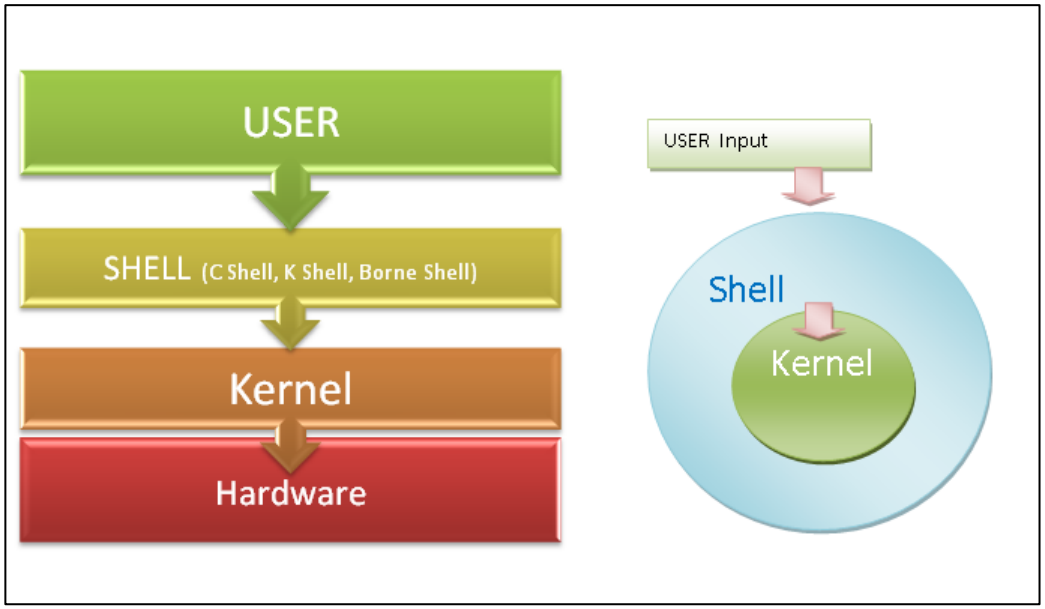


What is a Computer (2/2)

- 拿地球來做比喻的話
  - kernel: 地心
  - shell: 地殼
  - application: 房子、各種建築物
- shell = command interpreter = user interface
- shell 大抵上分兩種
  - GUI (graphical user interface, 圖形化使用者介面): Windows UI …
  - CUI (character user interface, 文字型使用者介面): sh, bash, csh, tcsh, ksh, zsh …
    - a.k.a. CLI (command-line interface, 命令行介面)
- bash: **B**ourne-**A**gain**SH**ell
  1. 最常用
  2. e.g. `Terminal.app` in OS X, `Cygwin.exe`, `MobaXterm` in Windows

Shell 的功能

- User 透過在 terminal (像 cmd, Cygwin terminal) 上輸入指令 (commands)，Shell 直譯它，並傳給 kernel，kernel 再驅動 hardware。



Shells on different Systems

Windows PC

	bash
PuTTY	Cygwin shell
graphical shell (Windows UI)	
OS (Widows)	
Computer (Intel x86 ISA), Keyboard, display, mouse..	

Unix-style commands:
ls, mv, cp, wc, who, ...

Mac OS X Computer

Terminal.app, iTerm.app, XTerm
text shell (bash, tcsh, ...)
graphical shell (Aqua UI)
OS (Unix)
Computer (Intel x86 ISA), Keyboard, display, mouse..

Unix-style commands:
ls, mv, cp, wc, who, ...

Source: Prof. Pai H. Chou

What can I do in a Shell?

- 顯示當下時間: `date`
- 列出檔案 (file listing): `ls`
- 重新命名或把 A 搬到 B (renaming, move A to B): `mv A B`
- 進入不同資料夾 (change directory): `cd some_where`
- 創造資料夾 (make directory): `mkdir dir_name`
- 刪除資料夾 (remove directory): `rmdir dir_name`
- 刪除檔案 (remove): `rm file_name`
- 更改一個檔案的timestamp，若無此檔案，則創造一個空的檔案: `touch abc.cpp`
- 編輯 (edit files with a text editor): `vi`, `vim`, `emacs`, `nano`…
- 編譯、組譯 (compile, assemble files): `cc`, `g++`, `as`, `lcs`…
- 跑程式 (run programs): `./a.exe`, `./a.out`…
- 登入其他 server (secure shell): `ssh username@ip`
- ✓查詢某個指令的輔助說明文件 (manual)：`man some_command`
- 連接檔案內容至另一個檔案尾端 (concatenate): `cat a.txt >> b.txt`
- 將檔案內容連接至螢幕 (讀檔輸出至螢幕): `cat filename`

What is a Command?

- 其實是個執行檔 (在 /bin/, /sbin/, /usr/bin/, /usr/local/bin/ … 裡)
- 格式：verb. [obj.] [-attribute_1 obj_1] [-attribute_2 obj_2] [-attribute_3] …
- 中括號代表這個東西可有可無的意思，參照 usage message: https://en.wikipedia.org/wiki/Usage_message
- 附加選項通常用 - 加一個英文字母(短選項)，或 -- 加一個字串 (長選項)。

```
ls                                  # list files
ls -a                               # list files all(including hidden files)
ls abc/                             # list files in abc/
ls -a abc/
vim abc.cpp
    # 用 vim 開啟一個叫做 abc.cpp 的檔案，無論存不存在
vim -b a.exe                        # open binary files with vim
g++ abc.cpp
    # compile a file called abc.cpp with g++
    # output file default name = a.exe or a.out
./a.out                             # 執行編譯後產生的程式
g++ abc.cpp -o abc.exe              # specify output file name
./abc.exe                           # 執行編譯後產生的程式
```

要知道的基本知識(重要常識) I

1. 指令其實就是執行檔、就是支程式，放在系統環境變數 PATH 所指定的一堆 folder 下，例如 `ls` 是放在 `/bin/` 下的 `ls` 執行檔，所以打 `ls` 其實是在執行 `/bin/ls` 這支程式
2. 在 Linux 上，名字加斜線代表它是資料夾 (Windows 則是反斜線)
   - e.g., `abc/`, `mycodes/`
3. `.` 或 `./` 代表當下所在資料夾
4. `..` 或 `../` 代表當下所在資料夾的上一層資料夾
5. `~` 或 `~/` 代表「使用者」根目錄 (一打開 `terminal` 所處的資料夾)
6. / 開頭的目錄是「系統」根目錄 (e.g., /bin/, /src/, /usr/bin/, etc.)
7. `.` 開頭的是隱藏檔或隱藏資料夾，要用 `ls -a` 才看得到
   - e.g. `.bashrc`, `.vimrc`, `.ssh/`, etc.
8. `*` 代表該資料夾下全部的檔案與資料夾
9. 按 tab 可以自動完成指令
10. 按↑可以回看打過的指令
11. 按 `Ctrl + L` 將螢幕清空 (或打 `clear` 指令)

要知道的基本知識(重要常識) II

1. 打 history 指令可以看所有打過的指令
2. history 的一些用法：https://blog.gtwang.org/linux/mastering-linux-command-line-history/
3. ctrl + c 中斷跑不完的程式
4. ctrl + d 登出
5. ctrl + d 也是檔尾字元 (Windows 下是 ctrl + z)
6. 通常 # 是 root 的 prompt，% 或 $ 是使用者 prompt。prompt 就是在告訴你「你要在這裡打字喔」的意思。
7. # 也有另一個意思，就是註解
8. 通常一個指令加 -h 或 --help 可看這個指令的用法，例如想知道 cowsay 這指令怎麼用就打 cowsay -h
9. man 後接一個指令會看到較完整的文件，例如 man cowsay (太完整這裡不列出，按 q 可離開文件)
10. 通常一個指令加 -v 或 --version 可看這個指令(程式)的版本，例如想知道 ruby 的版本就打 ruby -v

```
Slighten@SlightenCheng ttys000-bash 02:17 ~
$ cowsay -h
cow{say,think} version 3.03, (c) 1999 Tony Monroe
Usage: cowsay [-bdgpstwy] [-h] [-e eyes] [-f cowfile]
          [-l] [-n] [-T tongue] [-W wrapcolumn] [message]
```

```
Slighten@SlightenCheng ttys000-bash 02:23 ~
$ ruby -v
ruby 2.4.0p0 (2016-12-24 revision 57164) [x86_64-darwin16]
```

要知道的基本知識(重要常識) III

1. 如果你是 superuser (又稱做 **root**、**administrator** 或 **supervisor**)，你就可以隨意動系統裡的任何東西，若只是一般 user，那你就是 restricted，不是所有東西都能動，例如刪除 ls 這種重要系統程式的操作，一般使用者就不能做
2. 可以「sudo + 指令」，就是以 superuser 身份執行這個指令，要打密碼，但如果你是一般使用者沒有 root 密碼，那就無法執行
3. `su -` 或 `sudo su -` 可以把自己變成 superuser，當然需要 root 密碼。
   - `su` vs. `su -`: http://unix.stackexchange.com/questions/7013/why-do-we-use-su-and-not-just-su
   - `su` vs. `sudo su`: http://askubuntu.com/questions/620936/difference-between-su-and-sudo-su
4. Superuser 可以透過 `yum` (CentOS, RedHat, ...), `apt-get` (Ubuntu), `wget` (normal Linux), 或 `brew` (MacOS) 來管理套件 (安裝、刪除...)
   - 例如在 MacOS 上想裝 `astyle`，就 `brew install astyle` (安裝 Homebrew: https://brew.sh/index_zh-tw.html)
   - CentOS 上想裝 `perl`，就 `yum install perl`
   - ...
5. Superuser 可以透過 `adduser` 來創造使用者，`passwd` 修改該使用者密碼
   - `adduser Slighten`
   - `passwd Slighten`

Basic Shell Commands

- % is a prompt (不是要你打 %), # means comment
  - % **ls** *# list files*
  - % **ls** -al *# list files all, long*
  - % **cd** dirname *# change directory*
  - % **cd ..** *# go up one level (上移)*
  - % **cd** - *# go to last place (上一頁)*
  - % **cd** *# go to home directory*
  - % **cd ~** *# go to home directory*
  - % **mkdir** dirname *# make a directory*
  - % **pwd** *# print working directory*
  - % **more** filename *# look at a file*
  - % **cat** filename *# dump file content*
  - % **cat** a >> b *# append a's content to b*
  - % **diff** A B *# compare 2 files A, B*
  - % **man** ls *# see manual of ls*
  - % **man** man *# see manual of man*

More Shell Commands

- % **mv** oldname newname
  # rename a file as newname,
  # or "moving" it to another location
- % **cp** file1 file2
  # copy file1, call new one file2, or
  # copy file of same name to new loc
- % **rm** filename *# remove a file*
- % **rmdir** dir *# remove directory*
  - *But the directory must be empty*
  - *% **rm** -rf dir # recursively, force*
  - % rm -rf * # delete all things in cur. dir.
  - (Recall: * 代表該資料夾下全部的檔案與資料夾)

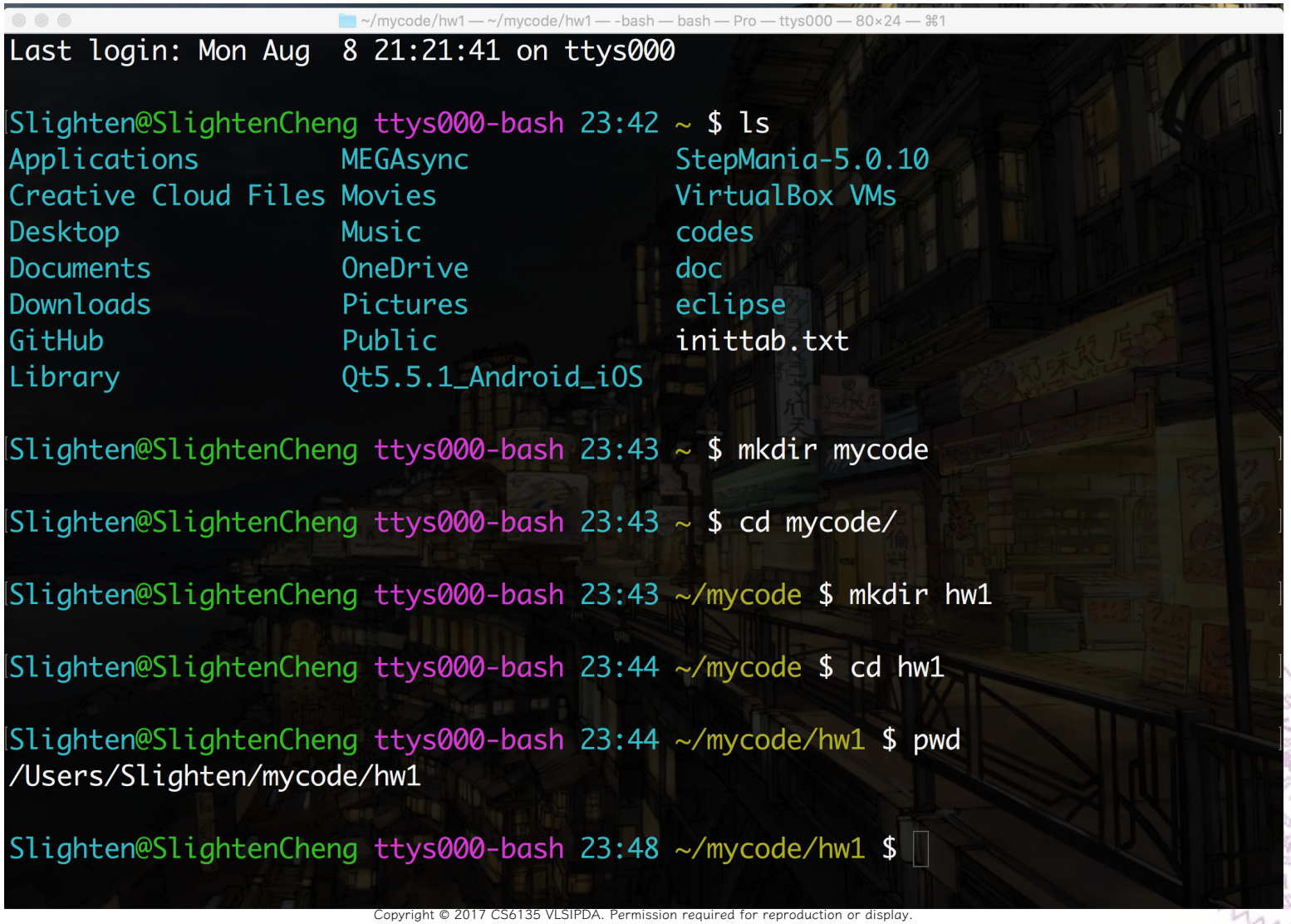
A Routine to Write a Program (1/5)

- 先打開 Terminal.app (OSX) 或 Cygwin.exe (Windows)
- 四步驟：先創資料夾，然後進入該處編輯檔案儲存，編譯該檔案，執行產生的執行檔。

% is a prompt (不是要你打 %), # means comment

- % pwd # 顯示所在資料夾路徑
- % ls # 顯示這個資料夾底下的資料夾與檔案
- % mkdir mycodes # 創造一個叫做 mycodes 的資料夾
- % cd mycodes # 進入它
- % mkdir hw1 # 在 mycodes 下創造一個叫做 hw1 的資料夾
- % cd hw1 # 進入它
- % pwd # 顯示現在路徑

A Routine to Write a Program (2/5)



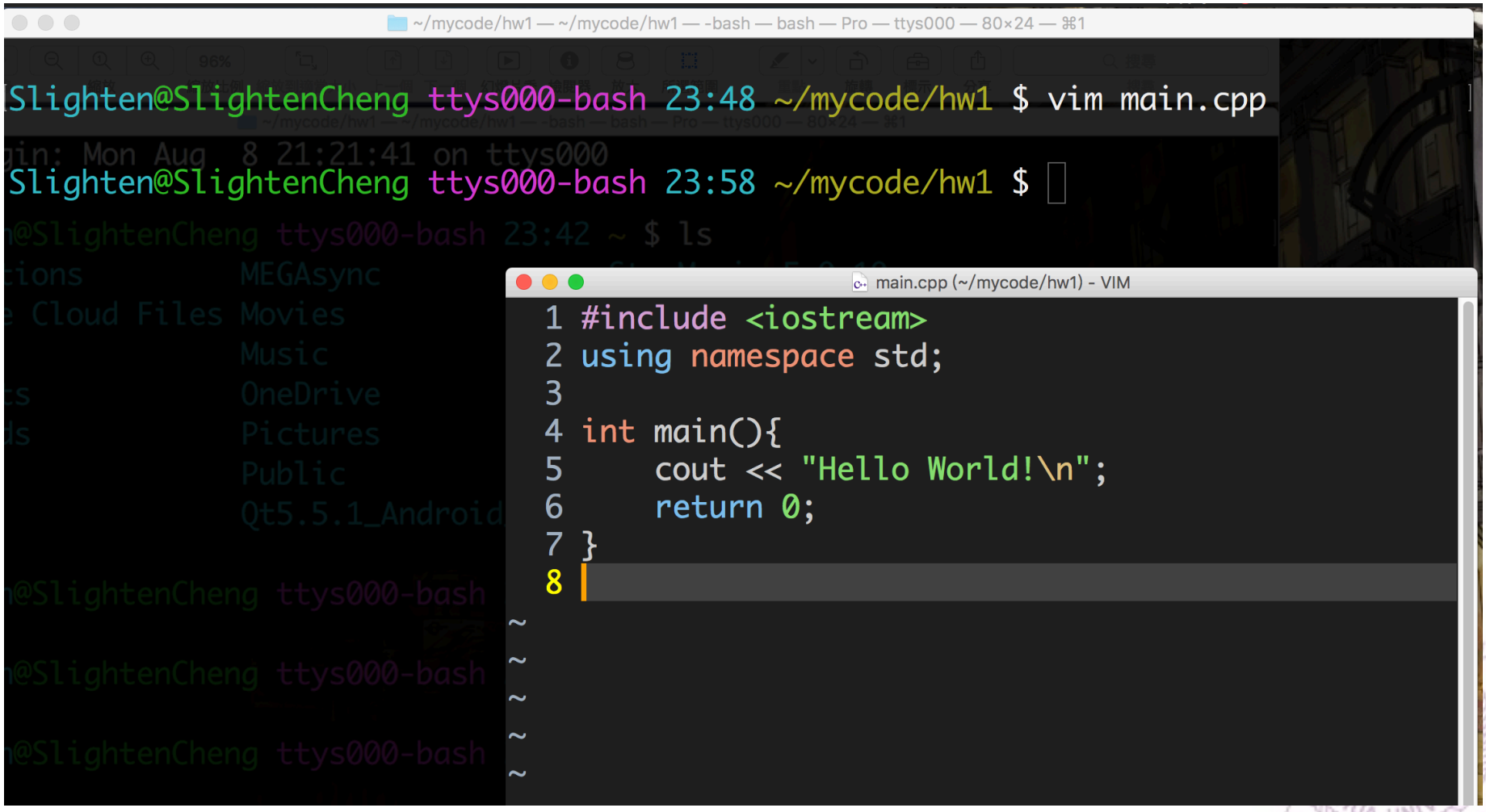
A Routine to Write a Program (3/5)

- 按 Ctrl + L 將螢幕清空 (或 % clear)

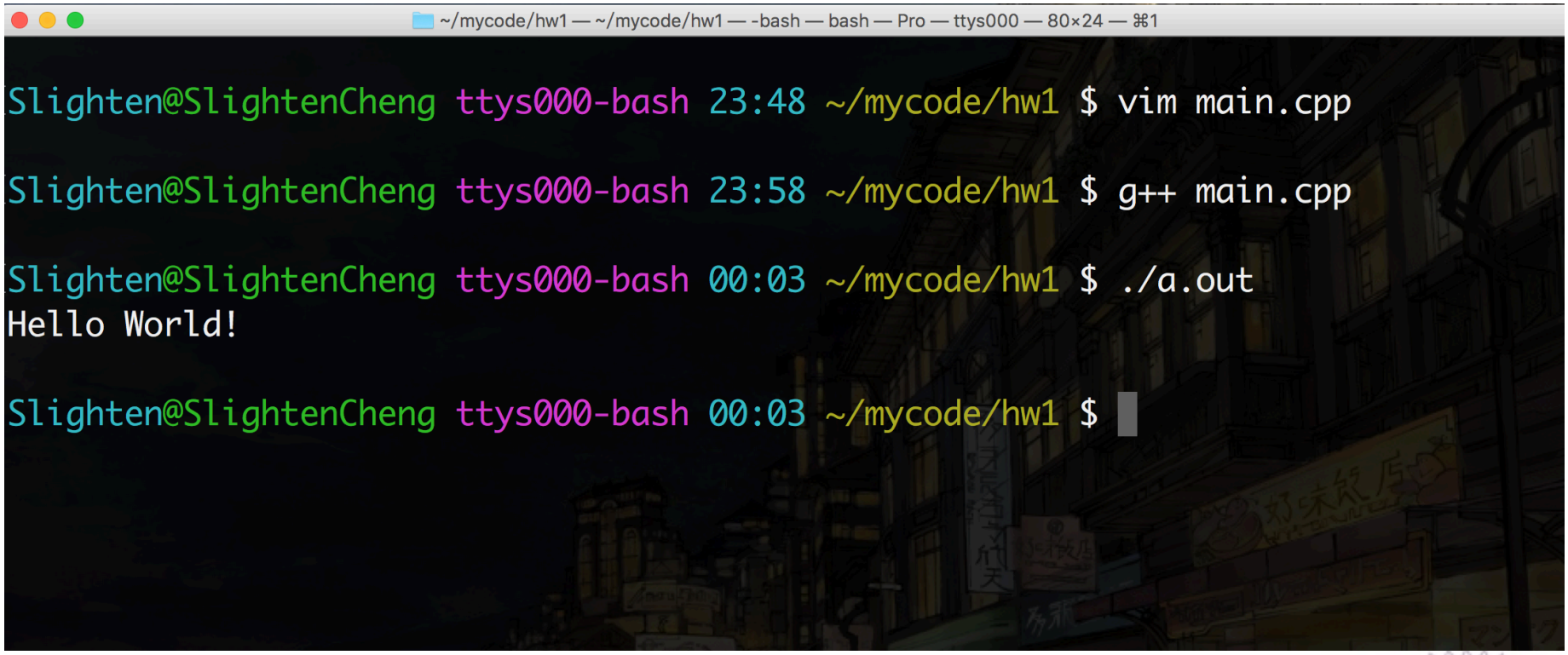
- % vim main.cpp
  - # 用 vim 開啟一個檔案，名叫 main.cpp
- 在 vim 打好程式碼 (vim 的操作後面會提)
- 存檔，離開 vim (ZZ)，回到 Terminal.app
- # 這部分也可以使用其他純文字編輯器做編輯
- # 例如用 Sublime Text 寫 code 存成 main.cpp 存在這個資料夾

- % g++ main.cpp
  - # 用 g++ 編譯器編譯 main.cpp，default 產生 a.out
- % ./a.out # 執行程式看結果

A Routine to Write a Program (4/5)



A Routine to Write a Program (5/5)



Outline

- Shell
- Cygwin
- Vim

What is Cygwin

- Linux-like (Unix-like) environment for Windows making it possible to port software running on POSIX systems (such as Linux, BSD, and Unix systems) to Windows.
- 簡單來說就是在 Windows 上的 Linux 模擬器。
- 所以只有 Windows 平台上有，Mac OS 用 Terminal.app 就行了

What is Linux (1/2)

- **Linux** 是一種自由和開放原始碼的類UNIX作業系統。
- 嚴格來講，術語 **Linux** 只表示作業系統核心本身。
- Linux 最初是作為支援 英特爾x86 架構的個人電腦的一個自由作業系統。
- 目前 Linux 已經被移植到更多的電腦硬體平台，遠遠超出其他任何作業系統。
- 最快的前 10 名超級電腦執行的都是基於Linux核心的作業系統。

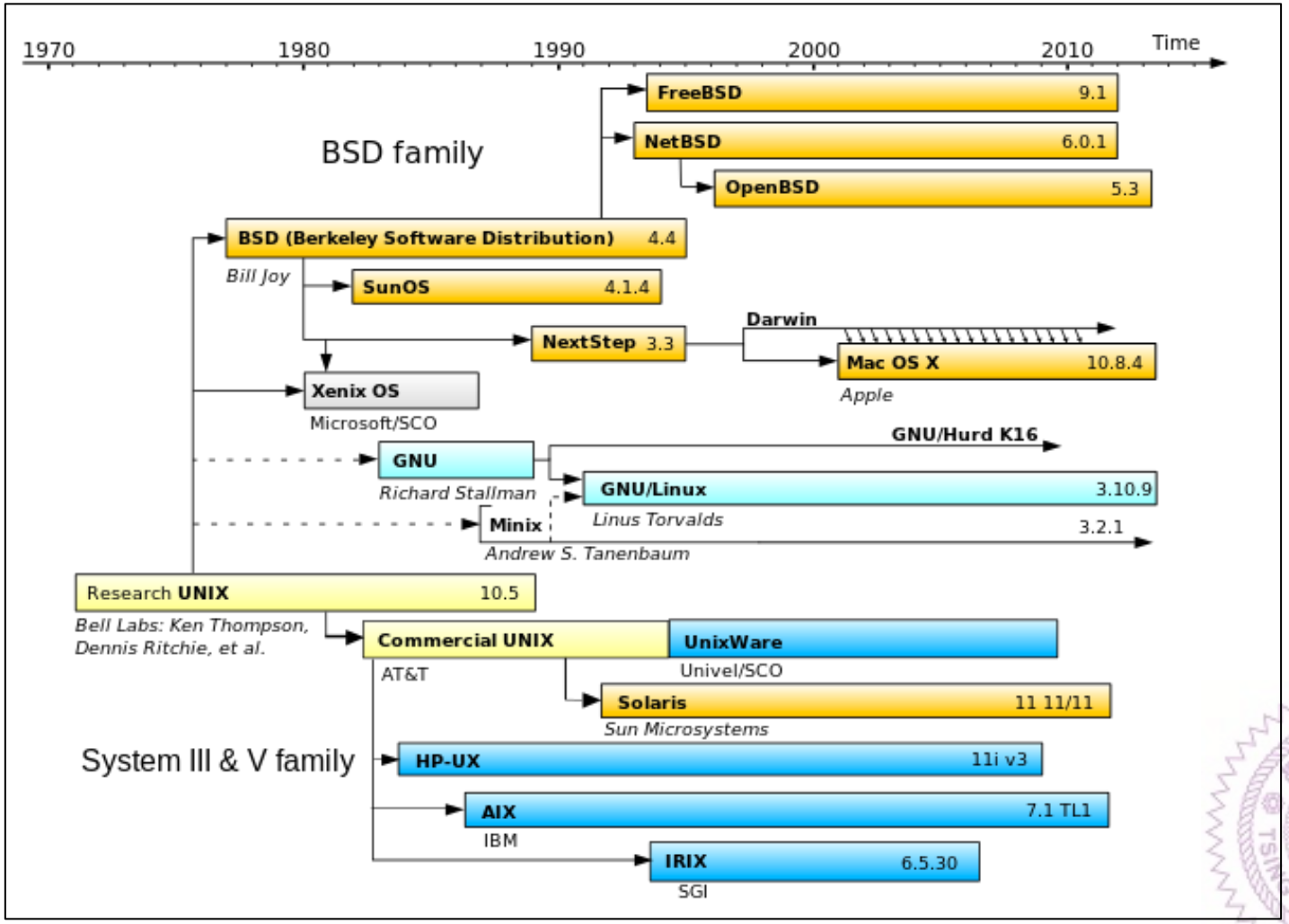
Source: wiki

What is Linux (2/2)

- Linux 也廣泛應用在嵌入式系統上，如手機（Mobile Phone）、平板電腦（Tablet）、路由器（Router）、電視（TV）和電子遊戲機等。在行動裝置上廣泛使用的Android作業系統就是建立在 Linux 核心之上。
- Linux 被打包成供個人電腦和伺服器使用的 Linux 套件，一些流行的主流 Linux 發布版，包括 Debian（及其衍生版本 Ubuntu、Linux Mint）、Fedora（及其相關版本Red Hat Enterprise Linux、CentOS）和 openSUSE 等。
- Linux 套件包含 Linux 核心和支撐核心的實用程式庫，通常還帶有大量可以滿足各類需求的應用程式。
- 個人電腦使用的Linux套件通常包含 X Window 和一個相應的桌面環境，如 GNOME 或 KDE。桌面 Linux 作業系統常用的 應用程式，包括 Firefox 網頁瀏覽器、LibreOffice 辦公軟體、GIMP 圖像處理工具等。

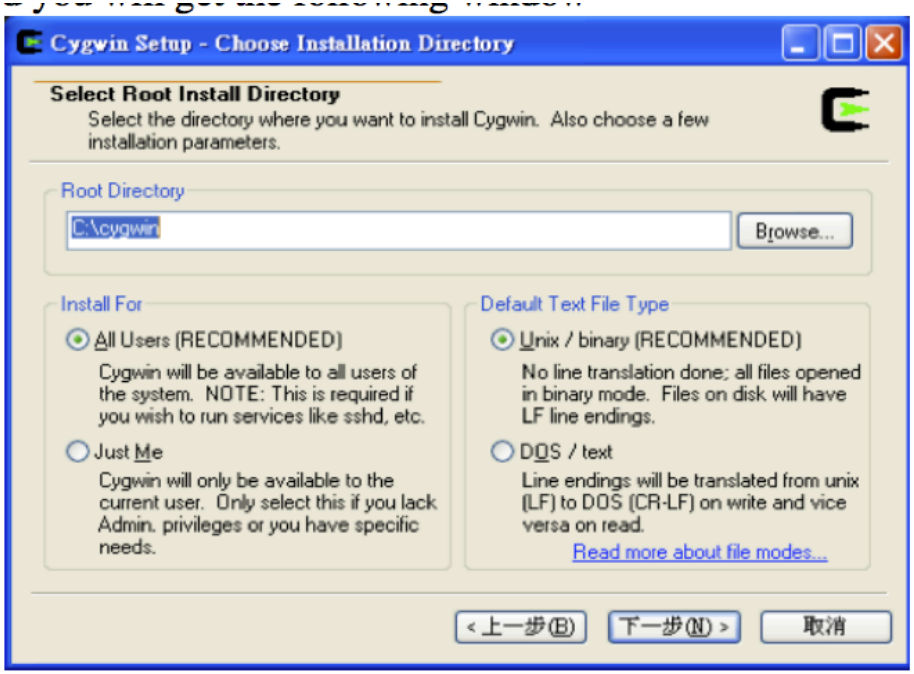
Source: wiki

Unix Genealogy Tree



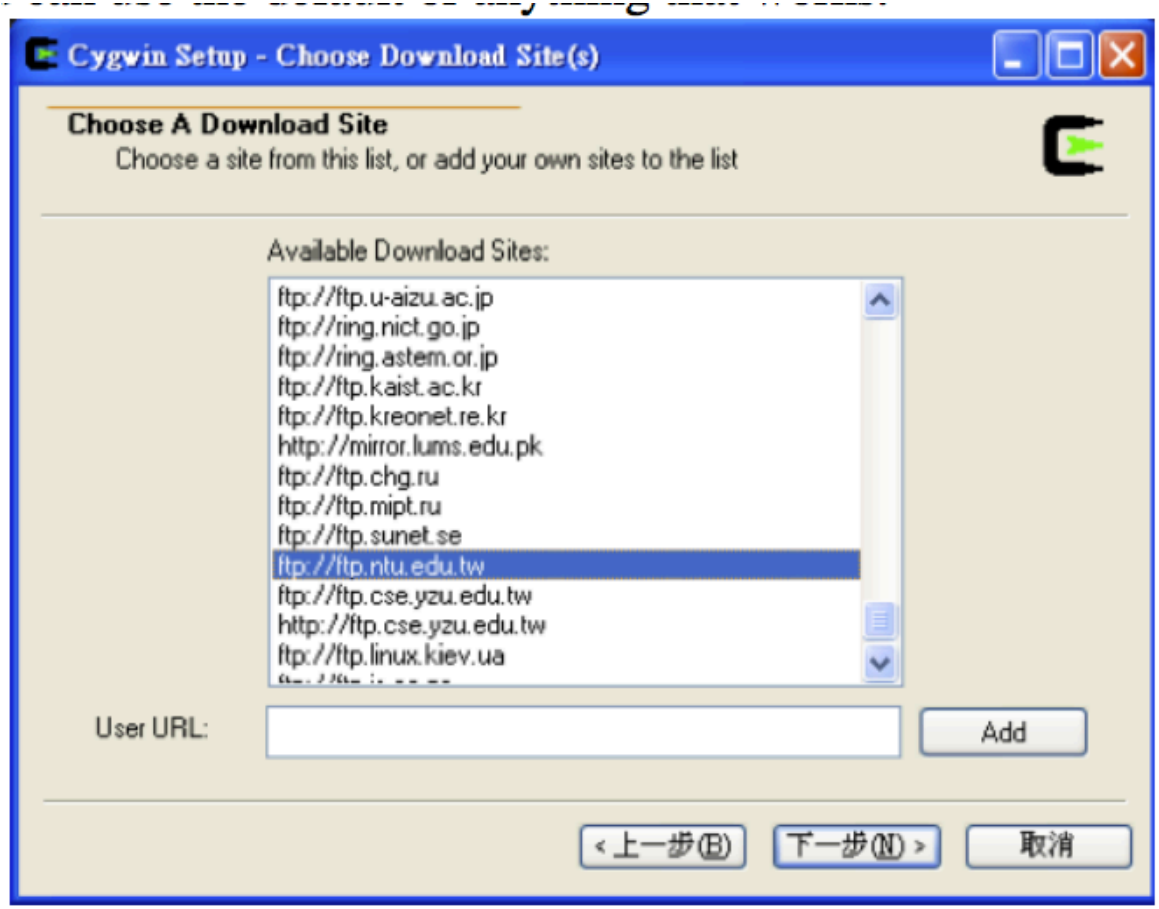
Demo: Install Cygwin (1/4)

- Download: https://cygwin.com/install.html
- 看你系統是 32 位元 (x86) 還 64 位元 (x86_64) (我的電腦右鍵看內容)
- 點開 setup-x86.exe (setup-x86_64.exe) 安裝，下一步



Demo: Install Cygwin (2/4)

- 這邊隨便指定一個下載套件的來源 (最好是 .edu，網路較穩定)



Demo: Install Cygwin (3/4)

- 在 Cygwin 上搜尋安裝、更新或刪除各種套件的畫面。

Demo: Install Cygwin (4/4)

- Minimally, it is recommended that you install
- 編譯器： g++ (gcc-c++) (gcc: GNU Compiler Collection).
  - – Develop > clang: C/C++/ObjC Compiler frontend based on LLVM
  - – Develop > gcc-g++: GNU Compiler Collection (C++)
  - – Develop > gdb: The GNU Debugger
  - – Develop > mingw-gcc-g++: GNU Compiler Collection (C++)
- make
  - – Develop > make: The GNU version of the 'make' utility
- 純文字編輯器：vim (vi-improved text editor).
  - – Editors > vim: Vi IMproved - enhanced vi editor

- 沒有找到就不用裝，想要什麼套件可以之後重點開安裝檔再裝，不急。
- e.g., python, java, nano, emacs 等等

Outline

- Shell
- Cygwin
- Vim

Text Editors in Unix

- Linux 上有以下幾種常用的純文字編輯器：
  - VI: (command-line)
    - 純文字編輯器 (plain text editor), available in most CLI
    - http://docs.freebsd.org/44doc/usd/12.vi/paper.pdf
  - Vim: 有語法高亮 (syntax highlighting)
  - Emacs: (command-line; also windows)
    - powerful programmable text editor
  - nano
- 通常 Linux 系統(剛安裝時)沒有 GUI (像工作站)，所以大多數人都在 terminal 上使用 Vim 編輯任何文字檔。
- CentOS 上有人會用 gedit 這個 GUI 純文字編輯器。

VI: Modal Editor

- Modal 就是「有模式 (mode) 的」，擁有幾種不同 mode 做切換
- 像 Caps Lock (大寫 vs 小寫), Insert
- Vim 有 2 種 modes

1. 命令模式 (Command mode) # 指令操作模式
   - 一開啟檔案 default 的模式
   - keys interpreted as commands
   - move cursor, delete, copy/paste, …
   - i (insert), a (append), or o (open to) change to Insert Mode
2. 插入模式 (Insert mode) # 打字模式
   - keys interpreted as data in document
   - Esc to change to Command Mode

Basic Vi Commands

- In Command mode:
  - h # left
  - j # down
  - k # up
  - l # right
  - w # skip a word forward
  - b # skip a word backward
  - i # insert (go to insert mode)
  - esc # back to command mode
  - G # 跑到頁尾
  - gg # 跑到頁首

More Vi Commands

- In Command mode:
  - dd # delete one line
  - d2d # delete 2 lines
  - dw # delete word
  - p # paste (deleted or yanked item)
  - yy # "yank" a line (copy onto "clipboard")
  - { # back paragraph
  - } # forward paragraph
  - . # repeat last insert
  - / # find next pattern, n: next, N: last
  - ? # find previous pattern, n: next, N: last

Vi Line Commands

- In Command mode:
  - `:w` # save
  - `:q` # quit
  - `:q!` # quit without saving
  - `:w filename` # save to another file name
  - `:e filename` # edit another file
  - `:e #` # back to current file
  - `:set nu` # show line numbers
  - `:set nonu` # hide line numbers
  - `:set ic` # case-insensitive search
  - `:set noic` # case-sensitive
  - `:%s/abc/xxx/g` # replace "abc" with "xxx" globally
    - 範圍內字串取代請參照：http://vim.wikia.com/wiki/Ranges

Question: How do I Save & Quit?

1. `:w` then `:q`
2. or just `:wq`
3. or `ZZ`

Vim: Vi-IMproved

- New feature, including syntax highlighting
  - more flexible cursor movements
    - ←↓↑→ 取代 `h, j, k, l`
- Syntax highlighting, do
  - `:syntax on` or `:syntax`
- Multiple-undo and redo
  - `u` undo, `^R` redo

Custom Resource Files

- Resource file: 初始設定檔，放在「使用者根目錄下」，以 . 開頭
- bash: .bashrc
- vim: .vimrc
- 在執行 bash 前，也就是開啟 terminal.app 或 cygwin.exe 的瞬間， .bashrc 會先被載入 (執行 tcsh 前 .tcshrc 會先被載入, …)
- 想在執行 bash 時，重新載入 .bashrc 可使用指令
  - % `source ~/.bashrc`
- 一些 default 的 .bashrc template：
  https://gist.github.com/marioBonales/1637696
- 在執行 vim 前 .vimrc 會先被載入
- 所以基本上 .xxxrc 都是使用者自定的設定檔 (rc == resource)

- 在 Unix/Linux 上，用 . 開頭的檔案都是隱藏檔
- 需 `ls -a` 才看得到

Custom Prompt

- Prompt 的格式隨系統而異，基本上是定義在 PS1 環境變數下，在 .bashrc 裡自定義 PS1 變數
- 例如我的 prompt
- 與 PS1 變數

```
Slighten@SlightenCheng ttys000-bash 01:34 ~
$
```

```
86 # prompt
87 export PS1="\[\e]0;\w\a\]\n\[\e[36m\]Slighten\[\e[32m\]@\H \[\e[35m\]\l\s \[\e[36m\]\A \[\e[33m\]\w\[\e[0m\]\n\$ "
```

- 簡單一點的 prompt
- PS1 變數就比較單純

```
[Slighten@localhost ~]$
[Slighten@localhost ~]$ echo $PS1
[\u@\h \W]\$
```

- 想自訂 prompt 格式: http://ezprompt.net/

Troubleshooting: Cannot Type Backspace in Vim on Cygwin

- 在 Insert mode 下，可能不能 backspace 刪字
- 需要在 command mode 打上這行解決
  - `:set backspace=start,eol,indent`
- 你可以在根目錄下創建 .vimrc 寫在裡面
- 每次開啟 vim 都會 default load .vimrc

- 一樣，若不喜歡每次開 vim 都要 :syntax on 或 :set nu，可以都把它們都寫在 .vimrc 裡面
- 也可以使用別人寫好的 .vimrc template (https://github.com/vgod/vimrc)
- Learn X in Y minutes — X = Vim 也有比較簡潔的 .vimrc template (https://learnxinyminutes.com/docs/zh-cn/vim-cn/)

Password-less ssh (1/2)

- You need to set up ssh public/private keys.
- First, generate the public/private keys pairs by dsa or rsa algorithms on your local machine. (only generate it once and it'll be OK)

```
% ssh-keygen -t dsa                    # keep entering all the way
```

  - It will generate id_dsa and id_dsa.pub in ~/.ssh/
  - Next, we copy the public key to the server and append it to .ssh/authorized_keys

```
% scp ~/.ssh/id_dsa.pub {username}@{serverIP}:.
```

  - Copy the public key to the server

```
% ssh {username}@{serverIP}          # log in to the server
% mkdir ~/.ssh                       # if the directory doesn't exist
% chmod 755 ~/.ssh
% cat ~/id_dsa.pub >> ~/.ssh/authorized_keys    # append l to r
```

  - append the public key to ~/.ssh/authorized_keys

```
% chmod 400 ~/.ssh/authorized_keys
```

  - Change mode (file permission) and make it inaccessible to others

```
% rm ~/id_dsa.pub
```

- We have set up the key pairs for your computer and the proxy.

Password-less ssh (2/2)

- We have set up the key pairs for your computer and the proxy. Now, we need to set up the key pairs for the proxy and ic21~ic29 (common file system)
- on proxy
- `% ssh-keygen -t dsa` `# keep entering all the way`
- `% chmod 755 ~/.ssh/authorized_keys` `# make it writable`
- `% cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys`
- `% chmod 400 ~/.ssh/authorized_keys` `# make it inaccessable`

- Done!

Appendix: Using MobaXterm

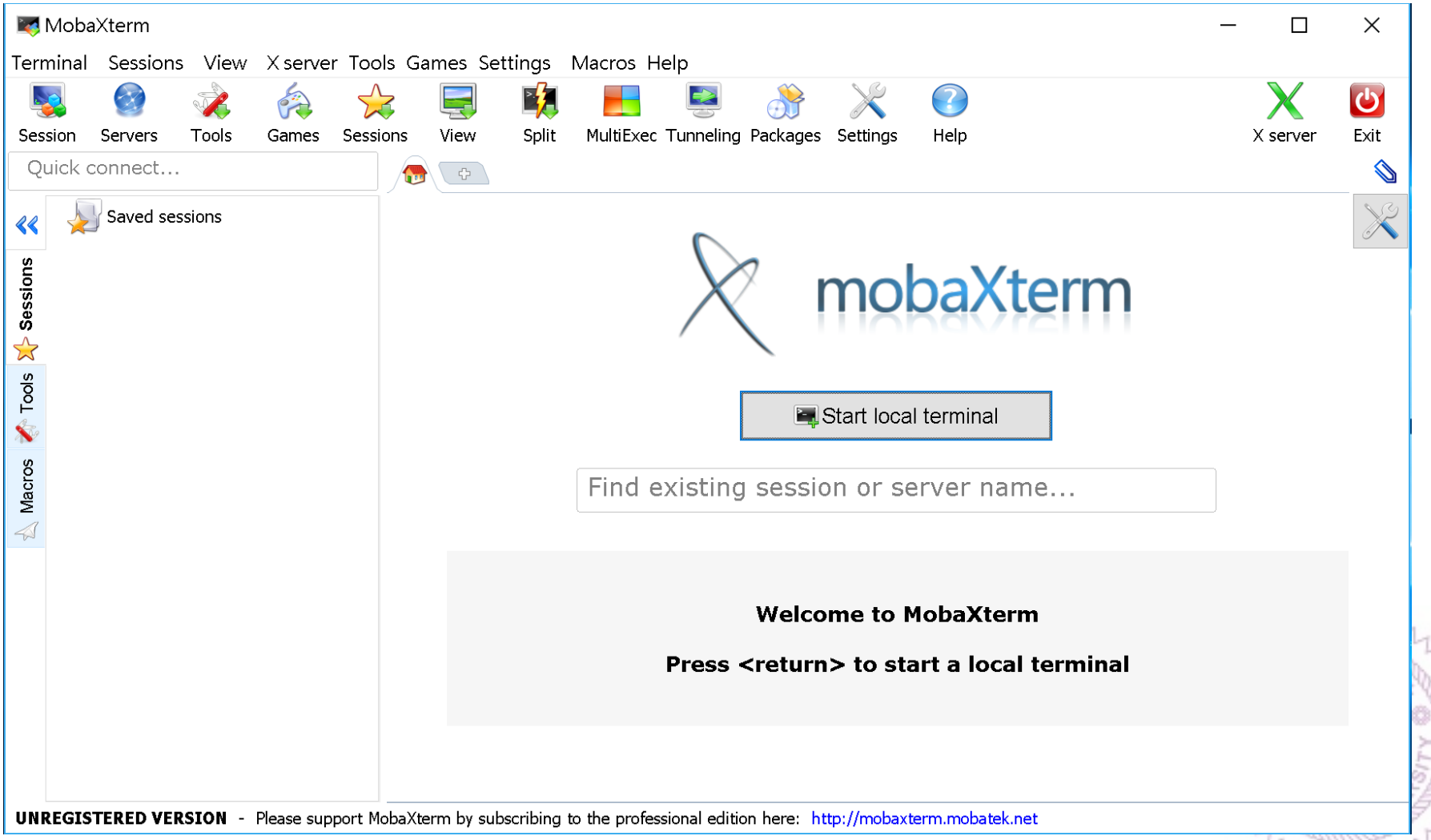
Appendix: Using MobaXterm

- Only supported in Windows
- Just like a lightweight Cygwin on Windows
- Cannot install additional packages like Cygwin

Download and Install

- Google it

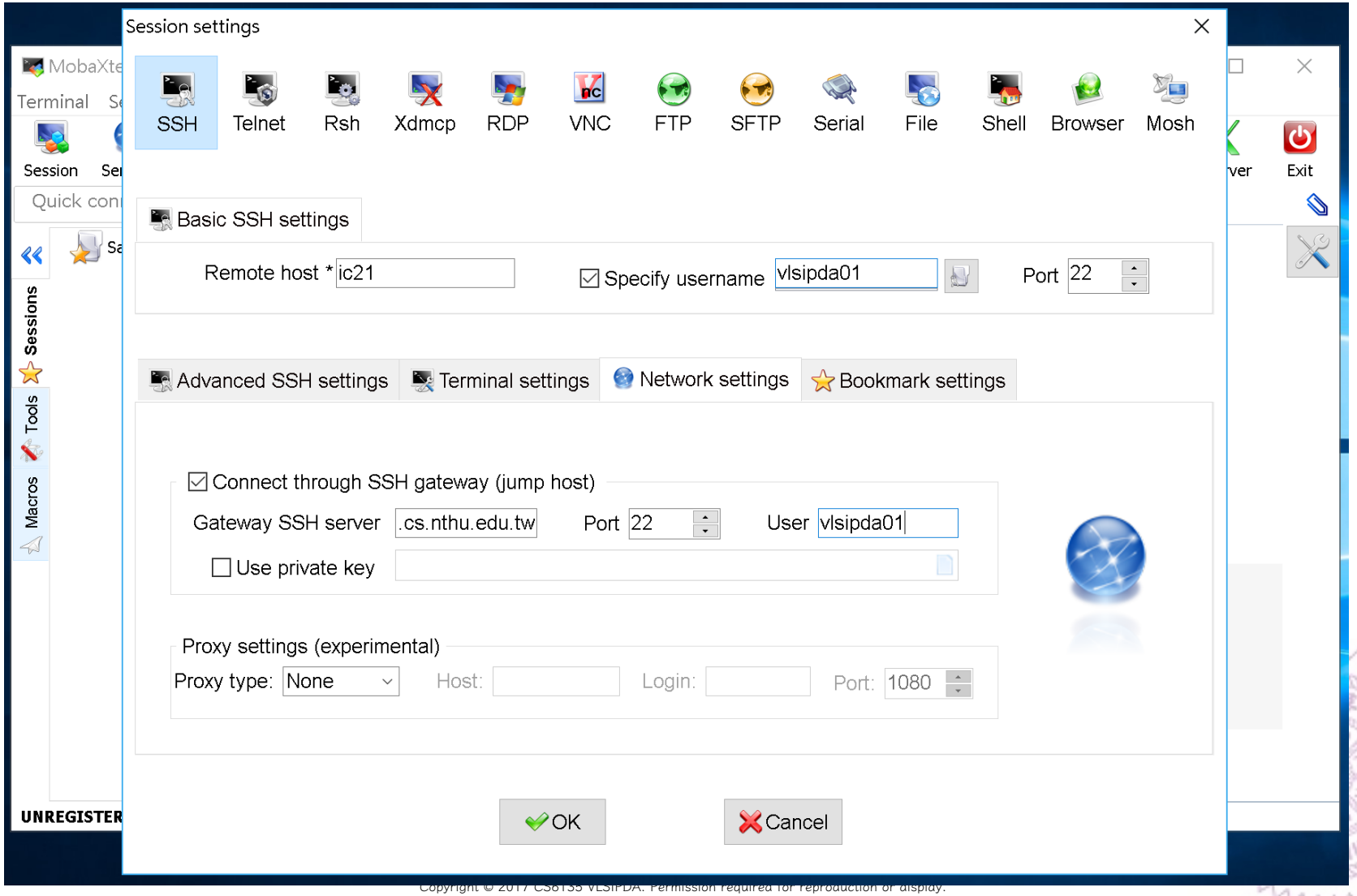
Open



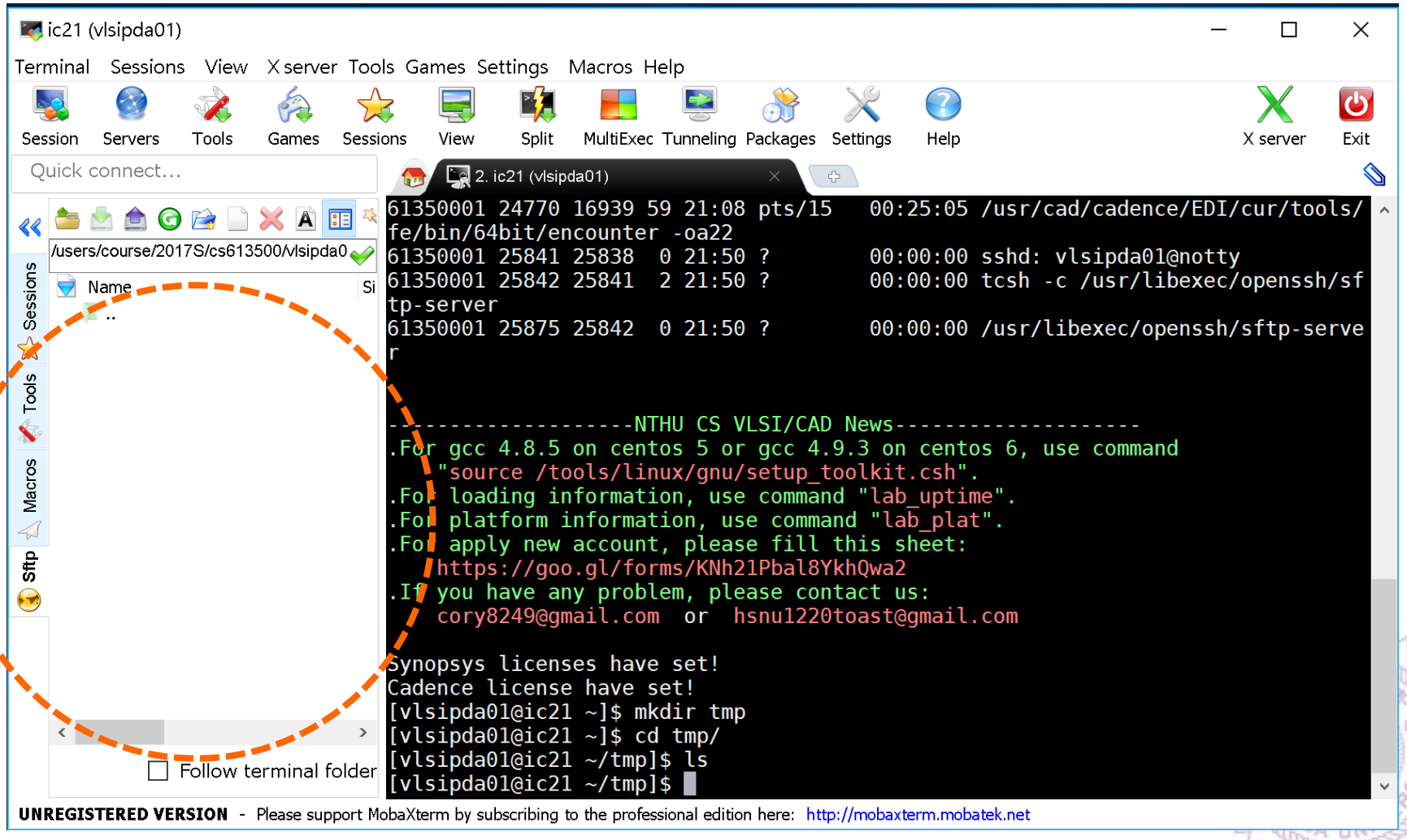
Edit a Session

- Click “Session” on the navigation bar.
  - Click ”SSH”
    - Remote Host: `ic21` (ic21~ic26)
    - Specify username: `vlsipda01` (User account)
  - Network Settings
    - Connect through SSH gateway (jump host)
    - Gateway SSH server: `nthucad.cs.nthu.edu.tw`
    - User: `vlsipda01` (User account)
    - OK

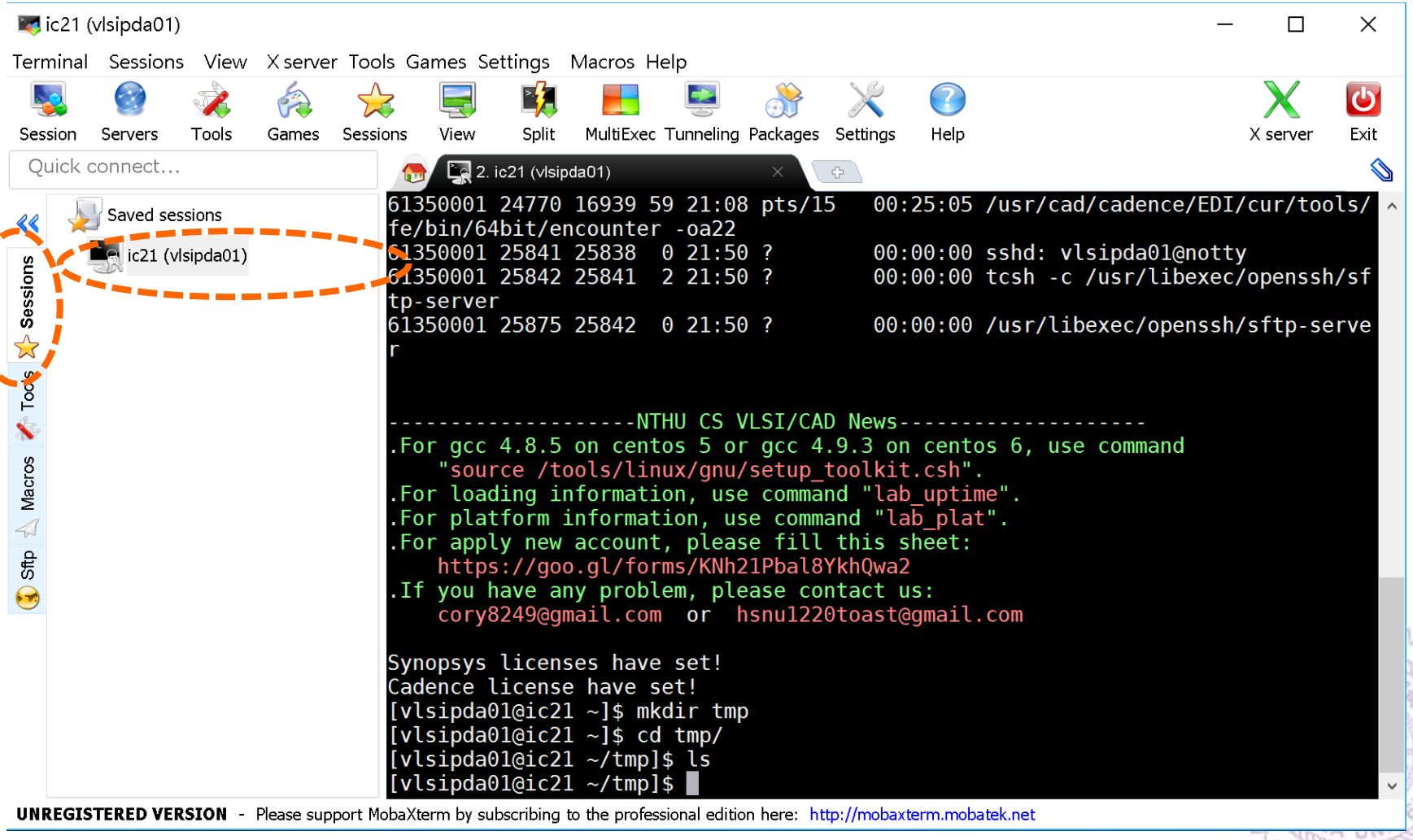
Edit a Session



Drag & Drop to Upload and Download (through sftp)



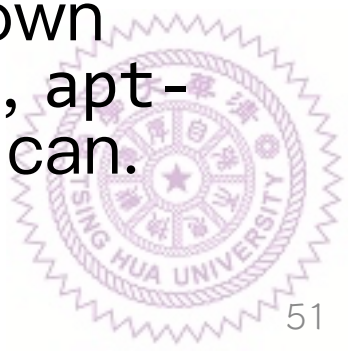
Double Click to Start a Session



Comparison

- Advantages
  1. Easy to learn, easy to use
  2. Automatically X11 forwarding
  3. Save your password(s)
  4. Portable

- Drawbacks
  1. For Windows only
  2. Non-upgradable (You cannot install your own packages on your local machine using `yum`, `apt-get`, or `brew`), but Cygwin or Terminal.app can.



References

- Learn X in Y minutes – X = Bash: https://learnxinyminutes.com/docs/zh-tw/bash-tw/
- Learn X in Y minutes – X = Vim: https://learnxinyminutes.com/docs/zh-cn/vim-cn/
- "Unix Guide with Vi Tutorial" by Professor Pai H. Chou (http://m105.nthu.edu.tw/~s105062901/Myweb/teaching/pdf/unix.pdf)