Computer Architecture

Lab1:Building Experiment Environment

VirtualBox Introduction

- VirtualBox can run many guest operating systems on a host operating system in X86/AMD64 machine.

- We use it to build a Linux environment for developing our full system simulation platform.

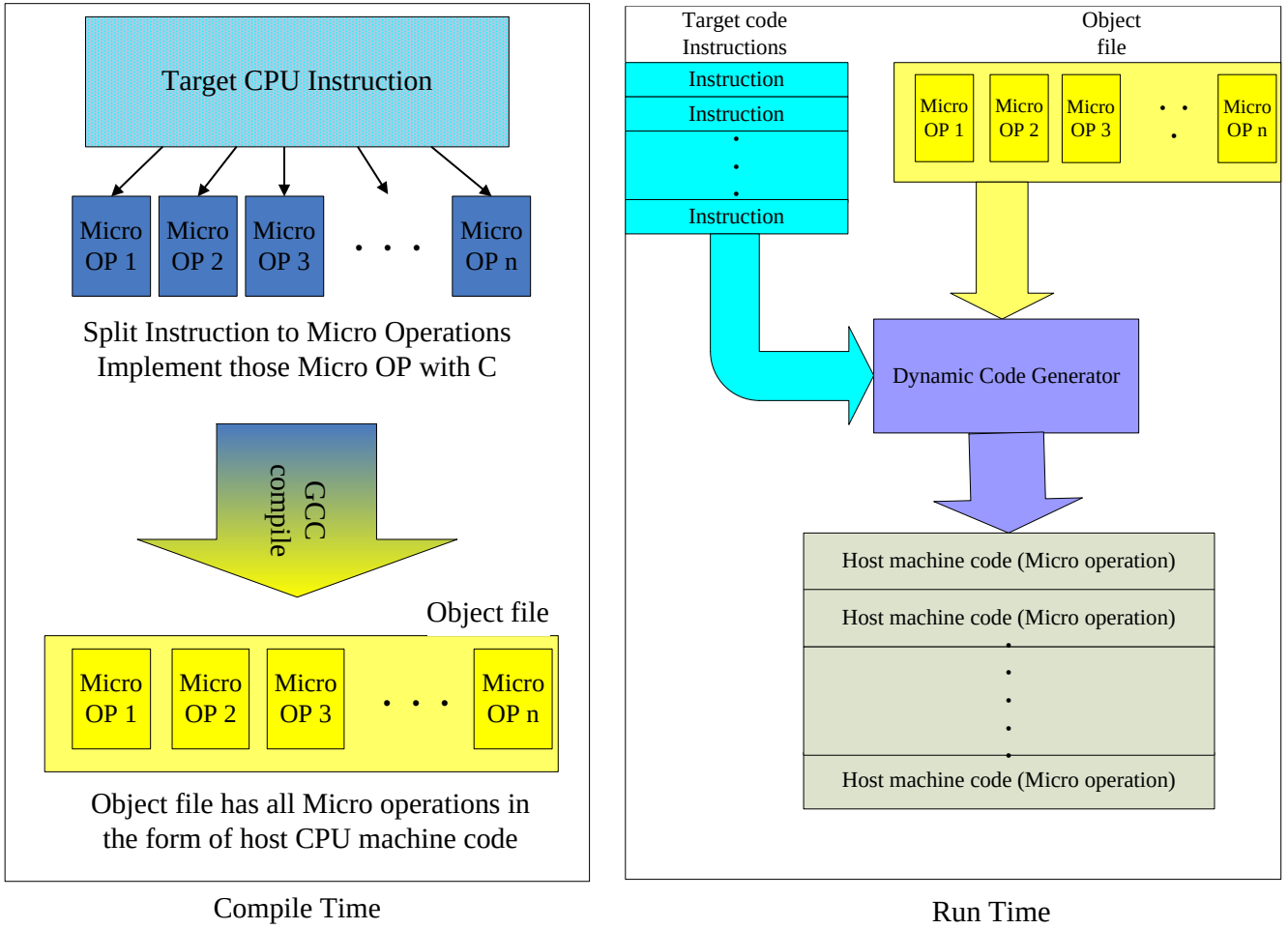
QEMU Introduction

- Unlike Virtualbox is binding with IA32, QEMU can run OSes and programs made for one machine on a different machine.

- QEMU makes this characteristic possible by supporting the portable dynamic translation.

QEMU Introduction (Cont.)

- portable dynamic translation

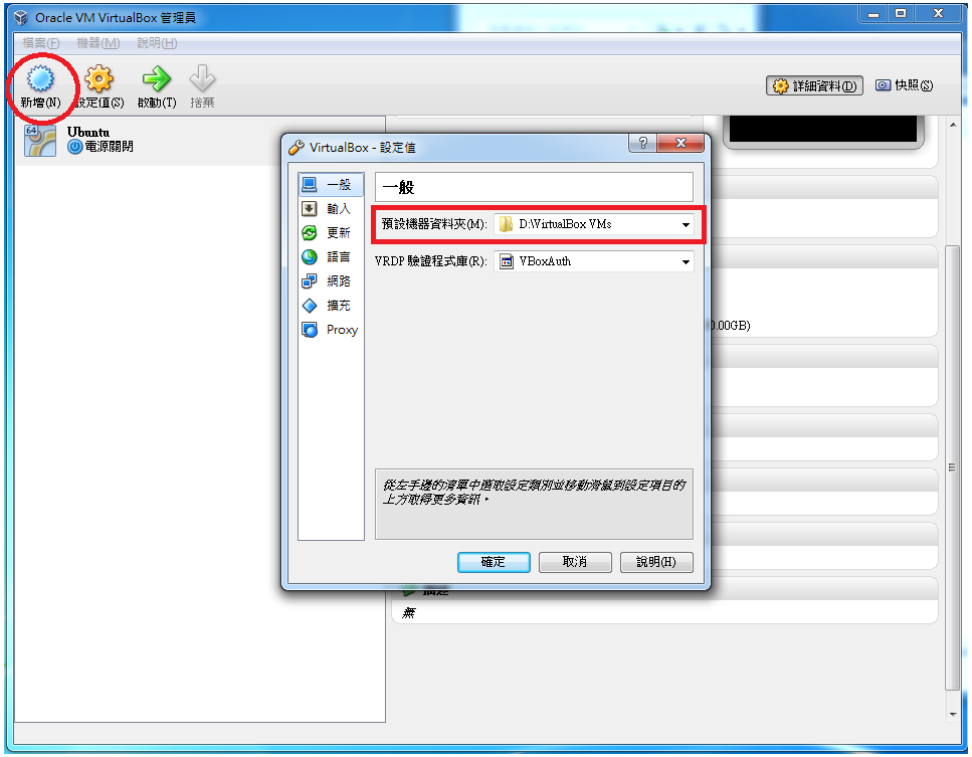


Building Linux Environment

- Get Virtualbox
  - https://www.virtualbox.org/wiki/Downloads
- Get CentOS image
  - http://61.164.110.188:82/Centos/
  - We use CentOS-5.8-x86_64-bin-DVD(1&2) as example.
  - You can chose the proper one or other Linux distribution such as Ubuntu.

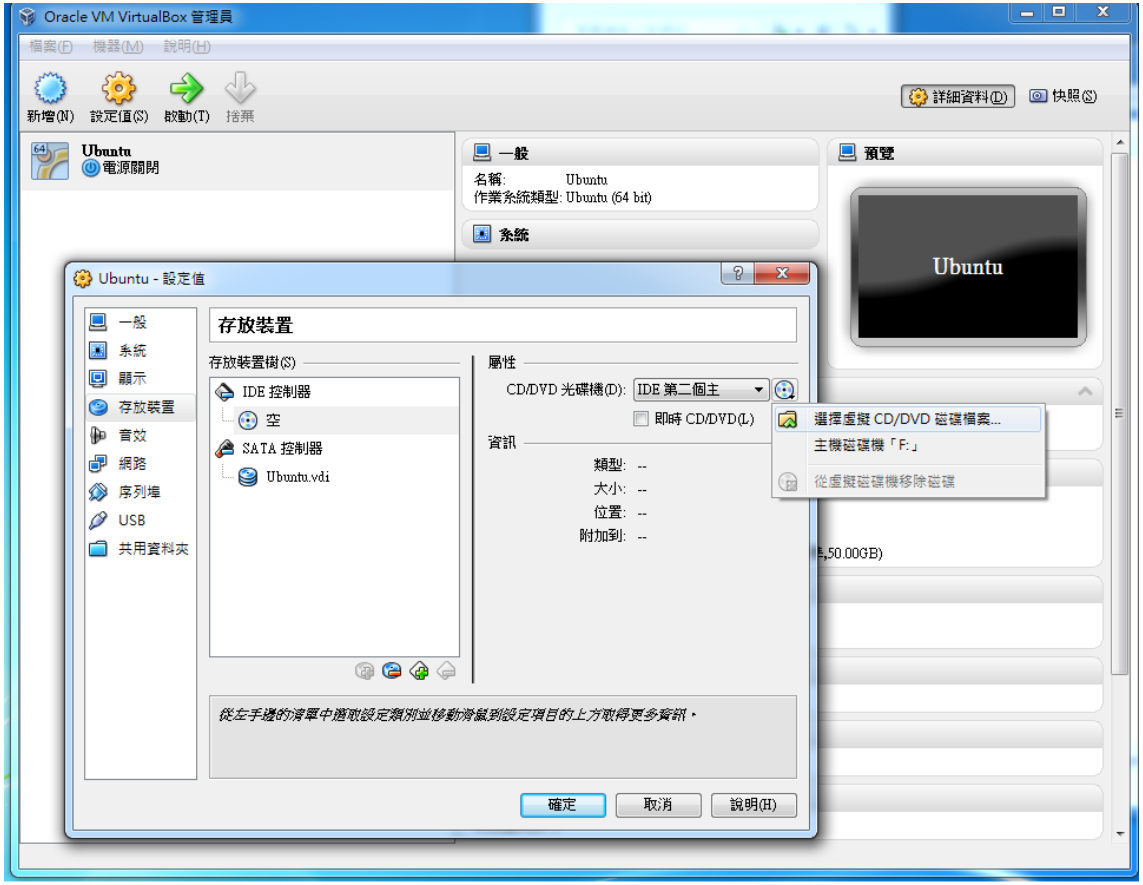
Building Virtual Machine

- Install Virtualbox
- Build the virtual machine
  - 點選新增後按照步驟執行(如果是採用centos，版本可選Red Hat 64)
  - 虛擬機器資料夾可在”檔案->喜好”設定內修改



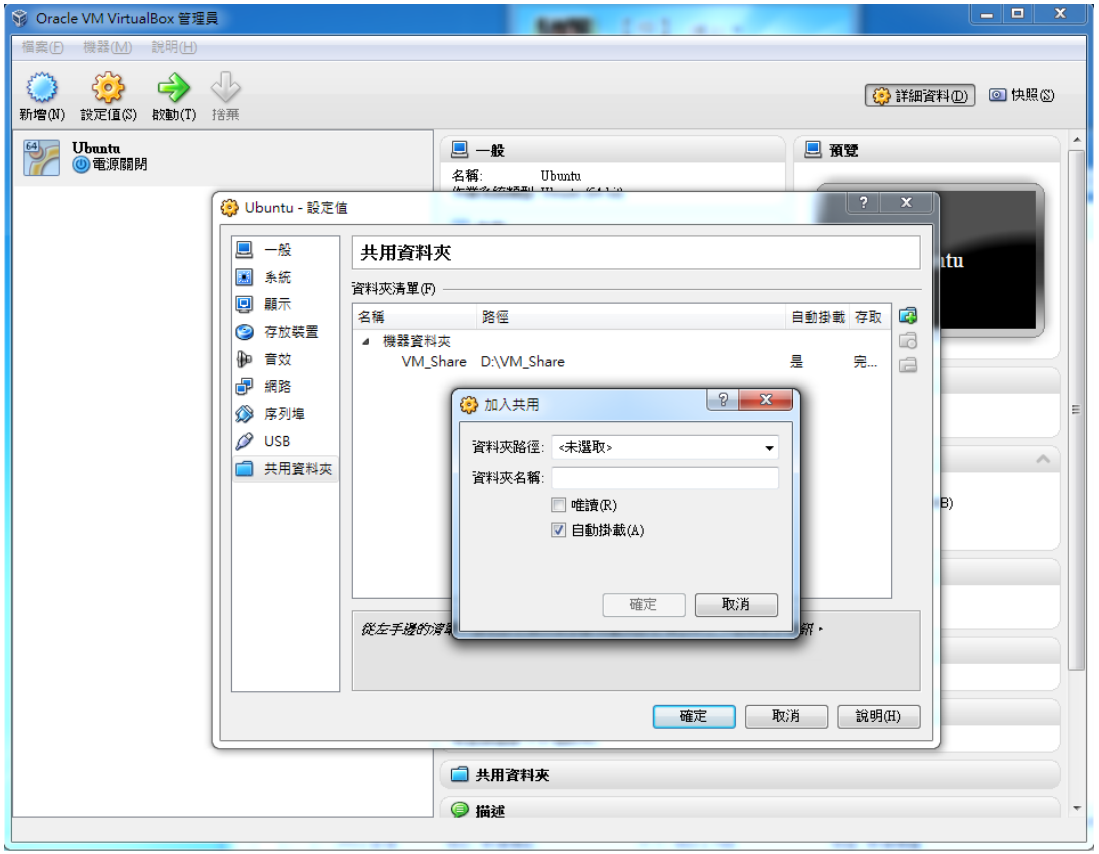
Setting Virtual Machine

- 點選系統->存放裝置 選擇光碟機圖示
  - 選擇虛擬光碟機並且選取剛剛所下載的CentOS映像檔



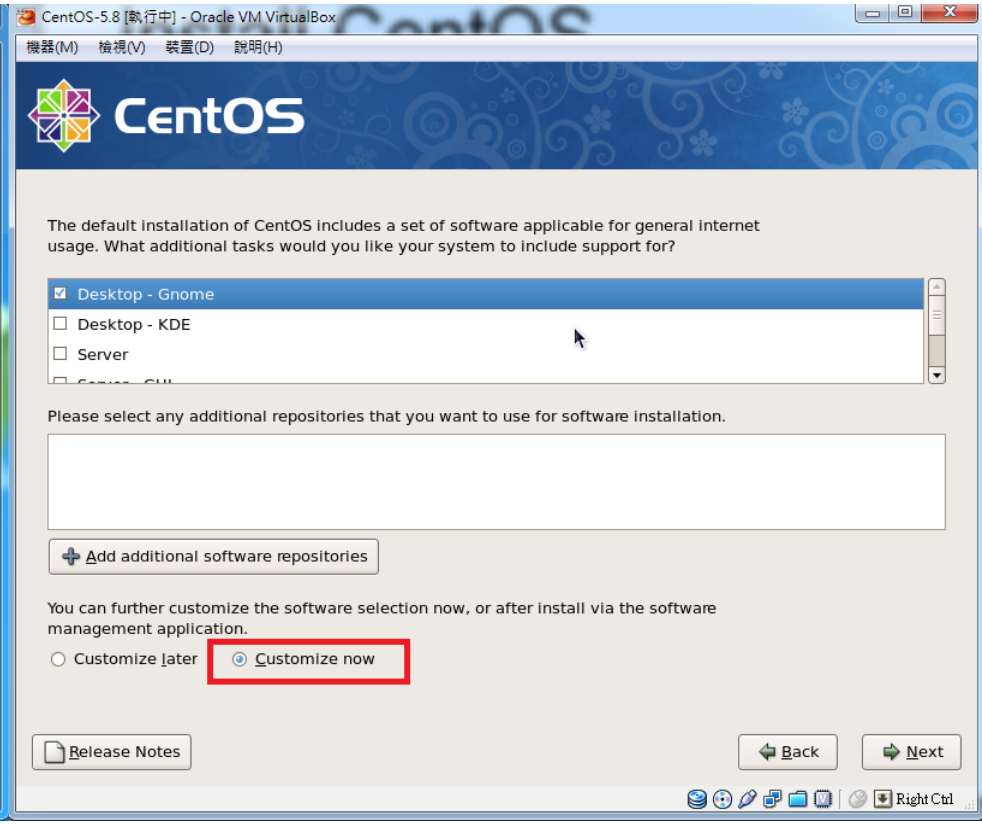
Setting Virtual Machine (Cont.)

- 選擇＂共用資料夾＂ 並且選擇掛載之資料夾，勾取自動掛載。之後此虛擬機器一開機即可透過此資料夾與host OS共享資料



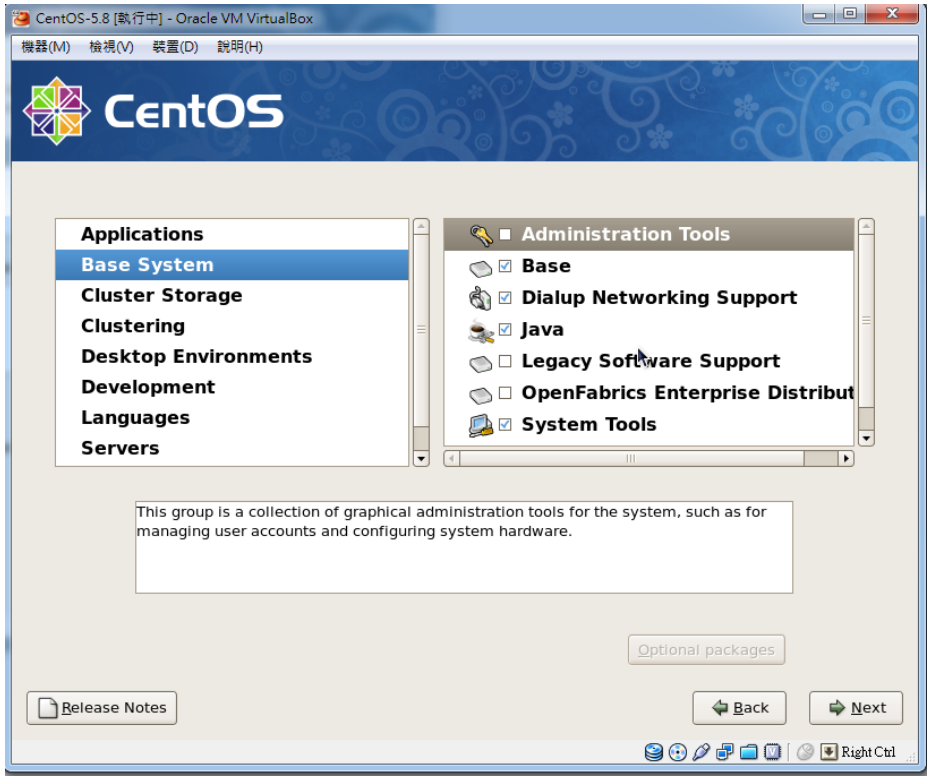
Install CentOS

- 執行啟動就會自動進入CentOS安裝程式



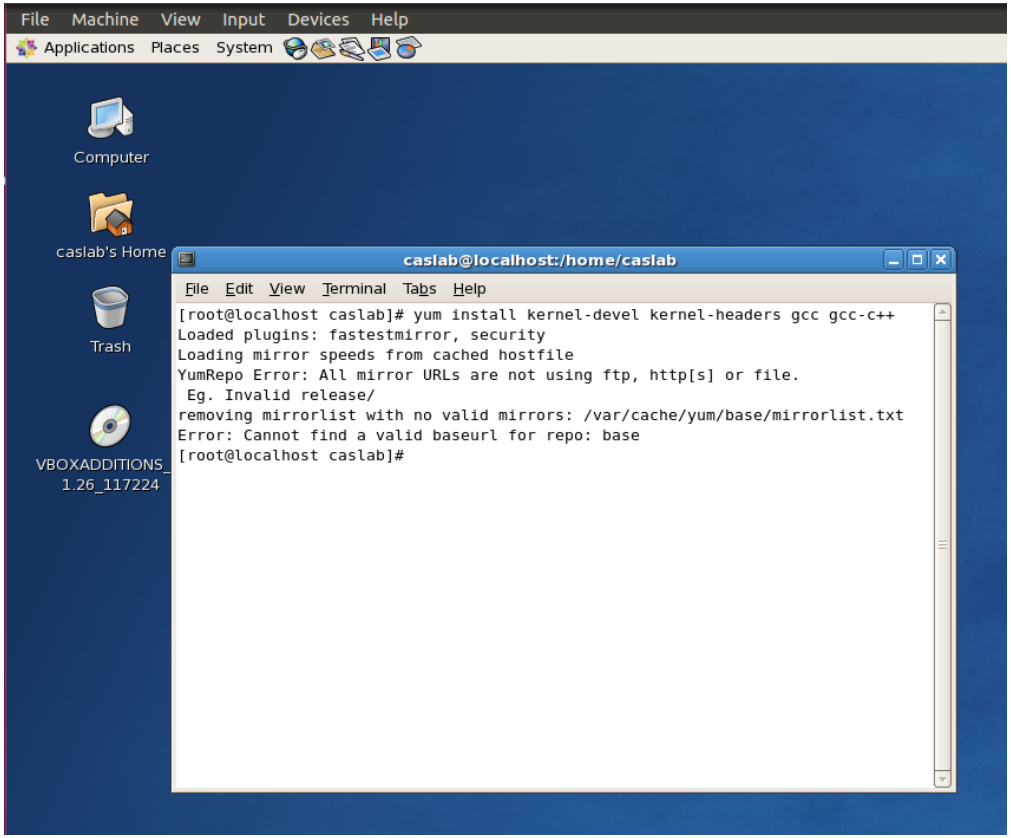
Install CentOS

- Base System選取JAVA和System Tools
- Development除了KDE不選，其他全選
- CentOS可用root登入，但之後的範例都是以使用者登入為預設狀況(使用者模式下的終端機以su指令可進入root模式)



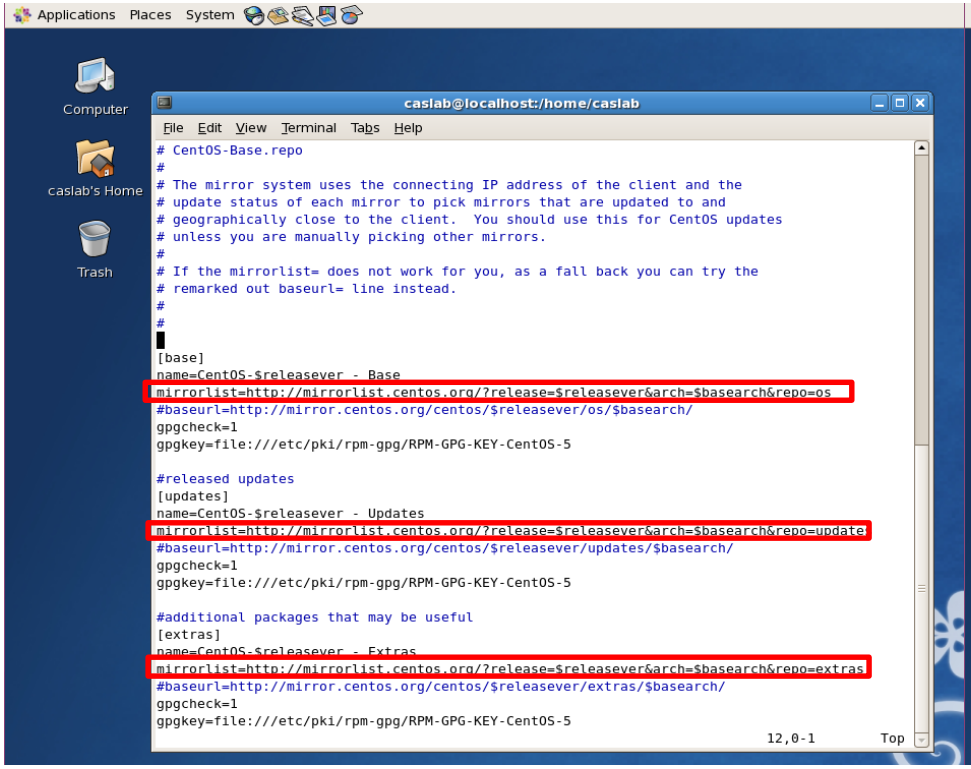
Install CentOS

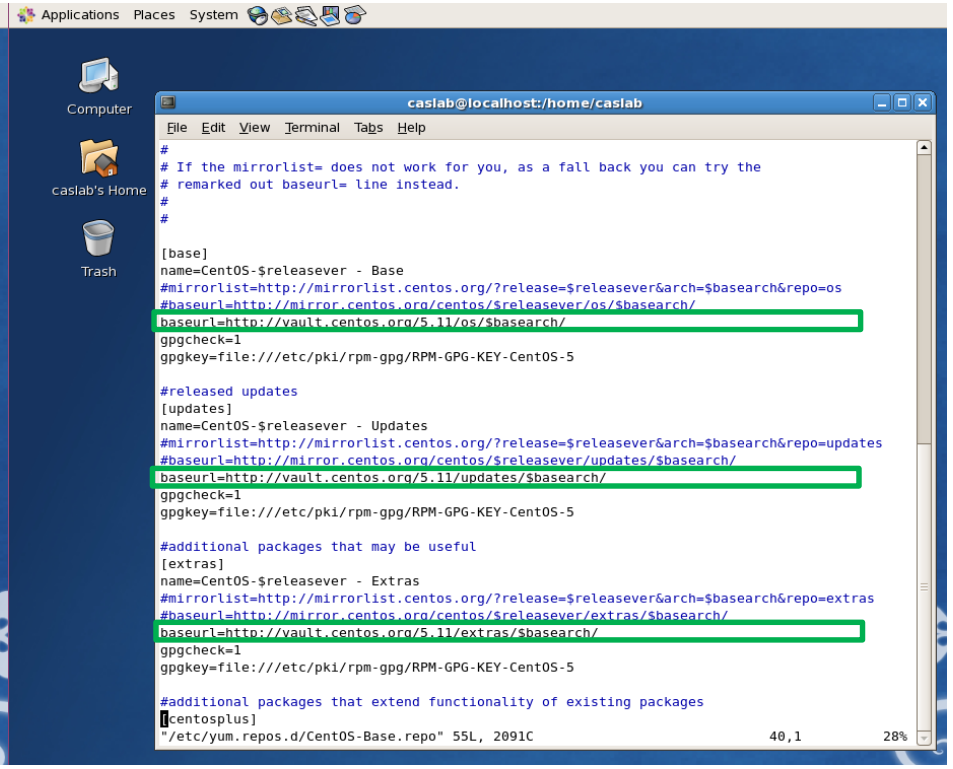
- 由於官方停止支援CentOS-5.8
- 修改CentOS-Base.repo



Install CentOS

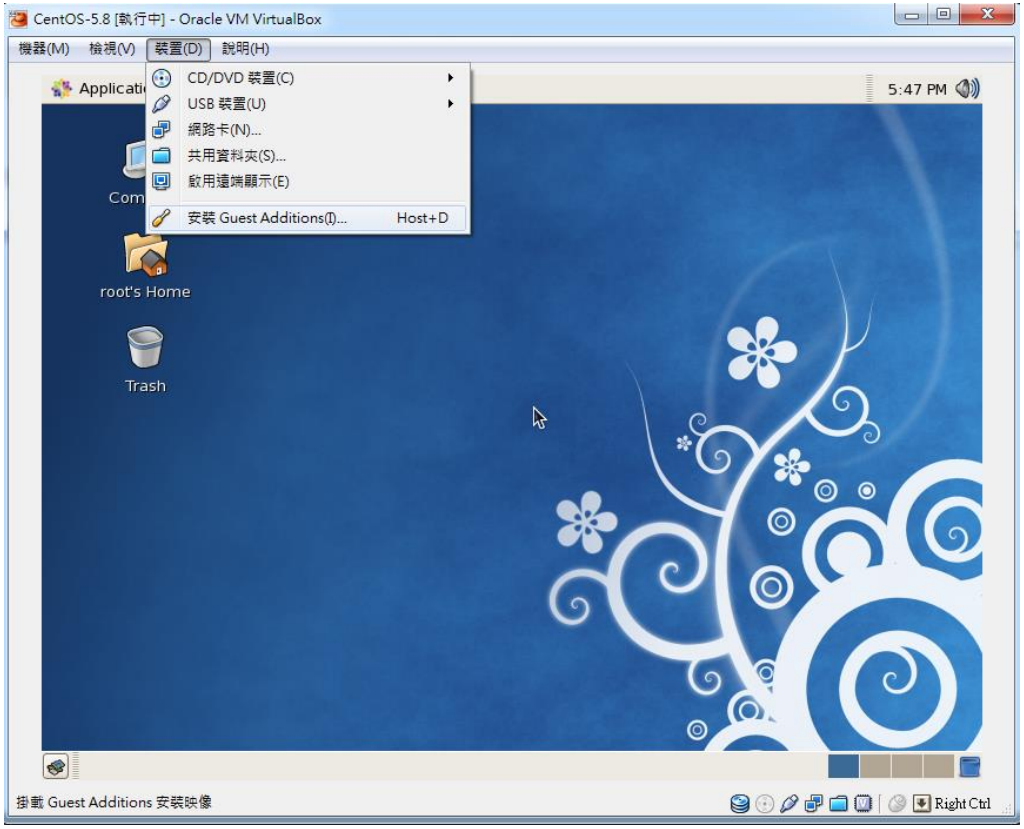
- 輸入指令 vim /etc/yum.repos.d/CentOS-Base.repo
- 將紅框內的部分註解掉，加上綠框內的部分





Install CentOS

- 安裝必要套件: yum install kernel-devel kernel-headers gcc gcc-c++
- 再以”#yum upgrade”來更新CentOS所有套件
- 更新完且重開機後，為了讓CentOS與Host OS互動性更好，我們必須安裝Guest Additions(執行VBoxLinuxAdditions.run後重開機)



Install QEMU

- Get QEMU
  - http://wiki.qemu.org/download/qemu-0.15.1.tar.gz
- 建立工作資料夾，並將qemu-0.15.1.tar.gz複製到workstation進行解壓縮
  - mkdir workstation
  - cp ./qemu-0.15.1.tar.gz ./workstation
  - cd workstation/
  - tar –zxvf qemu-0.15.1.tar.gz

Install QEMU(Cont.)

- 設定QEMU的模擬對象以及所要安裝的目錄
  - cd qemu-0.15.1
  - ./configure --target-list=arm-softmmu,arm-linux-user --prefix=/home/{your username}/qemu-bin
  - --target-list為我們所希望模擬的目標選項。這裡的arm-softmmu代表我們想要QEMU針對整個平台(包含CPU與周邊硬體)做模擬(如realview versatile family)，而arm-linux-user則是只做CPU指令集架構的轉換模擬。兩者的CPU指令集架構都是透過Binary Transation完成。
  - --prefix為安裝目的資料夾選項
  - make && make install
- 編譯過程如果因為glibc版本過舊會發生compile error
- 修改qemu-0.15.1/qemu-ga.c 第149行
- if (g_strcmp0(domain, “syslog”) == 0)必須改成 if (domain && strcmp(domain, "syslog") == 0)

Install QEMU(Cont.)

- 安裝完成後，/home/{your username}/qemu-bin/bin中有所有的QEMU執行檔
  - qemu-arm 為 arm-linux-user產物
  - qemu-system-arm 為 arm-softmmu 產物
  - 之後的實驗，QEMU的部分我們將透過這兩個執行檔完成。

Cross Compiler

- A cross compiler can build the executable code for the target platform other than the one on which the compiler is run.
- ARM cross compiler
  - ARM-elf-gcc, ARM-linux-gcc
  - RVDS, ADS
  - Build ARM executable code on X86/AMD64 machine

Setting ARM Linux GCC

- Get ARM-Linux-gcc (Mentor)
  - https://sourcery.mentor.com/sgpp/lite/arm/portal/subscription3057
- 我們以2011.03-41版本(Lite Edition)作範例
  - GCC 4.5.2
  - Glibc 2.13
  - Linux Kernel 2.6.38
- 如果是安裝Ubuntu 64bit版本，則需要另外安裝IA32函示庫。
  - sudo apt-get install ia32-libs

Setting ARM Linux GCC (Cont.)

- 解壓縮並且將執行檔宣告至預設執行區(壓縮檔先行放在家目錄)
  - tar –xvf arm-2011.03-41-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2
  - export PATH= “/home/{your username}/workstation/arm-2011.03/bin:$PATH”
  - 請注意: export此行指令，只要terminal重開就必須再次執行。可用 vim ~/.bashrc 加入此行指令讓terminal開啟時自動執行此行指令。
- 執行” arm-none-linux-gnueabi-gcc –v” 觀察是否設定完成。

Compile C code

- 利用編輯器完成一段簡短的C程式碼並且使用arm-linux-gcc編譯。
  - cd ~ && mkdir test
  - 在test資料夾中新增test.c 並寫幾行簡單程式
  - arm-none-linux-gnueabi-gcc test.c -o test.o
  - arm-none-linux-gnueabi-objdump –xD test.o > dump.txt
  - 從dump.txt可以看出透過cross compiler我們編出ARM code了!

test.c

```
1  #include <stdio.h>
2
3  int main(){
4
5          printf("COURSE : Computer Architecture\n");
6
7          return 0;
8
9  }
10
```

dump.txt

```
1
2  test.o:     file format elf32-littlearm
3  test.o
4  architecture: arm, flags 0x00000112:
5  EXEC_P, HAS_SYMS, D_PAGED
6  start address 0x00008380
7
```

Execute ARM code

- 最後透過我們一開始所編譯出的qemu-arm來執行test.o
  - 將所在目錄移至test資料夾
  - ~/qemu-bin/bin/qemu-arm –L /home/{your username}/workstation/arm-2011.03/arm-none-linux-gnueabi/libc test.o
  - 執行檔qemu-arm –L 後面所接的是cross compiler所提供的函示庫(Library)，最後的test.o則是我們剛才所編譯出來的檔案。當然您也可以將test.o與qemu-arm搬移出來直接./qemu-arm –L {arm-library位置} test.o
- 執行結果會直接秀在terminal上