嵌入式系統設計概論與實作

曾煜棋、吳昆儒

National Yang Ming Chiao Tung University

Last week

- 嵌入式應用: 語音助理
  1. Mel-Frequency Cepstral Coefficients
  2. Speech to text (STT)
  3. Text to speech (TTS)

- 語音識別 (Speech recognition)
- 自動語音辨識 (Automatic Speech Recognition, ASR)
- 電腦語音識別 (Computer Speech Recognition)
- 語音轉文字識別 (Speech To Text, STT)
- 自然語言處理 (Natural Language Processing, NLP)
  - 讓電腦擁有理解人類語言的能力

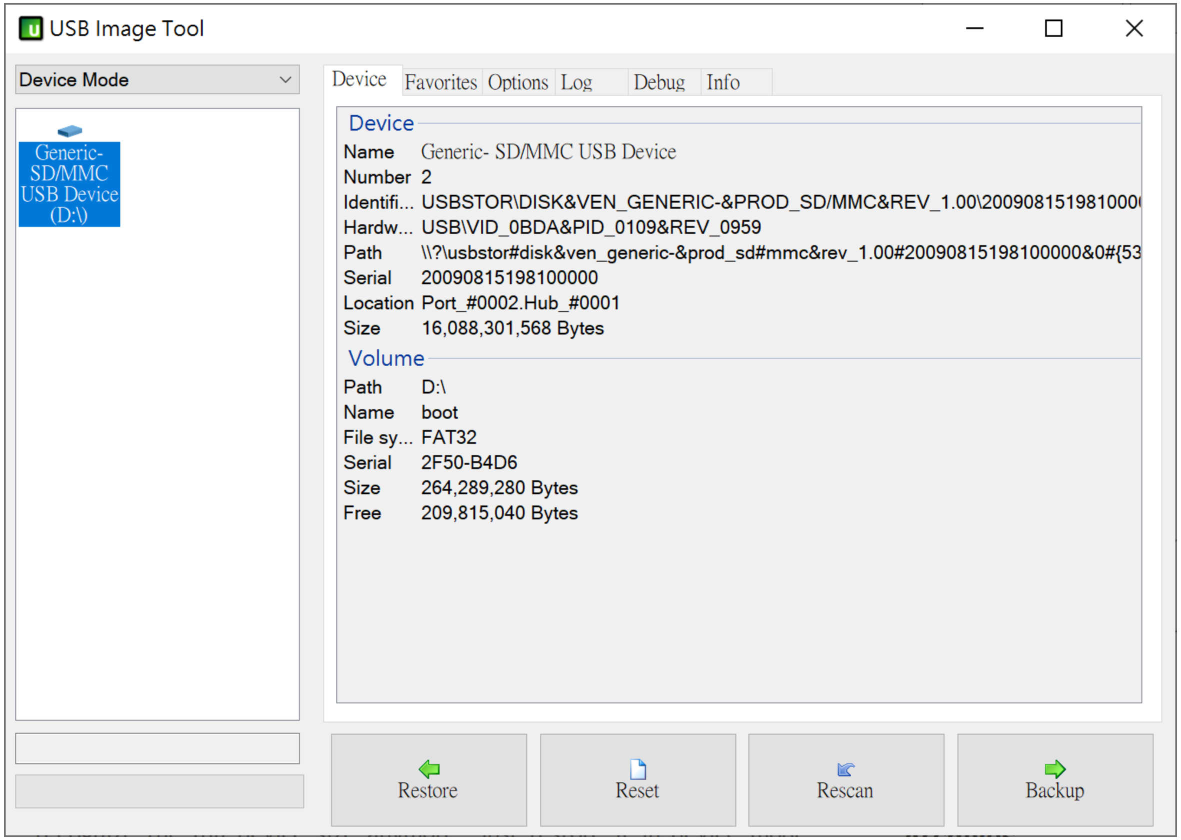
This week

- 嵌入式系統:
  - A. cross compile
    1. prepare a Linux system: Virtualbox + Ubuntu 18 (64bit)
    2. download toolchain for PI
    3. compile code on Virtualbox, then execute on PI
  - B. build kernel
    1. Download linux kernel
    2. Configure kernel
    3. Build, then copy to your PI

Warning: Your PI might be unstable.
Remember to backup you code!!
Remember to backup you code!!
Remember to backup you code!!

How to backup full SD card?

- Ex: USB Image Tool 1.81
  - https://www.alexpage.de/usb-image-tool/download/



Device Mode

Generic- SD/MMC USB Device (D:\)

Device

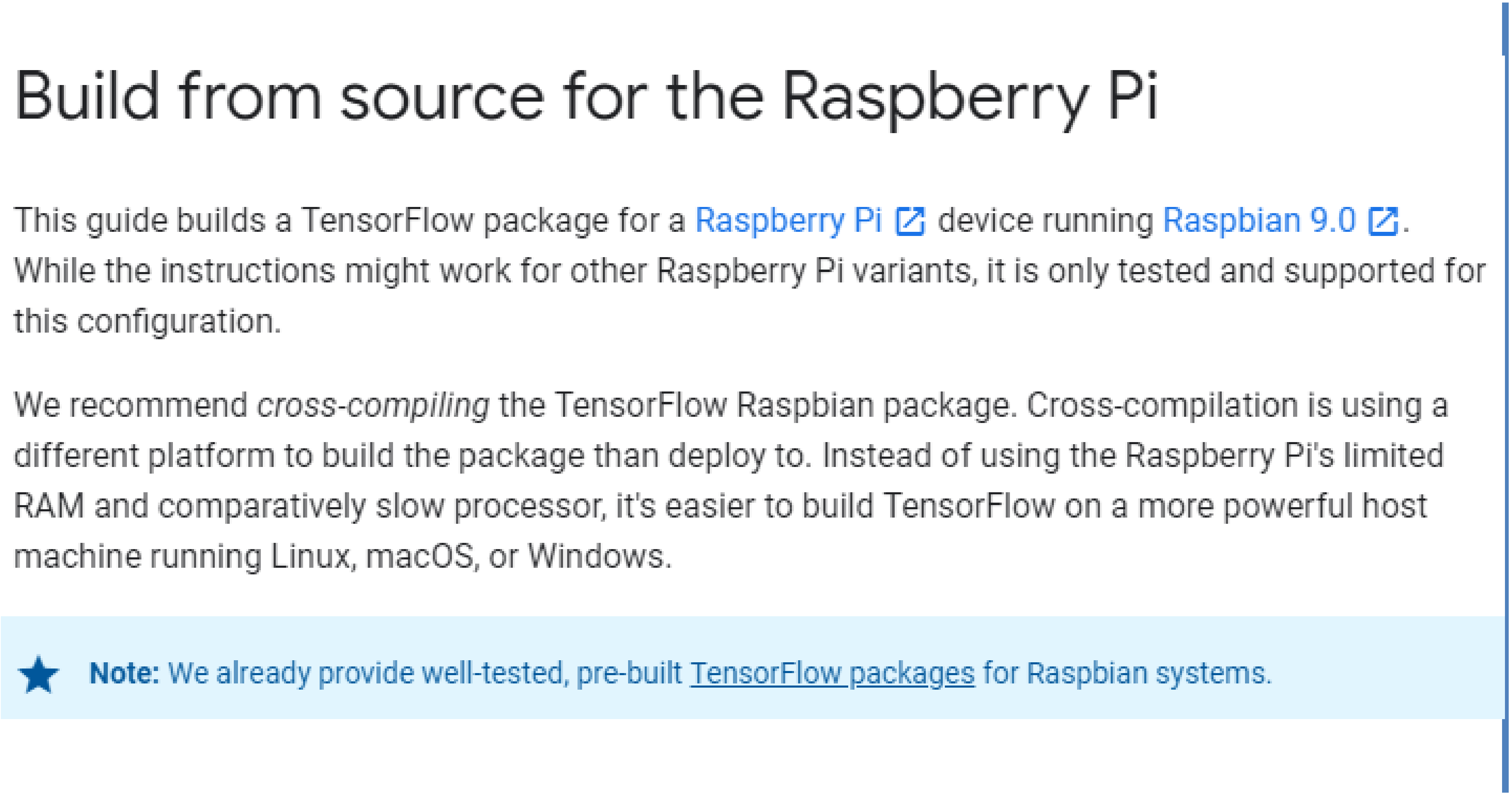
Name Generic- SD/MMC USB Device
Number 2
Identifi... USBSTOR\DISK&VEN_GENERIC-&PROD_SD/MMC&REV_1.00\200908151981000
Hardw... USB\VID_0BDA&PID_0109&REV_0959
Path \\?\usbstor#disk&ven_generic-&prod_sd#mmc&rev_1.00#20090815198100000&0#{53
Serial 20090815198100000
Location Port_#0002.Hub_#0001
Size 16,088,301,568 Bytes

Volume

Path D:\
Name boot
File sy... FAT32
Serial 2F50-B4D6
Size 264,289,280 Bytes
Free 209,815,040 Bytes

https://www.tensorflow.org/install/source_rpi

Ex: Build Tensorflow



Build from source for the Raspberry Pi

This guide builds a TensorFlow package for a Raspberry Pi device running Raspbian 9.0. While the instructions might work for other Raspberry Pi variants, it is only tested and supported for this configuration.

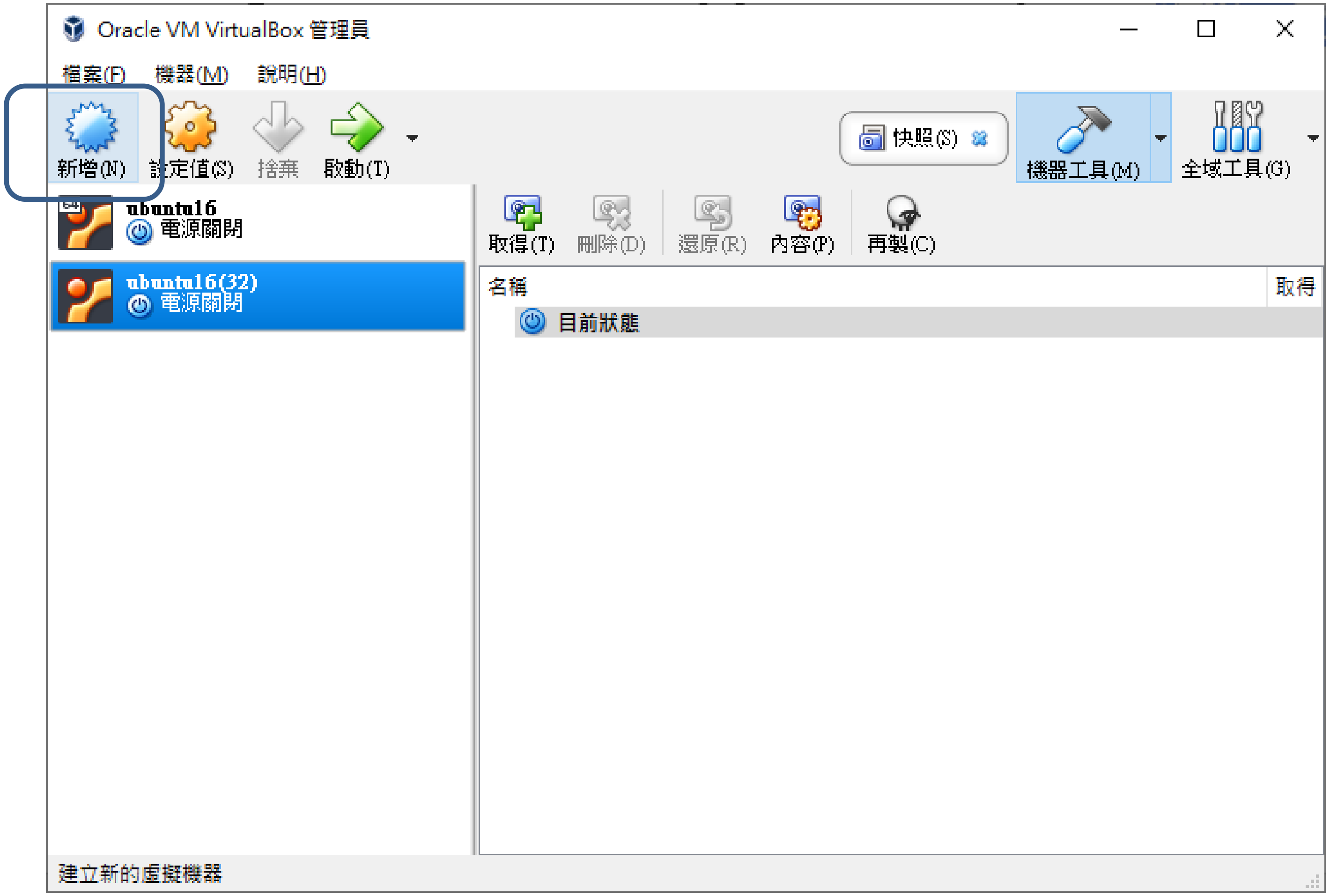
We recommend *cross-compiling* the TensorFlow Raspbian package. Cross-compilation is using a different platform to build the package than deploy to. Instead of using the Raspberry Pi's limited RAM and comparatively slow processor, it's easier to build TensorFlow on a more powerful host machine running Linux, macOS, or Windows.

★ **Note:** We already provide well-tested, pre-built TensorFlow packages for Raspbian systems.

Outline

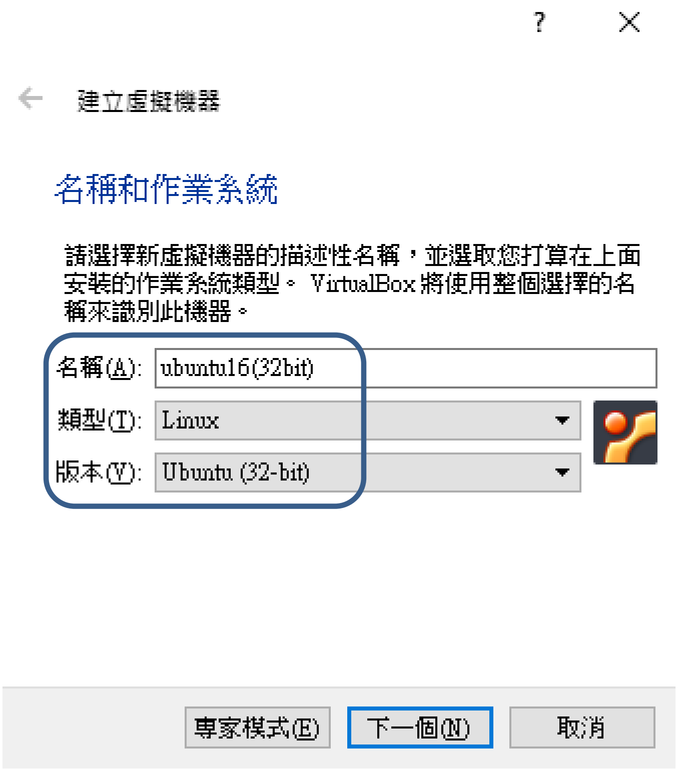
- 嵌入式系統:
  - A. cross compile
    1. prepare a Linux system: Virtualbox + Ubuntu 18 (64bit)
    2. download toolchain for PI
    3. compile code on Virtualbox, then execute on PI
  - B. build kernel
    1. Download linux kernel
    2. Configure kernel
    3. Build, then copy to your PI

1. Prepare a Linux system



建立新的虛擬機器

1. Prepare a Linux system



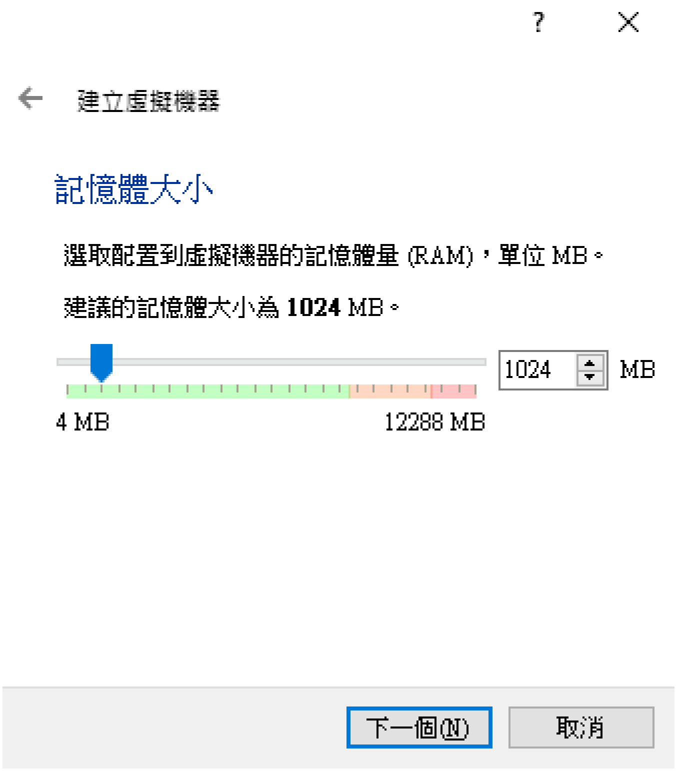
名稱和作業系統

請選擇新虛擬機器的描述性名稱，並選取您打算在上面安裝的作業系統類型。 VirtualBox 將使用整個選擇的名稱來識別此機器。

名稱(A): ubuntu16(32bit)

類型(T): Linux

版本(V): Ubuntu (32-bit)



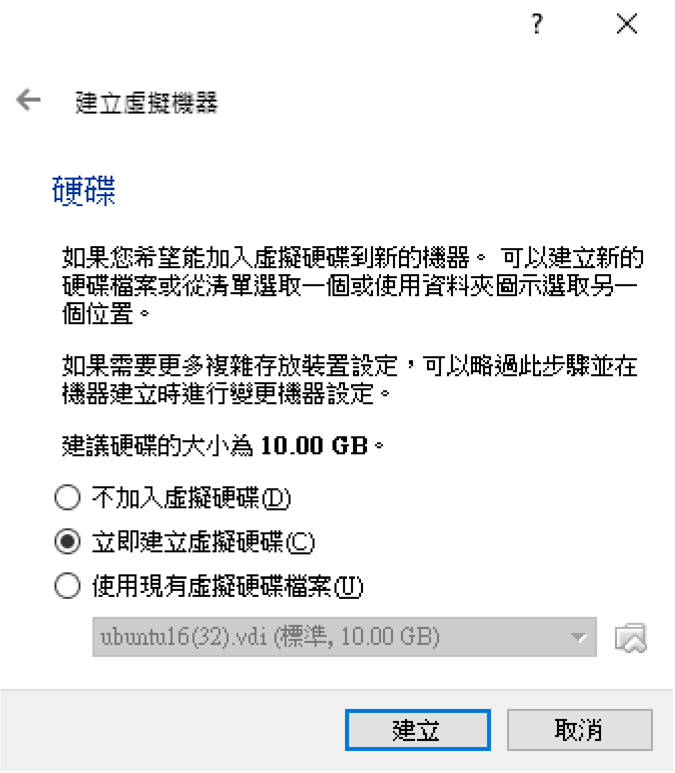
記憶體大小

選取配置到虛擬機器的記憶體量 (RAM)，單位 MB。

建議的記憶體大小為 **1024** MB。

1024 MB

1. Prepare a Linux system



建立虛擬機器

硬碟

如果您希望能加入虛擬硬碟到新的機器。 可以建立新的硬碟檔案或從清單選取一個或使用資料夾圖示選取另一個位置。

如果需要更多複雜存放裝置設定，可以略過此步驟並在機器建立時進行變更機器設定。

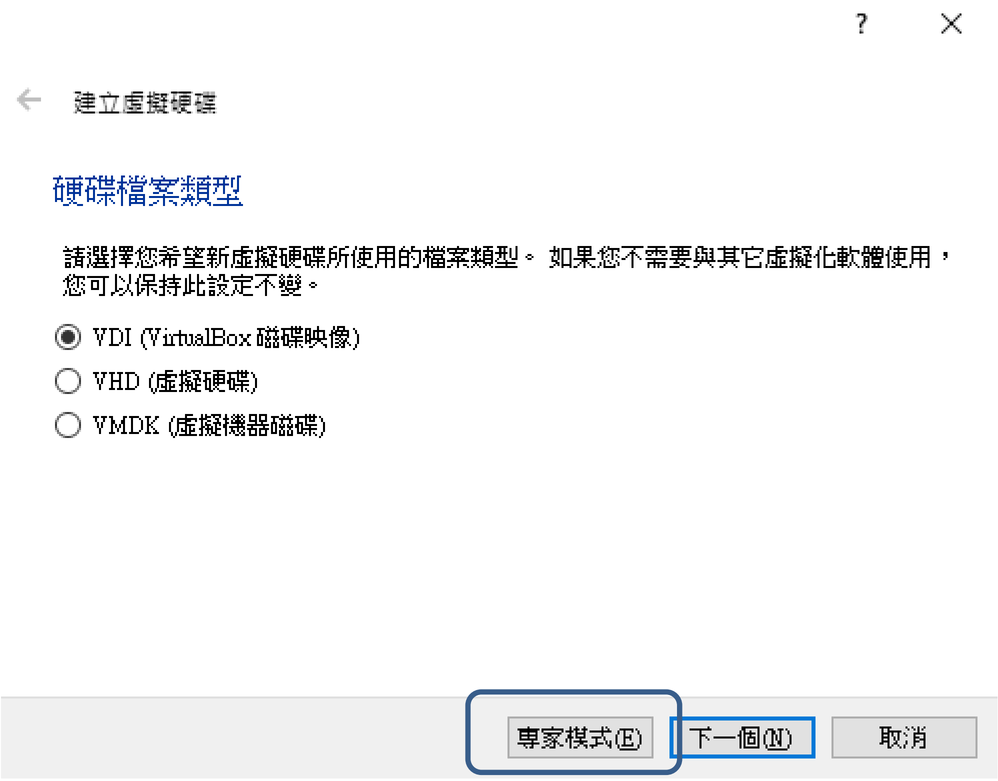
建議硬碟的大小為 **10.00 GB**。

○ 不加入虛擬硬碟(D)

● 立即建立虛擬硬碟(C)

○ 使用現有虛擬硬碟檔案(U)

ubuntu16(32).vdi (標準, 10.00 GB)



建立虛擬硬碟

硬碟檔案類型

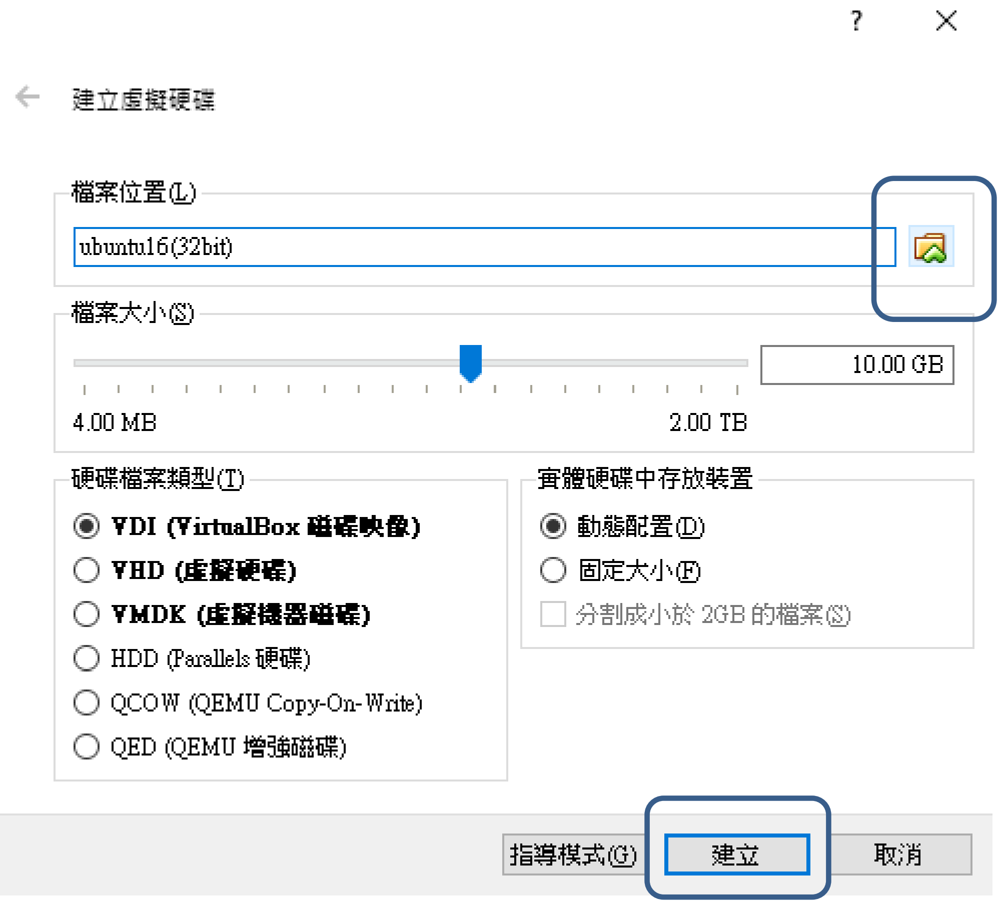
請選擇您希望新虛擬硬碟所使用的檔案類型。 如果您不需要與其它虛擬化軟體使用，您可以保持此設定不變。

● VDI (VirtualBox 磁碟映像)

○ VHD (虛擬硬碟)

○ VMDK (虛擬機器磁碟)

1. Prepare a Linux system



建立虛擬硬碟

檔案位置(L)

ubuntu16(32bit)

檔案大小(S)

10.00 GB

4.00 MB　2.00 TB

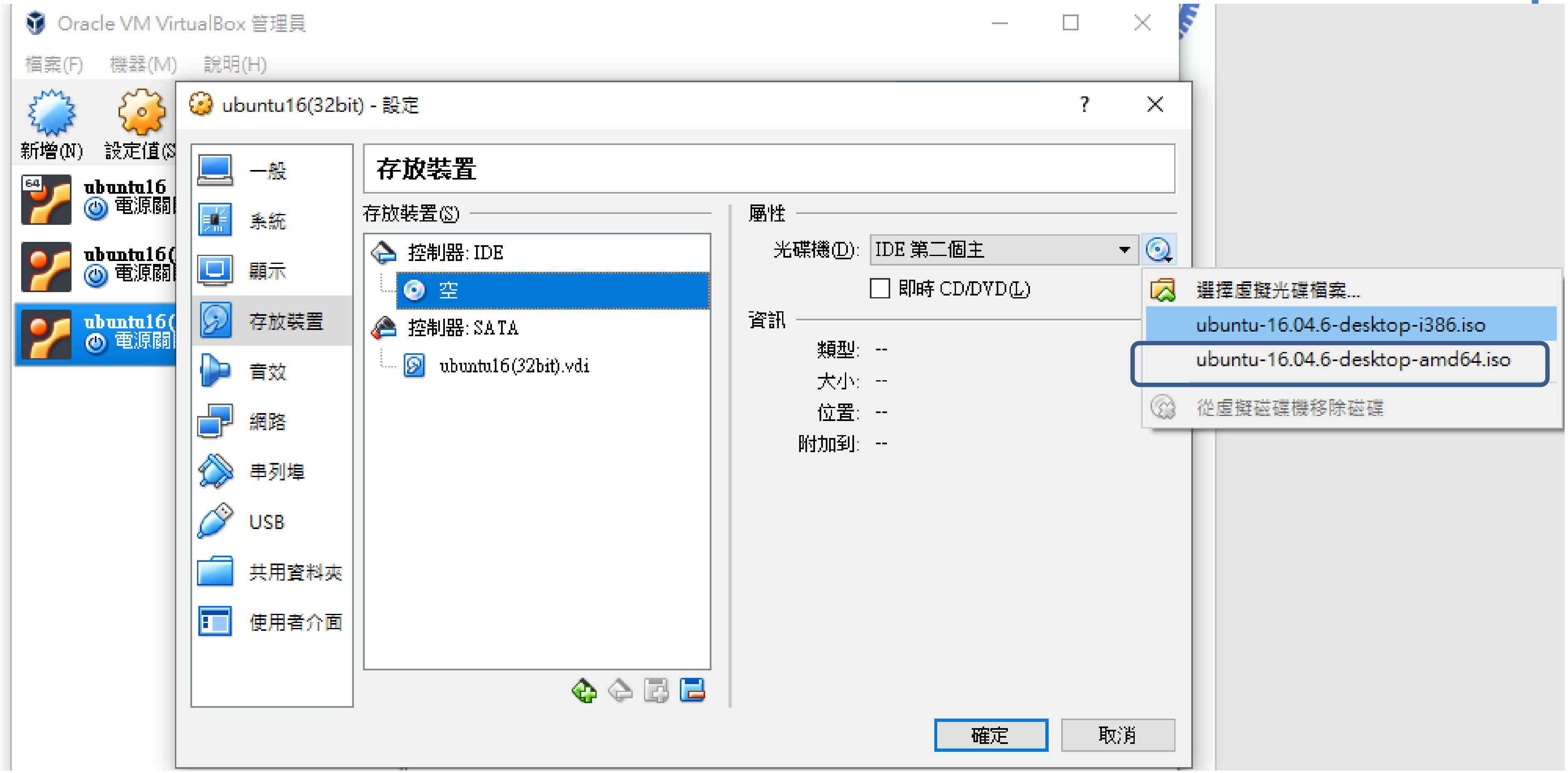
硬碟檔案類型(T)

- ● VDI (VirtualBox 磁碟映像)
- ○ VHD (虛擬硬碟)
- ○ VMDK (虛擬機器磁碟)
- ○ HDD (Parallels 硬碟)
- ○ QCOW (QEMU Copy-On-Write)
- ○ QED (QEMU 增強磁碟)

實體硬碟中存放裝置

- ● 動態配置(D)
- ○ 固定大小(F)
- □ 分割成小於 2GB 的檔案(S)

1. Prepare a Linux system



控制器: IDE
- 空

控制器: SATA
- ubuntu16(32bit).vdi

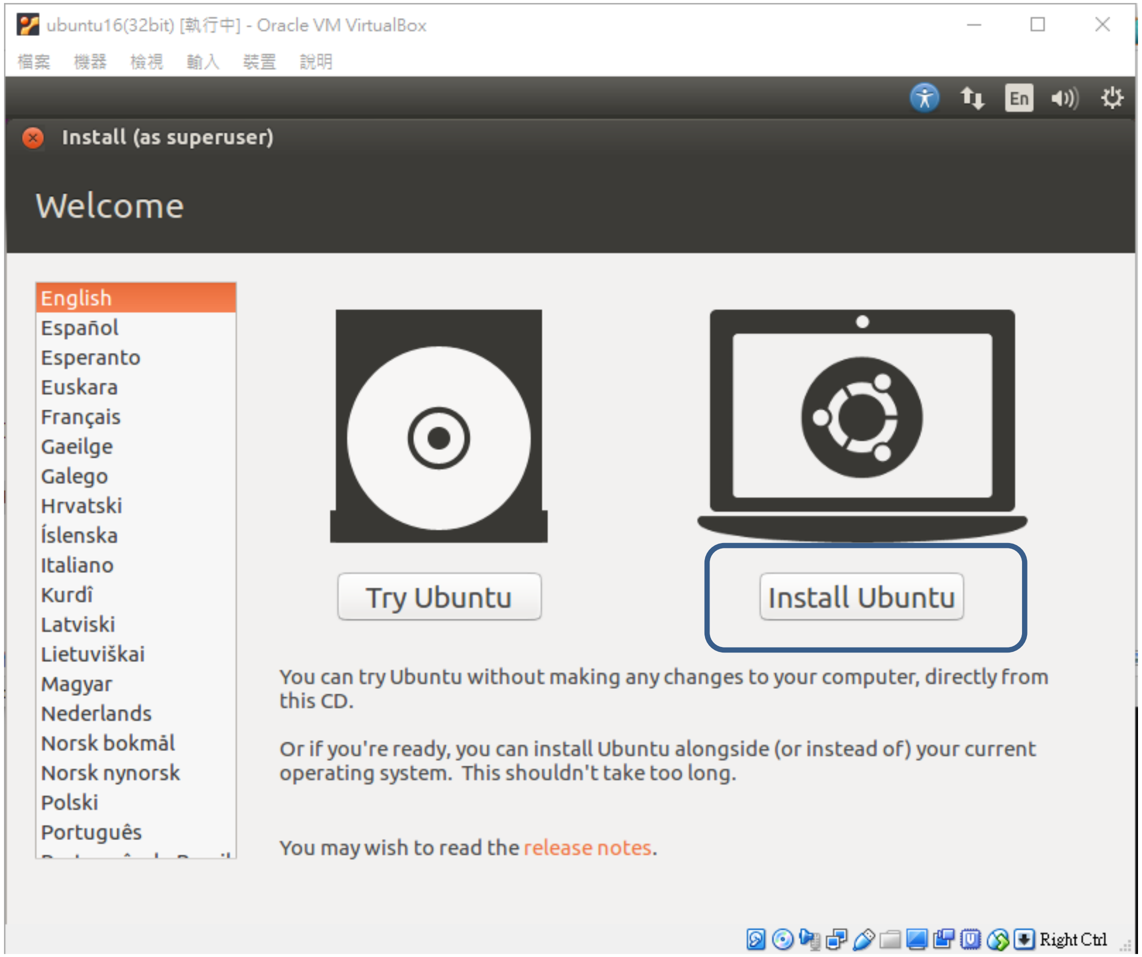
光碟機(D): IDE 第二個主

□ 即時 CD/DVD(L)

類型: --
大小: --
位置: --
附加到: --

http://ubuntu.cs.nctu.edu.tw/ubuntu-release/16.04.6/ubuntu-16.04.6-desktop-amd64.iso

1. Prepare a Linux system



ubuntu16(32bit) [執行中] - Oracle VM VirtualBox

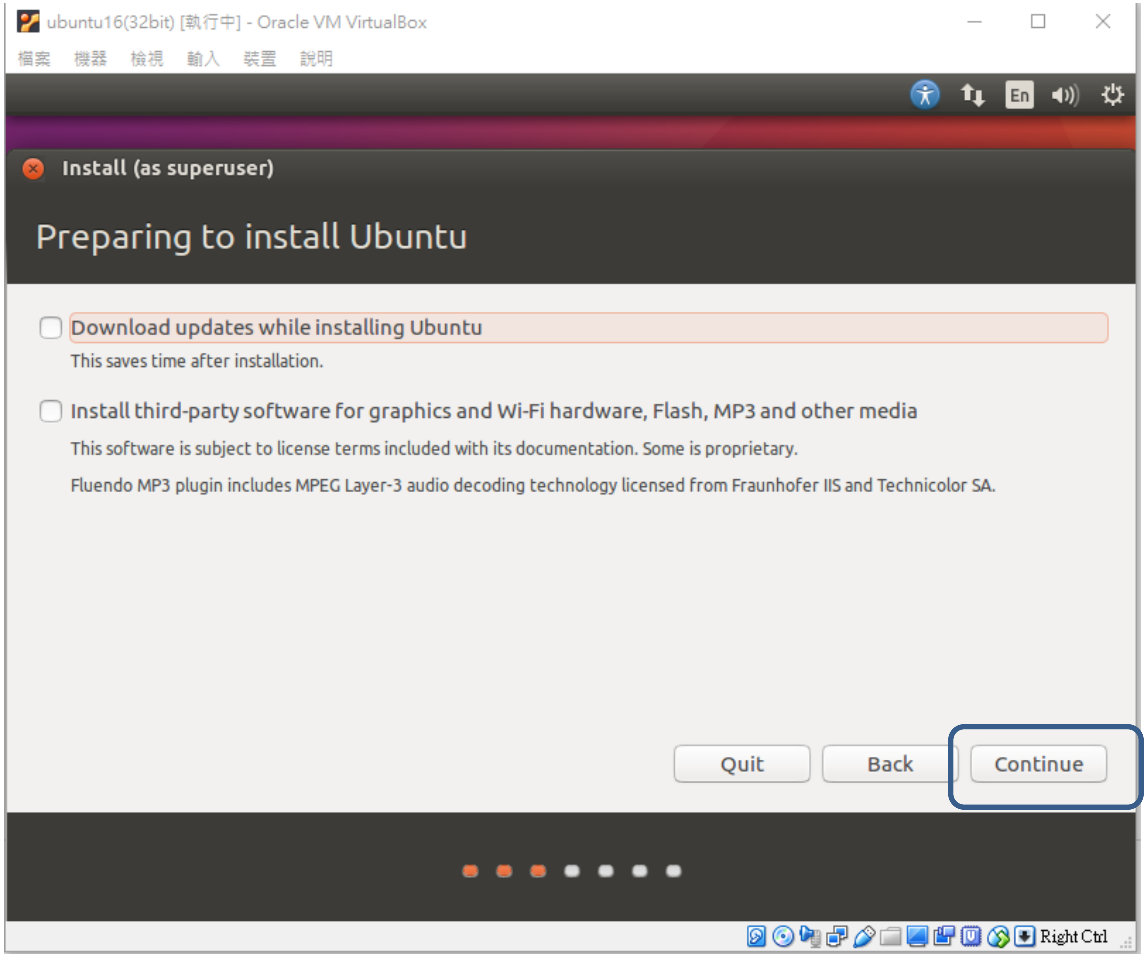
Welcome

You can try Ubuntu without making any changes to your computer, directly from this CD.

Or if you're ready, you can install Ubuntu alongside (or instead of) your current operating system. This shouldn't take too long.

You may wish to read the release notes.

1. Prepare a Linux system



ubuntu16(32bit) [執行中] - Oracle VM VirtualBox
檔案 機器 檢視 輸入 裝置 說明
En
Install (as superuser)

Preparing to install Ubuntu

□ Download updates while installing Ubuntu

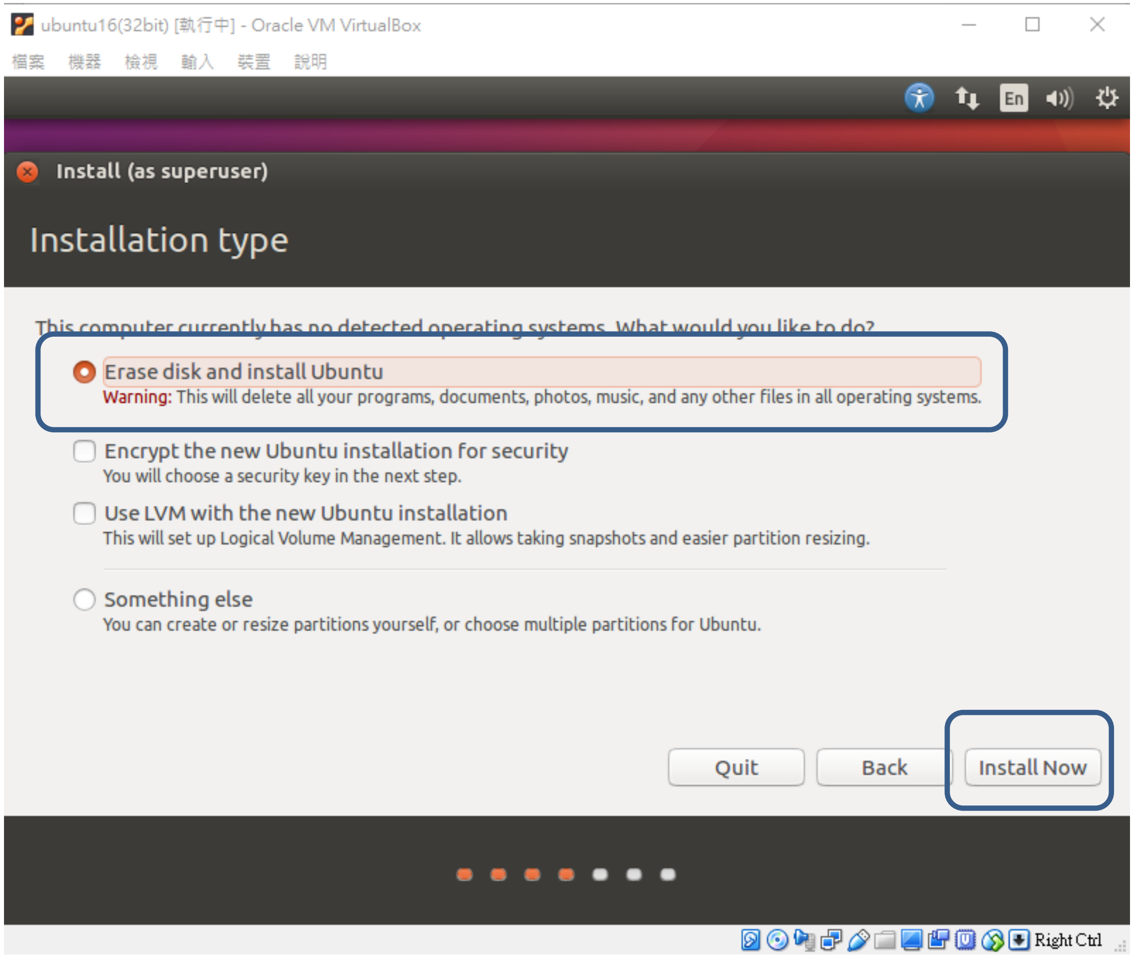
This saves time after installation.

□ Install third-party software for graphics and Wi-Fi hardware, Flash, MP3 and other media

This software is subject to license terms included with its documentation. Some is proprietary.

Fluendo MP3 plugin includes MPEG Layer-3 audio decoding technology licensed from Fraunhofer IIS and Technicolor SA.

1. Prepare a Linux system



ubuntu16(32bit) [執行中] - Oracle VM VirtualBox
檔案 機器 檢視 輸入 裝置 說明
En
Install (as superuser)

Installation type

This computer currently has no detected operating systems. What would you like to do?

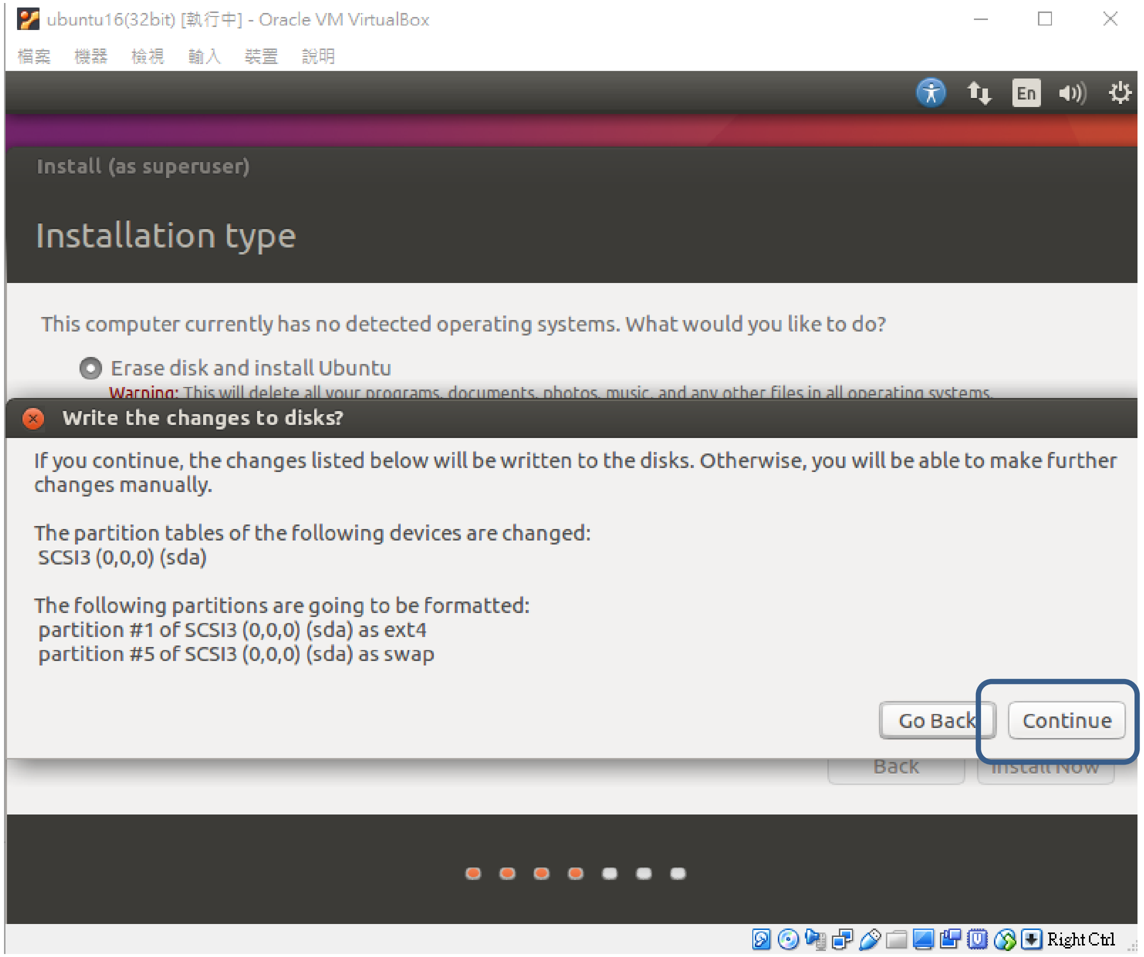
● Erase disk and install Ubuntu
Warning: This will delete all your programs, documents, photos, music, and any other files in all operating systems.

□ Encrypt the new Ubuntu installation for security
You will choose a security key in the next step.

□ Use LVM with the new Ubuntu installation
This will set up Logical Volume Management. It allows taking snapshots and easier partition resizing.

○ Something else
You can create or resize partitions yourself, or choose multiple partitions for Ubuntu.

1. Prepare a Linux system



ubuntu16(32bit) [執行中] - Oracle VM VirtualBox
檔案 機器 檢視 輸入 裝置 說明
En
Install (as superuser)

Installation type

This computer currently has no detected operating systems. What would you like to do?

○ Erase disk and install Ubuntu

Warning: This will delete all your programs, documents, photos, music, and any other files in all operating systems.

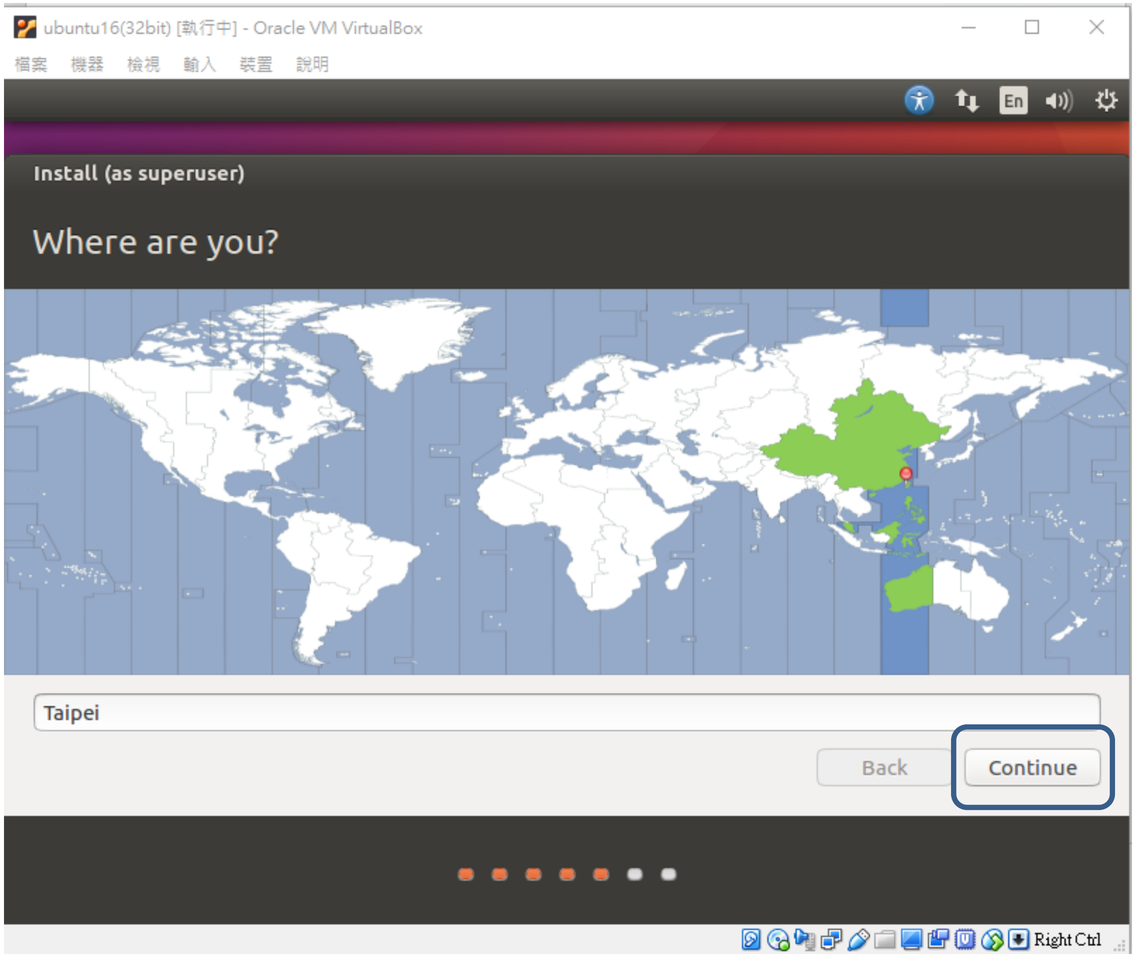
Write the changes to disks?

If you continue, the changes listed below will be written to the disks. Otherwise, you will be able to make further changes manually.

The partition tables of the following devices are changed:
SCSI3 (0,0,0) (sda)

The following partitions are going to be formatted:
partition #1 of SCSI3 (0,0,0) (sda) as ext4
partition #5 of SCSI3 (0,0,0) (sda) as swap

1. Prepare a Linux system



ubuntu16(32bit) [執行中] - Oracle VM VirtualBox

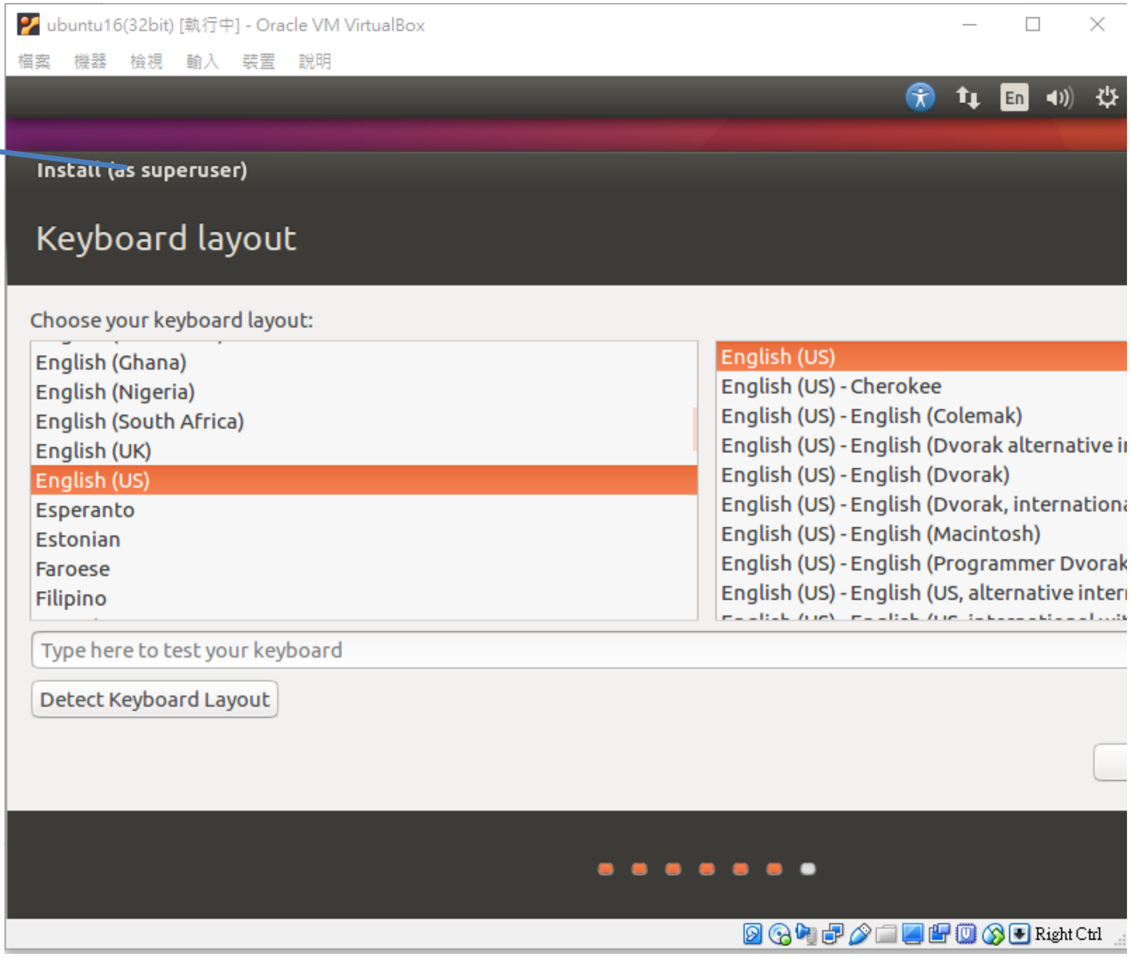
Install (as superuser)

Where are you?

Taipei

1. Prepare a Linux system

可用滑鼠拖曳


ubuntu16(32bit) [執行中] - Oracle VM VirtualBox
檔案 機器 檢視 輸入 裝置 說明
En
Right Ctrl

Install (as superuser)

Keyboard layout

Choose your keyboard layout:

- English (Ghana)
- English (Nigeria)
- English (South Africa)
- English (UK)
- English (US)
- Esperanto
- Estonian
- Faroese
- Filipino

- English (US)
- English (US) - Cherokee
- English (US) - English (Colemak)
- English (US) - English (Dvorak alternative i
- English (US) - English (Dvorak)
- English (US) - English (Dvorak, internationa
- English (US) - English (Macintosh)
- English (US) - English (Programmer Dvorak
- English (US) - English (US, alternative inter

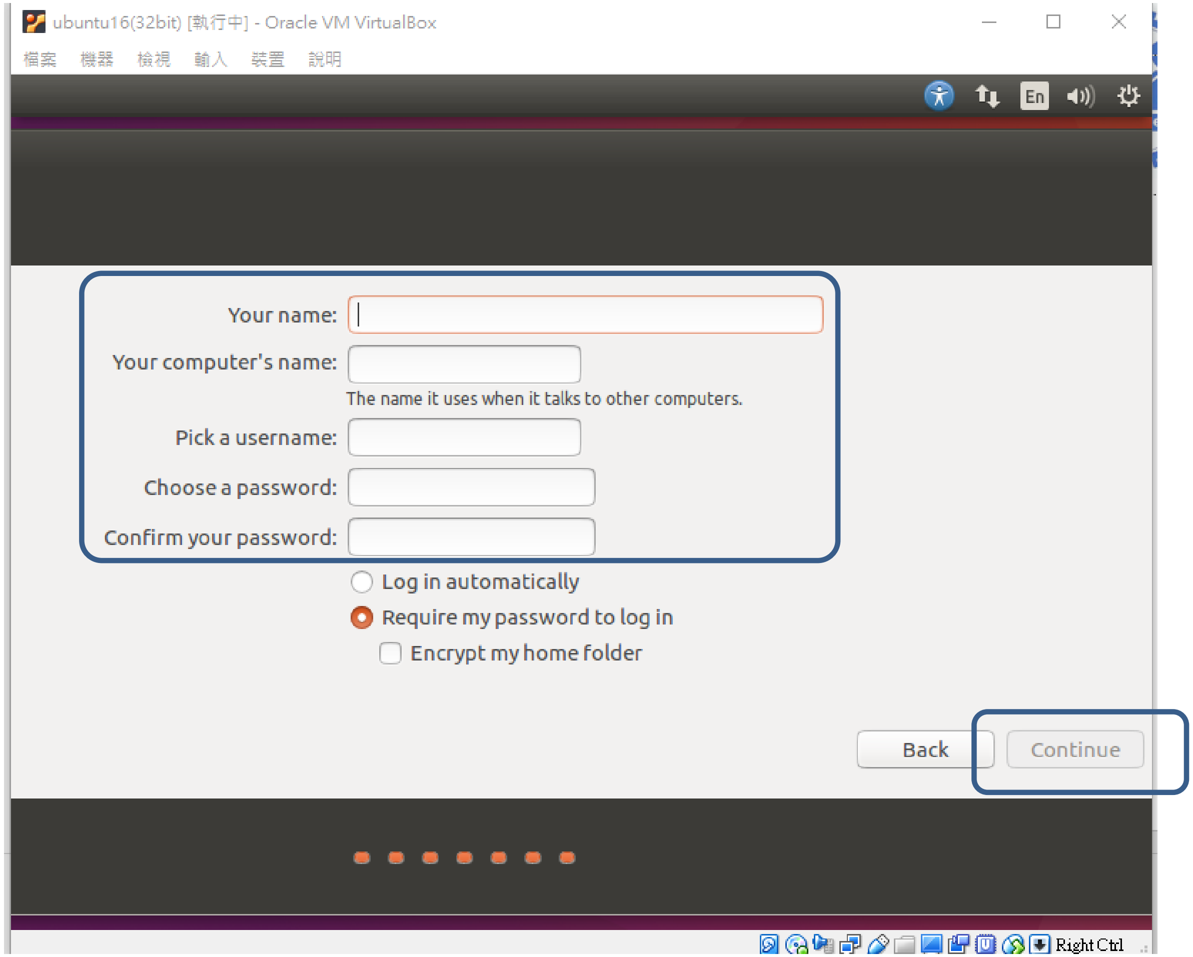
Type here to test your keyboard

Detect Keyboard Layout

continue

按鈕在畫面外

1. Prepare a Linux system



ubuntu16(32bit) [執行中] - Oracle VM VirtualBox

Your name:

Your computer's name:

The name it uses when it talks to other computers.

Pick a username:

Choose a password:

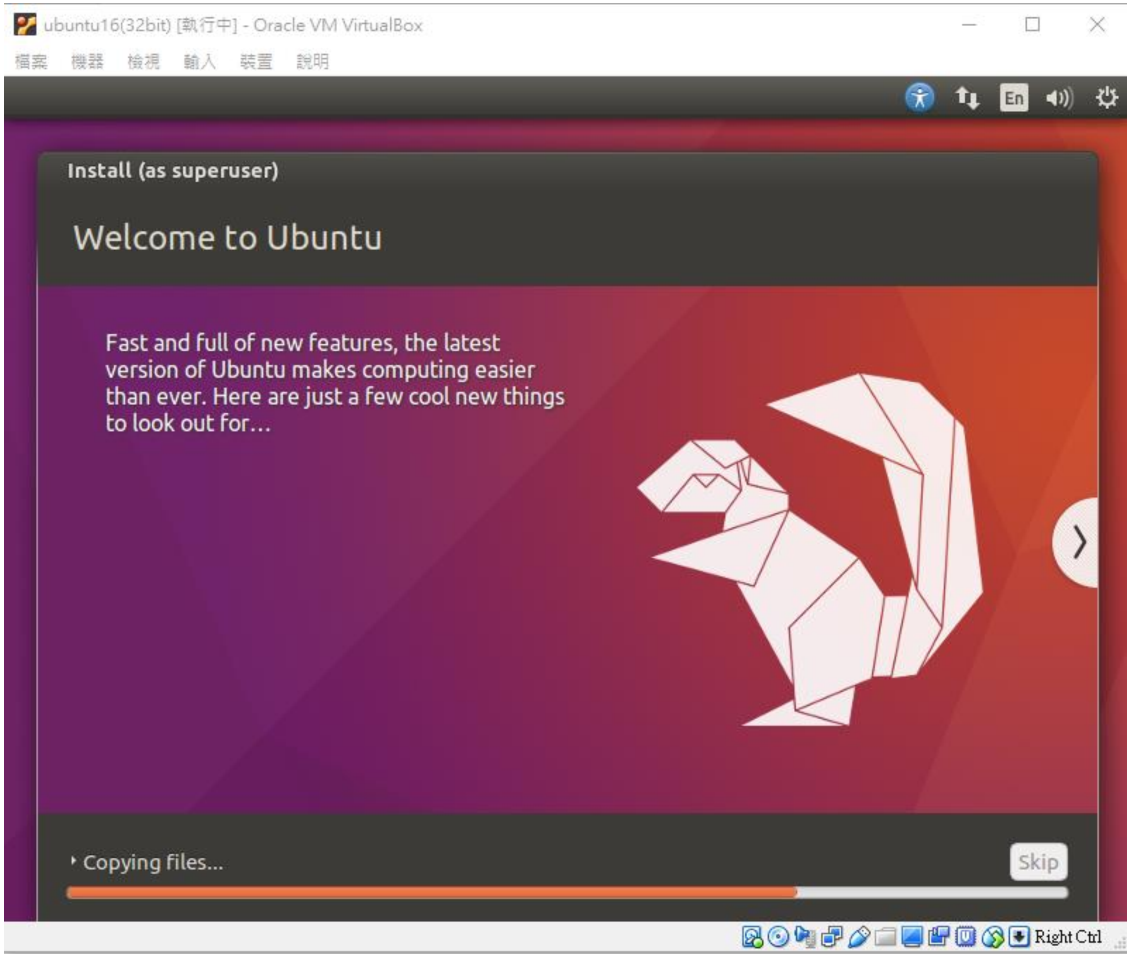
Confirm your password:

○ Log in automatically

● Require my password to log in

□ Encrypt my home folder

1. Prepare a Linux system



ubuntu16(32bit) [執行中] - Oracle VM VirtualBox

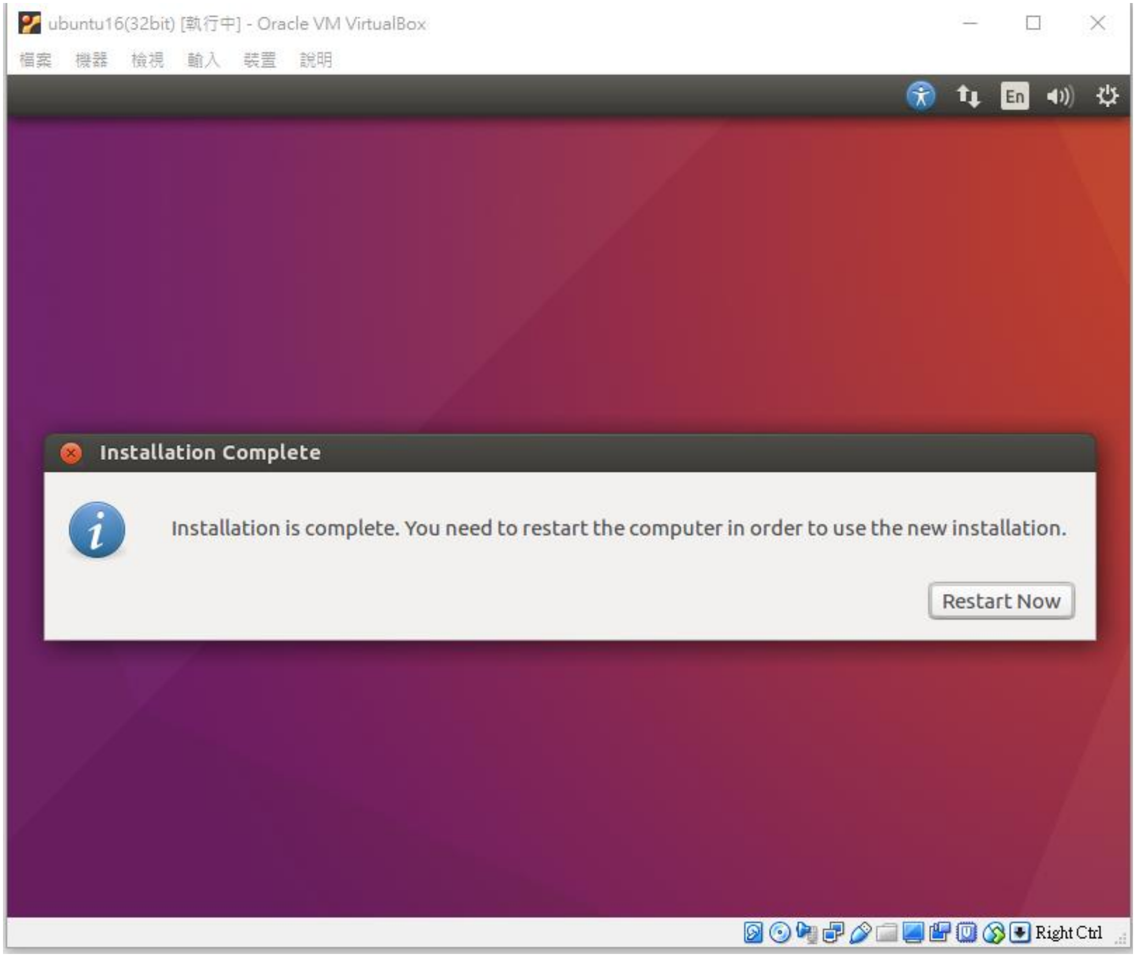
Install (as superuser)

Welcome to Ubuntu

Fast and full of new features, the latest version of Ubuntu makes computing easier than ever. Here are just a few cool new things to look out for…

‣ Copying files...

1. Prepare a Linux system


ubuntu16(32bit) [執行中] - Oracle VM VirtualBox
檔案 機器 檢視 輸入 裝置 說明
En
Right Ctrl

Installation Complete

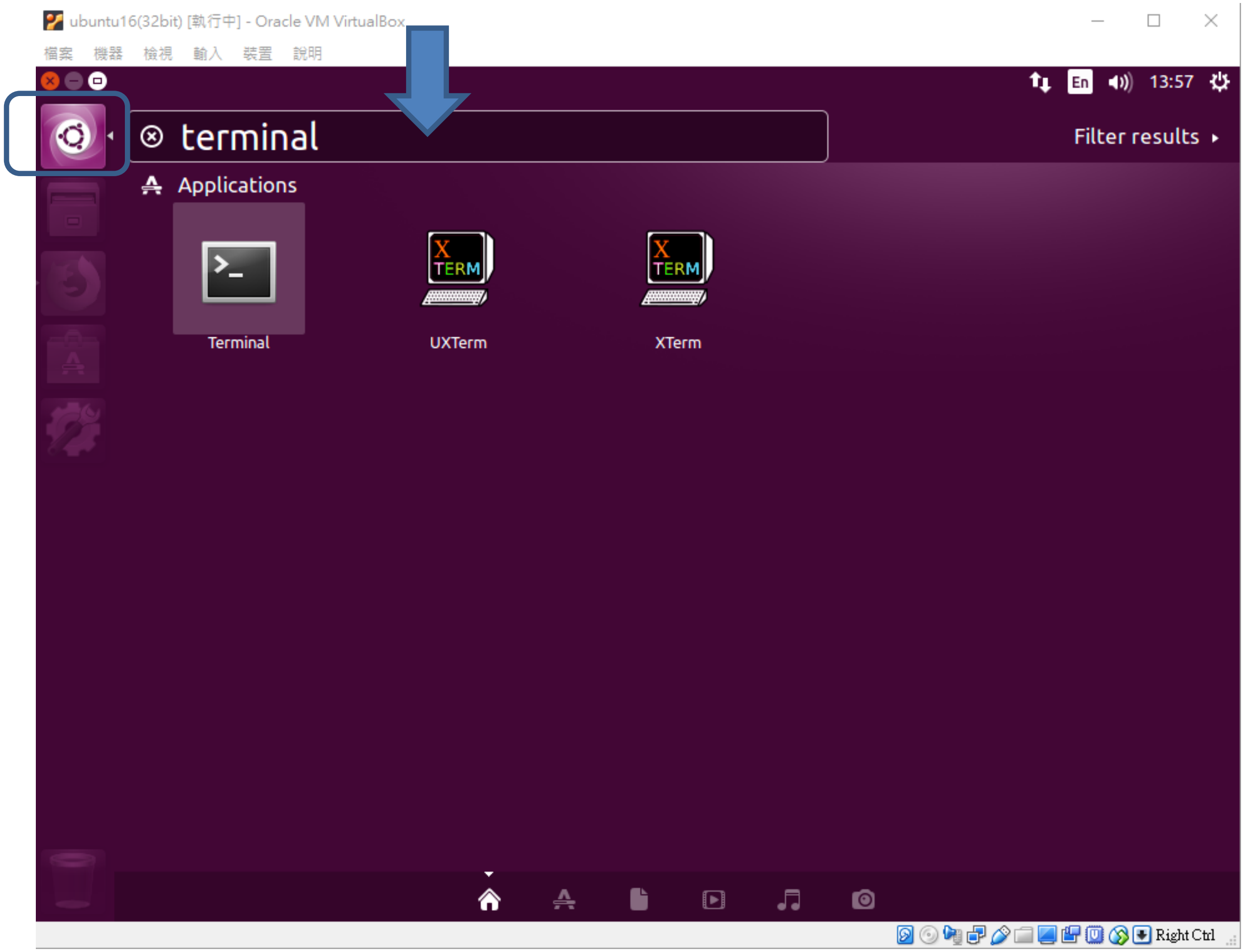
Installation is complete. You need to restart the computer in order to use the new installation.

Outline

- 嵌入式系統:
  - A. cross compile
    1. prepare a Linux system: Virtualbox + Ubuntu 18 (64bit)
    2. download toolchain for PI
    3. compile code on Virtualbox, then execute on PI
  - B. build kernel
    1. Download linux kernel
    2. Configure kernel
    3. Build, then copy to your PI

2. Virtualbox

- □ Open terminal: (type **terminal** here)


ubuntu16(32bit) [執行中] - Oracle VM VirtualBox
檔案 機器 檢視 輸入 裝置 說明
En
13:57
terminal
Filter results
Applications
Terminal
UXTerm
XTerm
Right Ctrl

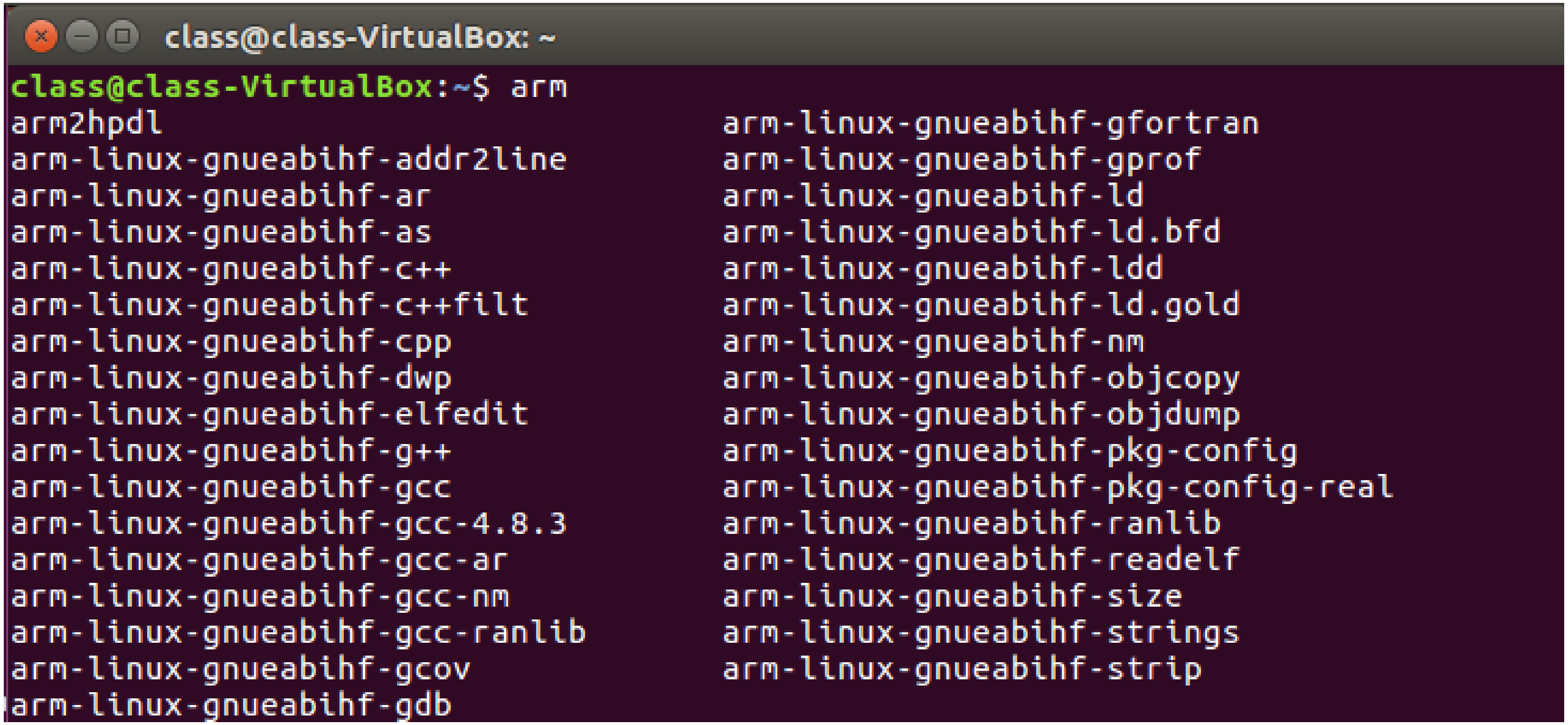
https://www.raspberrypi.org/documentation/linux/kernel/building.md
https://www.raspberrypi.com.tw/tag/cross-compiler/

2. Virtualbox

- Install required dependencies and toolchain
  - sudo apt install git bc bison flex libssl-dev make libc6-dev libncurses5-dev
- Install the 32-bit toolchain for a 32-bit kernel
  - sudo apt install crossbuild-essential-armhf

2. Check environment

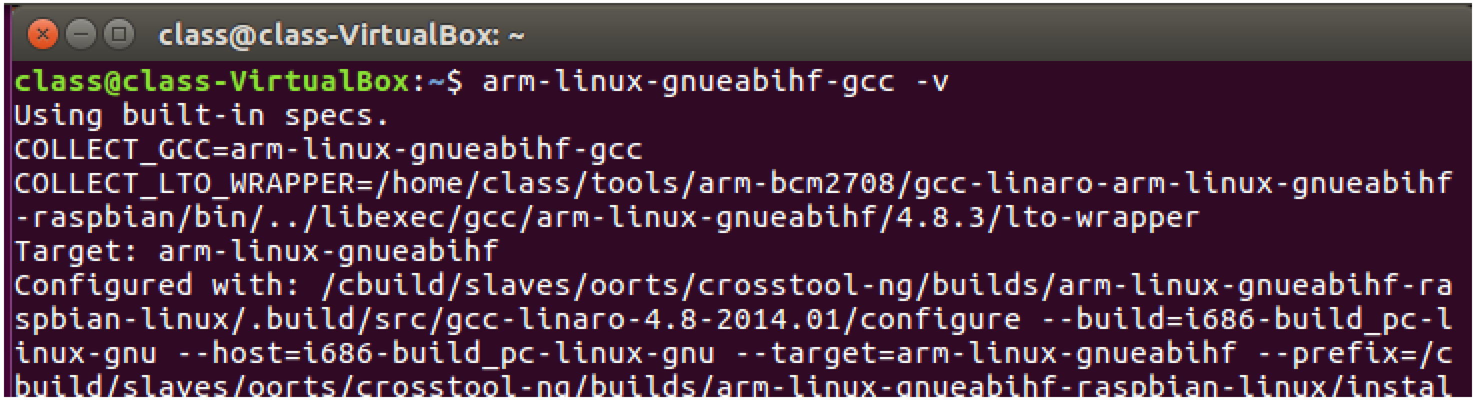
- In terminal:
  - Type **arm**, then press ***tab*** twice



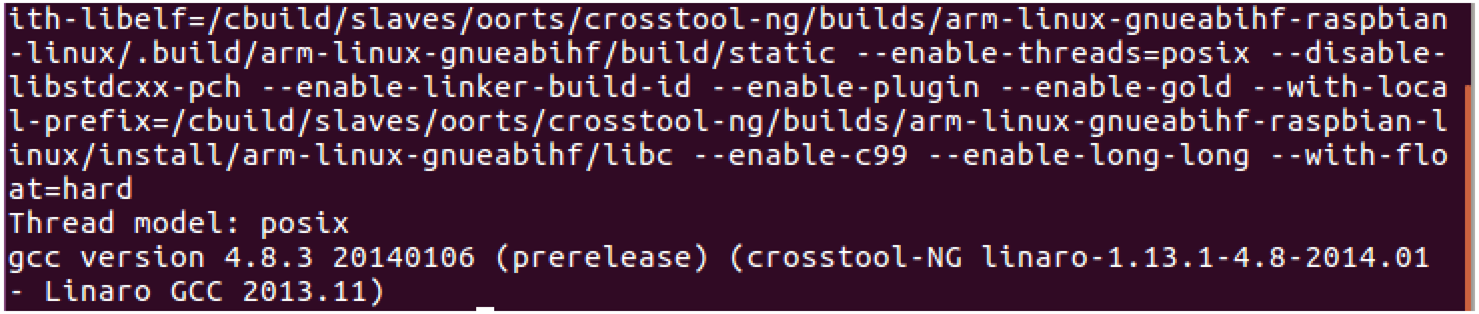
```
class@class-VirtualBox:~$ arm
arm2hpdl                            arm-linux-gnueabihf-gfortran
arm-linux-gnueabihf-addr2line       arm-linux-gnueabihf-gprof
arm-linux-gnueabihf-ar              arm-linux-gnueabihf-ld
arm-linux-gnueabihf-as              arm-linux-gnueabihf-ld.bfd
arm-linux-gnueabihf-c++             arm-linux-gnueabihf-ldd
arm-linux-gnueabihf-c++filt         arm-linux-gnueabihf-ld.gold
arm-linux-gnueabihf-cpp             arm-linux-gnueabihf-nm
arm-linux-gnueabihf-dwp             arm-linux-gnueabihf-objcopy
arm-linux-gnueabihf-elfedit         arm-linux-gnueabihf-objdump
arm-linux-gnueabihf-g++             arm-linux-gnueabihf-pkg-config
arm-linux-gnueabihf-gcc             arm-linux-gnueabihf-pkg-config-real
arm-linux-gnueabihf-gcc-4.8.3       arm-linux-gnueabihf-ranlib
arm-linux-gnueabihf-gcc-ar          arm-linux-gnueabihf-readelf
arm-linux-gnueabihf-gcc-nm          arm-linux-gnueabihf-size
arm-linux-gnueabihf-gcc-ranlib      arm-linux-gnueabihf-strings
arm-linux-gnueabihf-gcov            arm-linux-gnueabihf-strip
arm-linux-gnueabihf-gdb
```

2. Check environment

□ Test: arm-linux-gnueabihf-gcc -v



```
class@class-VirtualBox:~$ arm-linux-gnueabihf-gcc -v
Using built-in specs.
COLLECT_GCC=arm-linux-gnueabihf-gcc
COLLECT_LTO_WRAPPER=/home/class/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf
-raspbian/bin/../libexec/gcc/arm-linux-gnueabihf/4.8.3/lto-wrapper
Target: arm-linux-gnueabihf
Configured with: /cbuild/slaves/oorts/crosstool-ng/builds/arm-linux-gnueabihf-ra
spbian-linux/.build/src/gcc-linaro-4.8-2014.01/configure --build=i686-build_pc-l
inux-gnu --host=i686-build_pc-linux-gnu --target=arm-linux-gnueabihf --prefix=/c
build/slaves/oorts/crosstool-ng/builds/arm-linux-gnueabihf-raspbian-linux/instal
```



```
ith-libelf=/cbuild/slaves/oorts/crosstool-ng/builds/arm-linux-gnueabihf-raspbian
-linux/.build/arm-linux-gnueabihf/build/static --enable-threads=posix --disable-
libstdcxx-pch --enable-linker-build-id --enable-plugin --enable-gold --with-loca
l-prefix=/cbuild/slaves/oorts/crosstool-ng/builds/arm-linux-gnueabihf-raspbian-l
inux/install/arm-linux-gnueabihf/libc --enable-c99 --enable-long-long --with-flo
at=hard
Thread model: posix
gcc version 4.8.3 20140106 (prerelease) (crosstool-NG linaro-1.13.1-4.8-2014.01
- Linaro GCC 2013.11)
```

Outline

- 嵌入式系統:
  - A. cross compile
    1. prepare a Linux system: Virtualbox + Ubuntu 18 (64bit)
    2. download toolchain for PI
    3. compile code on Virtualbox, then execute on PI
  - B. build kernel
    1. Download linux kernel
    2. Configure kernel
    3. Build, then copy to your PI

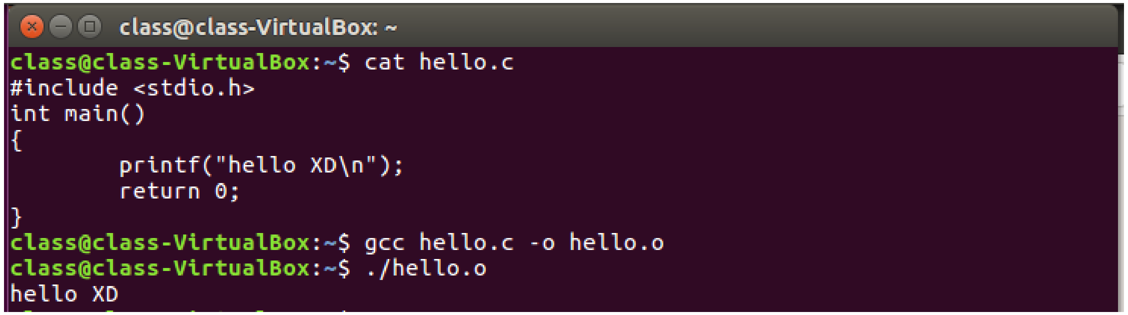
Write code

- Write C code:
  - nano hello.c   // write your code
  - gcc hello.c -o hello.o   // compile it, the output file is hello.o
  - ./hello.o   // execute hello.o

```
#include <stdio.h>
int main()
{
                printf("hello, world\n");
        return 0;
}
```

Compile

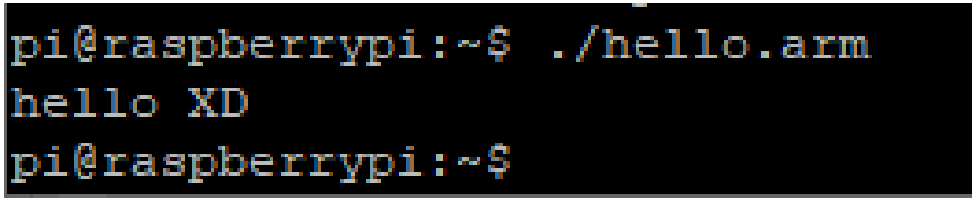
- □ Compile it and execute on PC:



class@class-VirtualBox: ~

```
class@class-VirtualBox:~$ cat hello.c
#include <stdio.h>
int main()
{
        printf("hello XD\n");
        return 0;
}
class@class-VirtualBox:~$ gcc hello.c -o hello.o
class@class-VirtualBox:~$ ./hello.o
hello XD
```

- □ Cross compile, then copy it to PI and execute:
  - □ (@PC) arm-linux-gnueabihf-gcc hello.c -o hello.arm
  - □ // 複製hello.arm到PI裡面
  - □ (@PI) chmod +x hello.arm
  - □ (@PI) ./hello.arm



```
pi@raspberrypi:~$ ./hello.arm
hello XD
pi@raspberrypi:~$
```

Discussion 1

□ Why do we need cross compile?

Quiz 1

- Cross compile the code, with:
  - 1. any word
  - 2. the specific word: COVID-19
  - Ex: I hate COVID-19.

Outline

- 嵌入式系統:
  - A. cross compile
    1. prepare a Linux system: Virtualbox + Ubuntu 18 (64bit)
    2. download toolchain for PI
    3. compile code on Virtualbox, then execute on PI
  - B. build kernel
    1. Download linux kernel
    2. Configure kernel
    3. Build, then copy to your PI

B. Kernel building

- Cross-compiling + Kernel building
  - To build the sources for cross-compilation, make sure you have the dependencies needed on your
  - Install required dependencies and toolchain
    - sudo apt install git bc bison flex libssl-dev make libc6-dev libncurses5-dev
  - Install the 32-bit toolchain for a 32-bit kernel
    - sudo apt install crossbuild-essential-armhf
  - (We have done for cross-compile)

1. Download source and build

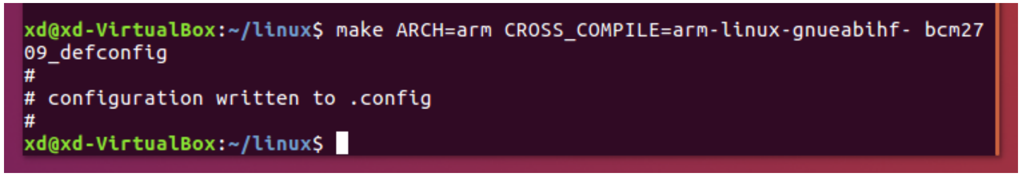
- Get sources (download the minimal source tree for the current branch)
  - git clone --depth=1 https://github.com/raspberrypi/linux

- Load default config and Build sources
  - cd linux
  - # For Pi 3, Pi 3+ or Compute Module 3:
  - KERNEL=kernel7
  - make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- bcm2709_defconfig
  - make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- zImage modules dtbs

  - # For Raspberry Pi 4:
  - KERNEL=kernel8
  - make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- bcm2711_defconfig
  - make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- zImage modules dtbs

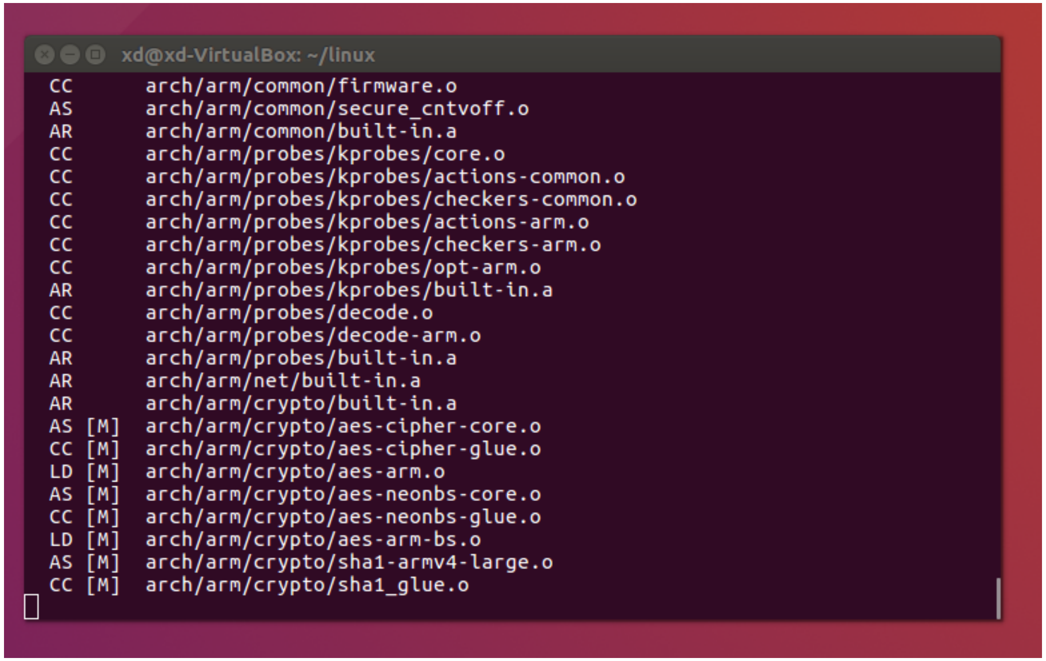
building

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- bcm2709_defconfig



```
xd@xd-VirtualBox:~/linux$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- bcm27
09_defconfig
#
# configuration written to .config
#
xd@xd-VirtualBox:~/linux$ 
```

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- zImage modules dtbs



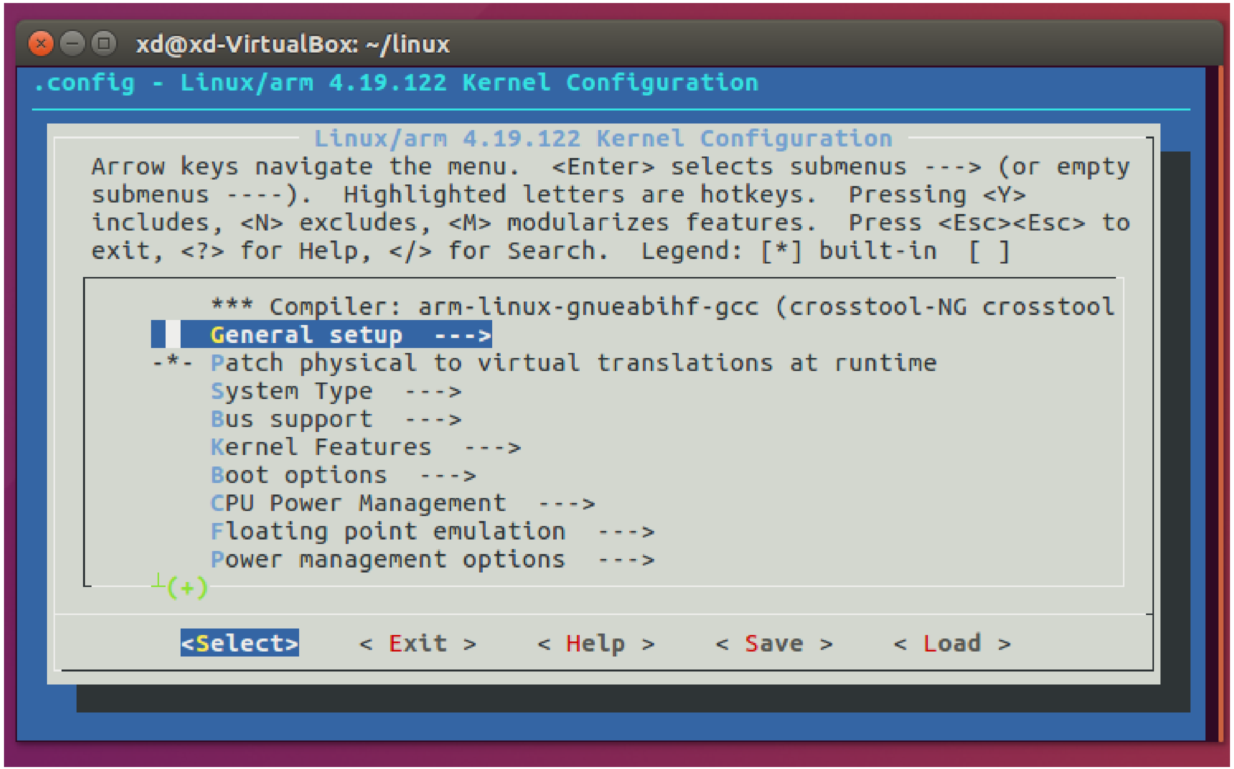
```
  CC      arch/arm/common/firmware.o
  AS      arch/arm/common/secure_cntvoff.o
  AR      arch/arm/common/built-in.a
  CC      arch/arm/probes/kprobes/core.o
  CC      arch/arm/probes/kprobes/actions-common.o
  CC      arch/arm/probes/kprobes/checkers-common.o
  CC      arch/arm/probes/kprobes/actions-arm.o
  CC      arch/arm/probes/kprobes/checkers-arm.o
  CC      arch/arm/probes/kprobes/opt-arm.o
  AR      arch/arm/probes/kprobes/built-in.a
  CC      arch/arm/probes/decode.o
  CC      arch/arm/probes/decode-arm.o
  AR      arch/arm/probes/built-in.a
  AR      arch/arm/net/built-in.a
  AR      arch/arm/crypto/built-in.a
  AS [M]  arch/arm/crypto/aes-cipher-core.o
  CC [M]  arch/arm/crypto/aes-cipher-glue.o
  LD [M]  arch/arm/crypto/aes-arm.o
  AS [M]  arch/arm/crypto/aes-neonbs-core.o
  CC [M]  arch/arm/crypto/aes-neonbs-glue.o
  LD [M]  arch/arm/crypto/aes-arm-bs.o
  AS [M]  arch/arm/crypto/sha1-armv4-large.o
  CC [M]  arch/arm/crypto/sha1_glue.o
```

第一次跑, 可能會花30-60min!

2. Configure kernel

# Use custom config by menuconfig and build

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig

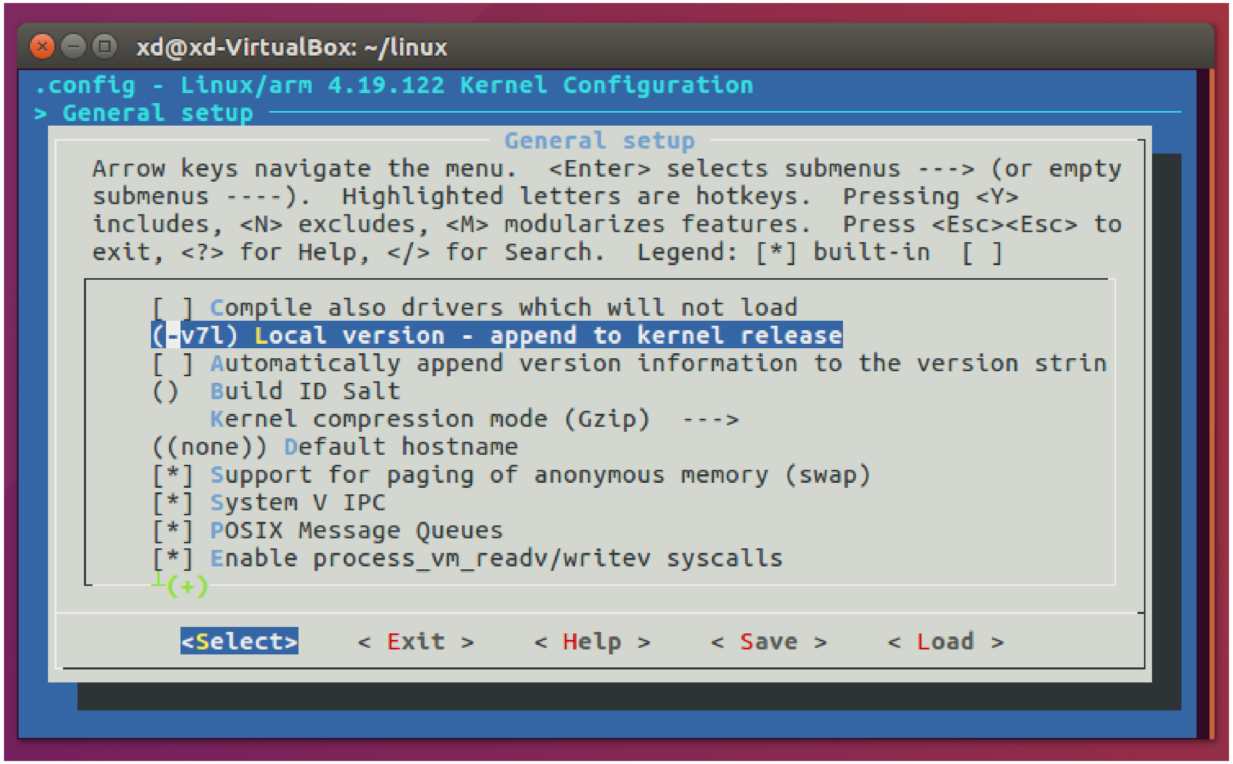


```
.config - Linux/arm 4.19.122 Kernel Configuration
                  Linux/arm 4.19.122 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]

        *** Compiler: arm-linux-gnueabihf-gcc (crosstool-NG crosstool
        General setup  --->
    -*- Patch physical to virtual translations at runtime
        System Type  --->
        Bus support  --->
        Kernel Features  --->
        Boot options  --->
        CPU Power Management  --->
        Floating point emulation  --->
        Power management options  --->
    (+)

      <Select>    < Exit >    < Help >    < Save >    < Load >
```

Go to “General setup”

2. Configure kernel

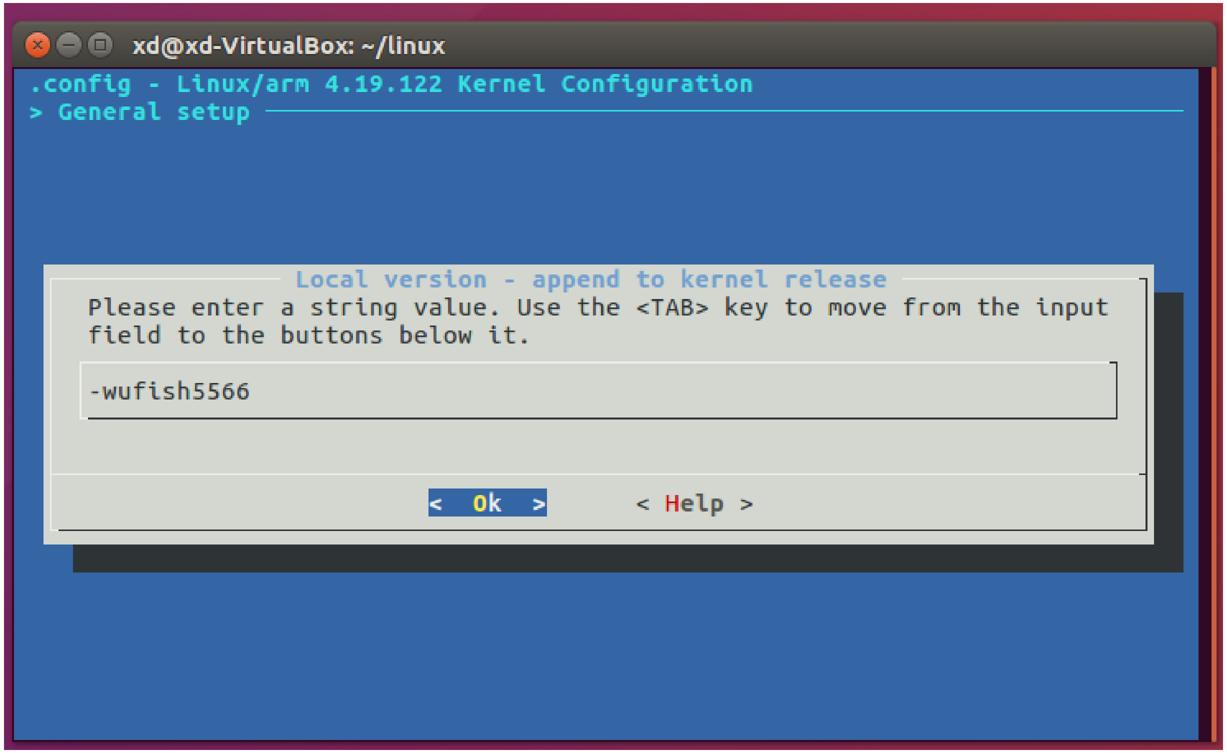
Select “Local version”



```
.config - Linux/arm 4.19.122 Kernel Configuration
> General setup
                                  General setup
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]

      [ ] Compile also drivers which will not load
      (-v7l) Local version - append to kernel release
      [ ] Automatically append version information to the version strin
      ()  Build ID Salt
          Kernel compression mode (Gzip)  --->
      ((none)) Default hostname
      [*] Support for paging of anonymous memory (swap)
      [*] System V IPC
      [*] POSIX Message Queues
      [*] Enable process_vm_readv/writev syscalls
      (+)

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

2. Configure kernel

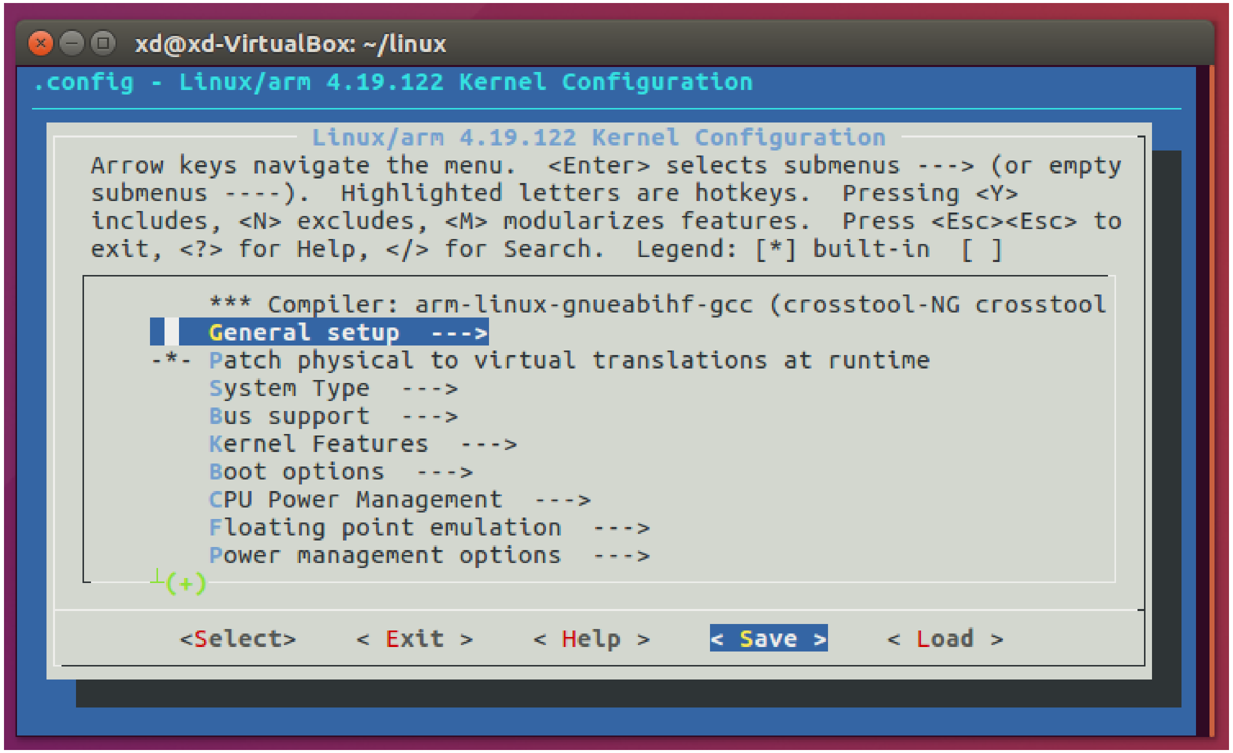
Put your local version



```
.config - Linux/arm 4.19.122 Kernel Configuration
> General setup

Local version - append to kernel release
Please enter a string value. Use the <TAB> key to move from the input
field to the buttons below it.

-wufish5566

<  Ok  >          < Help >
```

2. Configure kernel

save and exit



```
.config - Linux/arm 4.19.122 Kernel Configuration
 ─────────────────── Linux/arm 4.19.122 Kernel Configuration ───────────────────
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]

          *** Compiler: arm-linux-gnueabihf-gcc (crosstool-NG crosstool
          General setup  --->
      -*- Patch physical to virtual translations at runtime
          System Type  --->
          Bus support  --->
          Kernel Features  --->
          Boot options  --->
          CPU Power Management  --->
          Floating point emulation  --->
          Power management options  --->
      └(+)

       <Select>    < Exit >    < Help >    < Save >    < Load >
```

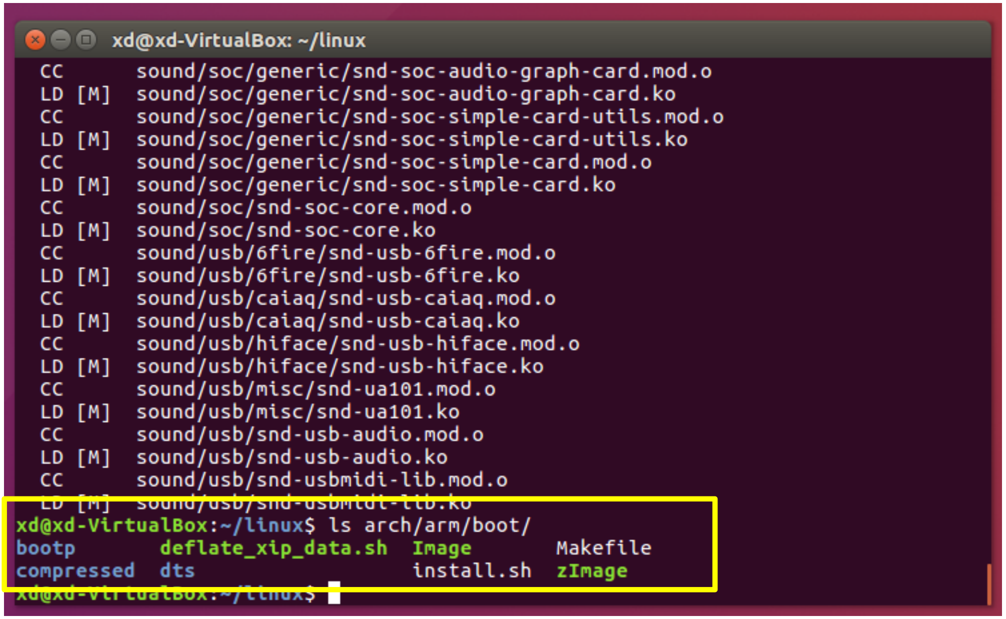
# start building

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- zImage modules dtbs

3. Build kernel

□ After building, your image locates:

□ arch/arm/boot/



xd@xd-VirtualBox: ~/linux

```
  CC      sound/soc/generic/snd-soc-audio-graph-card.mod.o
  LD [M]  sound/soc/generic/snd-soc-audio-graph-card.ko
  CC      sound/soc/generic/snd-soc-simple-card-utils.mod.o
  LD [M]  sound/soc/generic/snd-soc-simple-card-utils.ko
  CC      sound/soc/generic/snd-soc-simple-card.mod.o
  LD [M]  sound/soc/generic/snd-soc-simple-card.ko
  CC      sound/soc/snd-soc-core.mod.o
  LD [M]  sound/soc/snd-soc-core.ko
  CC      sound/usb/6fire/snd-usb-6fire.mod.o
  LD [M]  sound/usb/6fire/snd-usb-6fire.ko
  CC      sound/usb/caiaq/snd-usb-caiaq.mod.o
  LD [M]  sound/usb/caiaq/snd-usb-caiaq.ko
  CC      sound/usb/hiface/snd-usb-hiface.mod.o
  LD [M]  sound/usb/hiface/snd-usb-hiface.ko
  CC      sound/usb/misc/snd-ua101.mod.o
  LD [M]  sound/usb/misc/snd-ua101.ko
  CC      sound/usb/snd-usb-audio.mod.o
  LD [M]  sound/usb/snd-usb-audio.ko
  CC      sound/usb/snd-usbmidi-lib.mod.o
  LD [M]  sound/usb/snd-usbmidi-lib.ko
xd@xd-VirtualBox:~/linux$ ls arch/arm/boot/
bootp       deflate_xip_data.sh  Image       Makefile
compressed  dts                  install.sh  zImage
xd@xd-VirtualBox:~/linux$ 
```

3. Copy to your PI

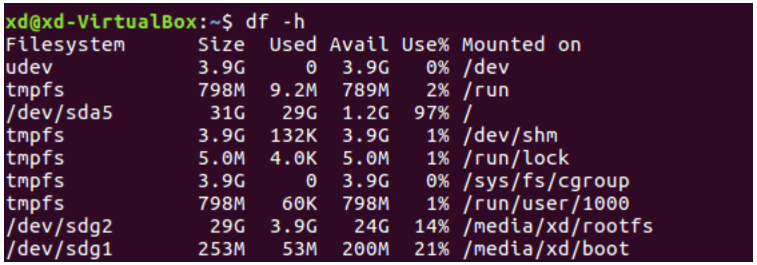
□ copy the kernel and Device Tree blobs to your SD card

1. check SD card state

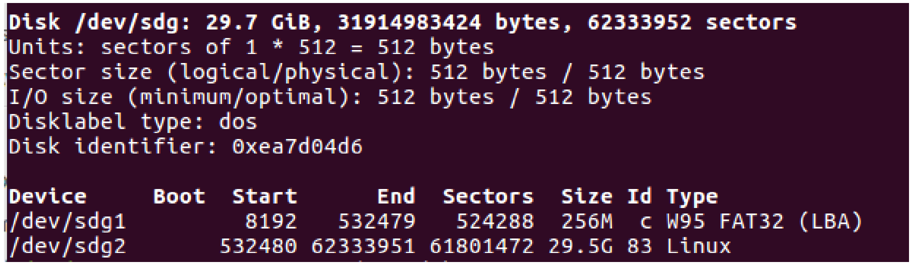
df -h

sudo fdisk -l

- 在虛擬機掛載SD卡時, 通常會自動掛載
- 可以用df -h檢查 before/after 的變化



```
xd@xd-VirtualBox:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            3.9G     0  3.9G   0% /dev
tmpfs           798M  9.2M  789M   2% /run
/dev/sda5        31G   29G  1.2G  97% /
tmpfs           3.9G  132K  3.9G   1% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           3.9G     0  3.9G   0% /sys/fs/cgroup
tmpfs           798M   60K  798M   1% /run/user/1000
/dev/sdg2        29G  3.9G   24G  14% /media/xd/rootfs
/dev/sdg1       253M   53M  200M  21% /media/xd/boot
```

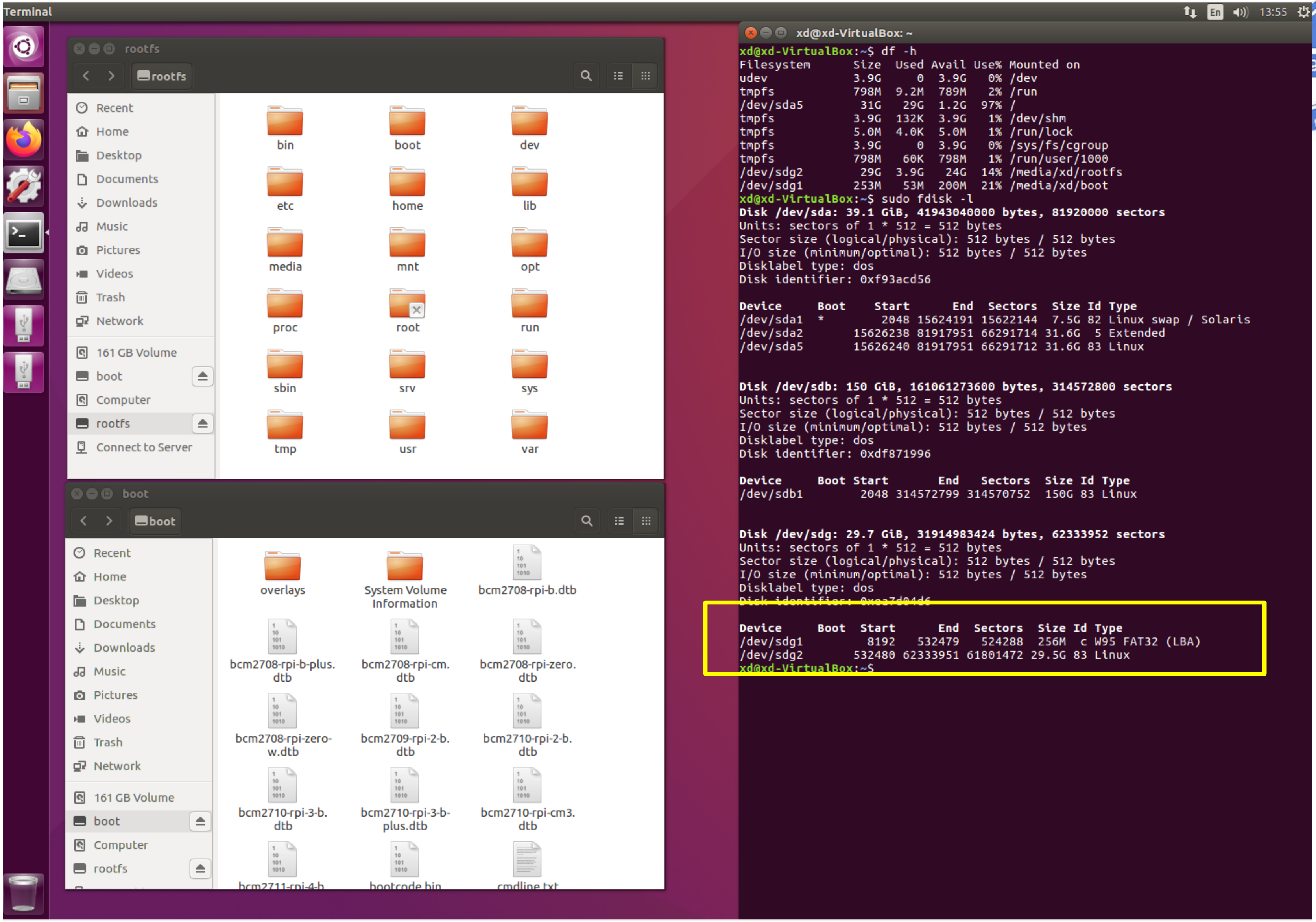


```
Disk /dev/sdg: 29.7 GiB, 31914983424 bytes, 62333952 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xea7d04d6

Device     Boot  Start      End  Sectors  Size Id Type
/dev/sdg1         8192   532479   524288  256M  c W95 FAT32 (LBA)
/dev/sdg2       532480 62333951 61801472 29.5G 83 Linux
```

2. back up old kernel

cp -rf sd/boot/* Desktop/boot_PI/

Ex: cp -rf /media/xd/boot/* Desktop/boot_PI/



```
xd@xd-VirtualBox:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            3.9G     0  3.9G   0% /dev
tmpfs           798M  9.2M  789M   2% /run
/dev/sda5        31G   29G  1.2G  97% /
tmpfs           3.9G  132K  3.9G   1% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           3.9G     0  3.9G   0% /sys/fs/cgroup
tmpfs           798M   60K  798M   1% /run/user/1000
/dev/sdg2        29G  3.9G   24G  14% /media/xd/rootfs
/dev/sdg1       253M   53M  200M  21% /media/xd/boot
xd@xd-VirtualBox:~$ sudo fdisk -l
Disk /dev/sda: 39.1 GiB, 41943040000 bytes, 81920000 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xf93acd56

Device     Boot    Start      End  Sectors  Size Id Type
/dev/sda1  *        2048 15624191 15622144  7.5G 82 Linux swap / Solaris
/dev/sda2       15626238 81917951 66291714 31.6G  5 Extended
/dev/sda5       15626240 81917951 66291712 31.6G 83 Linux


Disk /dev/sdb: 150 GiB, 161061273600 bytes, 314572800 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xdf871996

Device     Boot Start       End   Sectors  Size Id Type
/dev/sdb1        2048 314572799 314570752  150G 83 Linux


Disk /dev/sdg: 29.7 GiB, 31914983424 bytes, 62333952 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xea7d04d6

Device     Boot  Start      End  Sectors  Size Id Type
/dev/sdg1         8192   532479   524288  256M  c W95 FAT32 (LBA)
/dev/sdg2       532480 62333951 61801472 29.5G 83 Linux
xd@xd-VirtualBox:~$
```

3. Copy to your PI

□ copy the kernel and Device Tree blobs onto the SD card

3. copy new kernel to SD card

```
cd ~/linux
sudo  cp  arch/arm/boot/zImage                    sd_boot/kernel-madebyyou.img
sudo  cp  arch/arm/boot/dts/*.dtb                 sd_boot/
sudo  cp  arch/arm/boot/dts/overlays/*.dtb*       sd_boot/overlays/
sudo  cp  arch/arm/boot/dts/overlays/README       sd_boot/overlays/
```

4. edit the config.txt file, add the following setting

```
kernel=kernel-madebyyou.img
```

5. remove SD card

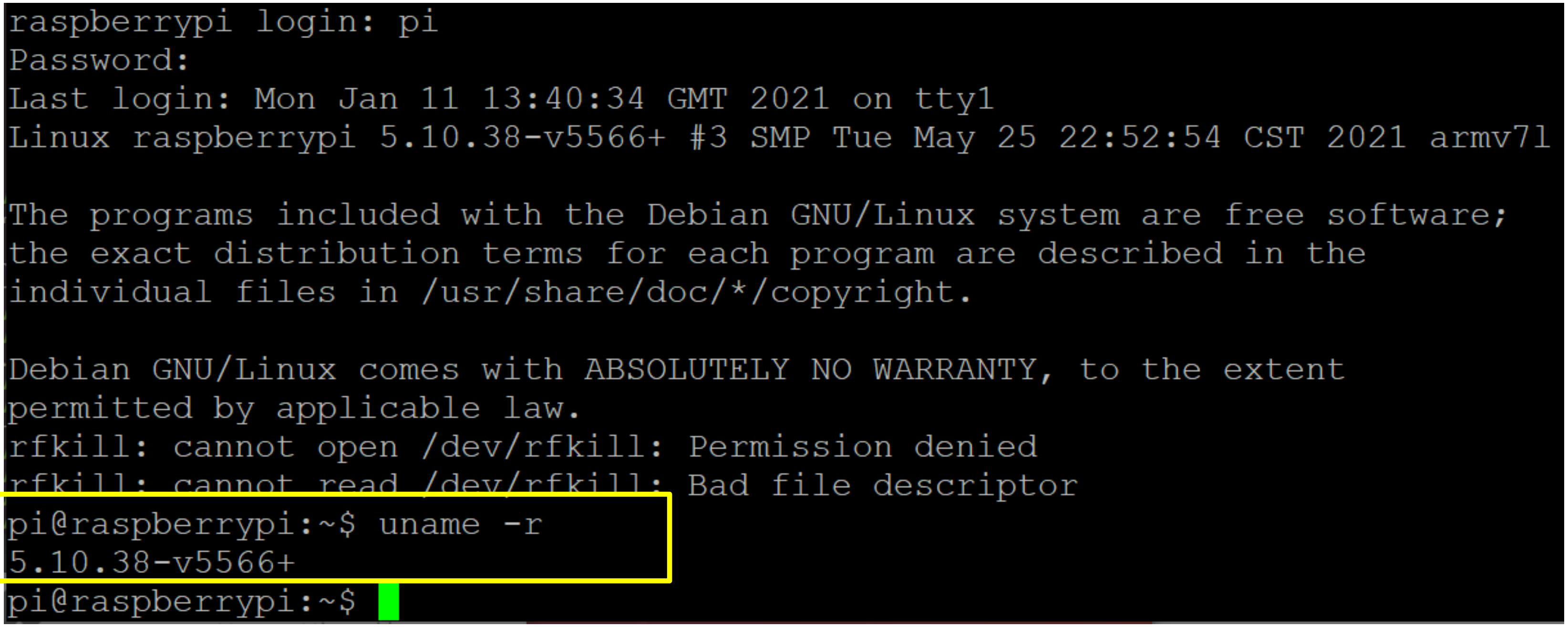
```
sudo  umount  /dev/sdX1
sudo  umount  /dev/sdX2
```

6. Insert SD card to PI and boot!!

Boot your PI

# check your kernel version

uname -r



```
raspberrypi login: pi
Password:
Last login: Mon Jan 11 13:40:34 GMT 2021 on tty1
Linux raspberrypi 5.10.38-v5566+ #3 SMP Tue May 25 22:52:54 CST 2021 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
rfkill: cannot open /dev/rfkill: Permission denied
rfkill: cannot read /dev/rfkill: Bad file descriptor
pi@raspberrypi:~$ uname -r
5.10.38-v5566+
pi@raspberrypi:~$
```

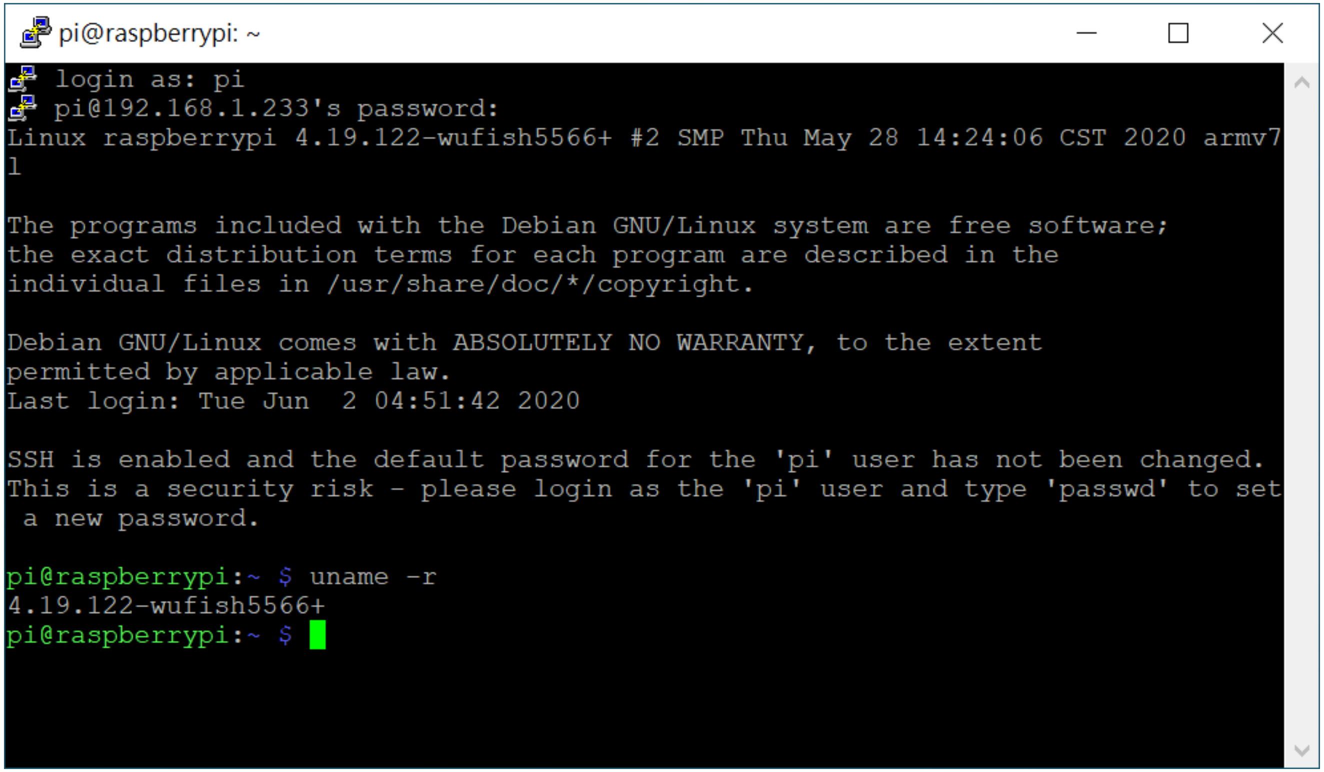
How to go back to original kernel?

# edit the config.txt file, remove the setting

~~kernel=kernel-madebyyou.img~~

Quiz2

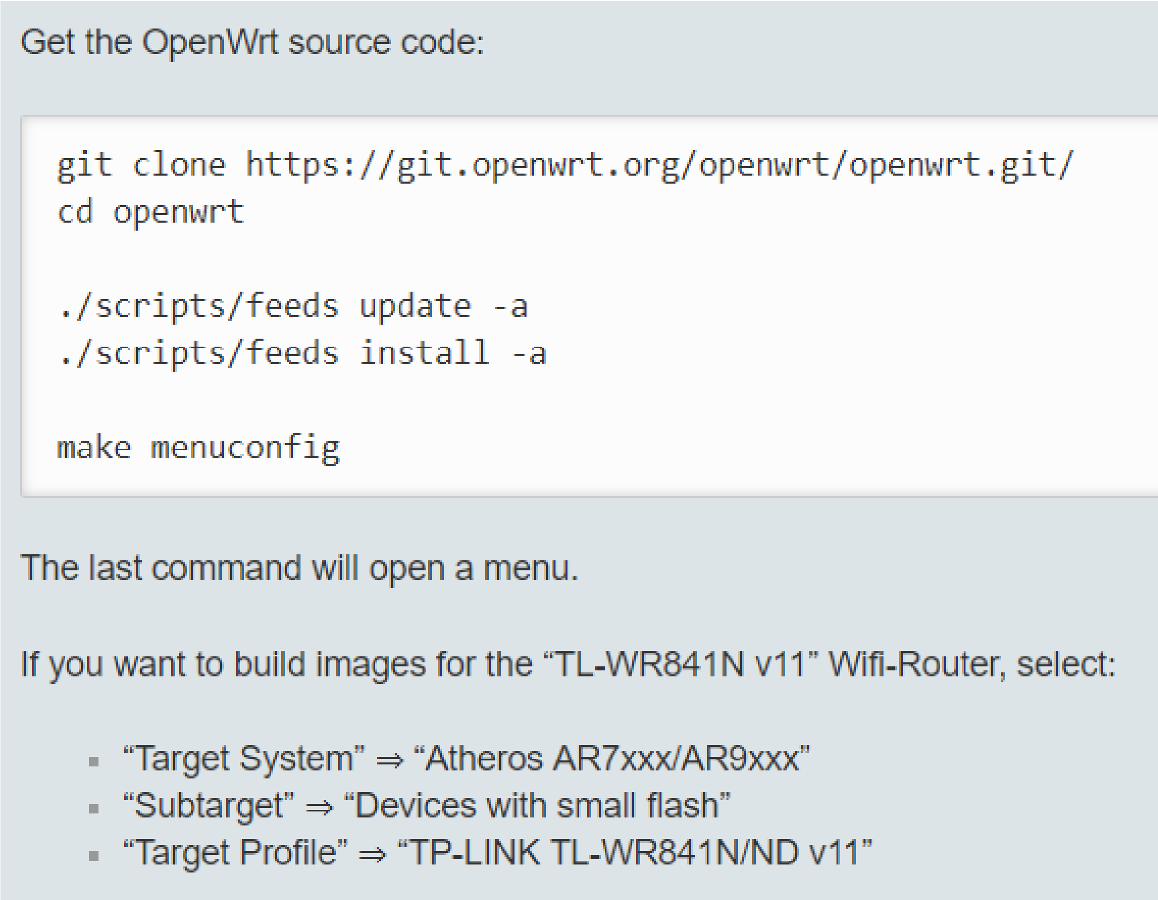
- □ Build your own kernel, **put your std_ID** in local version



```
login as: pi
pi@192.168.1.233's password:
Linux raspberrypi 4.19.122-wufish5566+ #2 SMP Thu May 28 14:24:06 CST 2020 armv7
l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Jun  2 04:51:42 2020

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
 a new password.

pi@raspberrypi:~ $ uname -r
4.19.122-wufish5566+
pi@raspberrypi:~ $
```

Build other kernel (system)

- Ex: OpenWRT



Get the OpenWrt source code:

```
git clone https://git.openwrt.org/openwrt/openwrt.git/
cd openwrt

./scripts/feeds update -a
./scripts/feeds install -a

make menuconfig
```

The last command will open a menu.

If you want to build images for the “TL-WR841N v11” Wifi-Router, select:

- “Target System” ⇒ “Atheros AR7xxx/AR9xxx”
- “Subtarget” ⇒ “Devices with small flash”
- “Target Profile” ⇒ “TP-LINK TL-WR841N/ND v11”

https://github.com/openwrt/openwrt

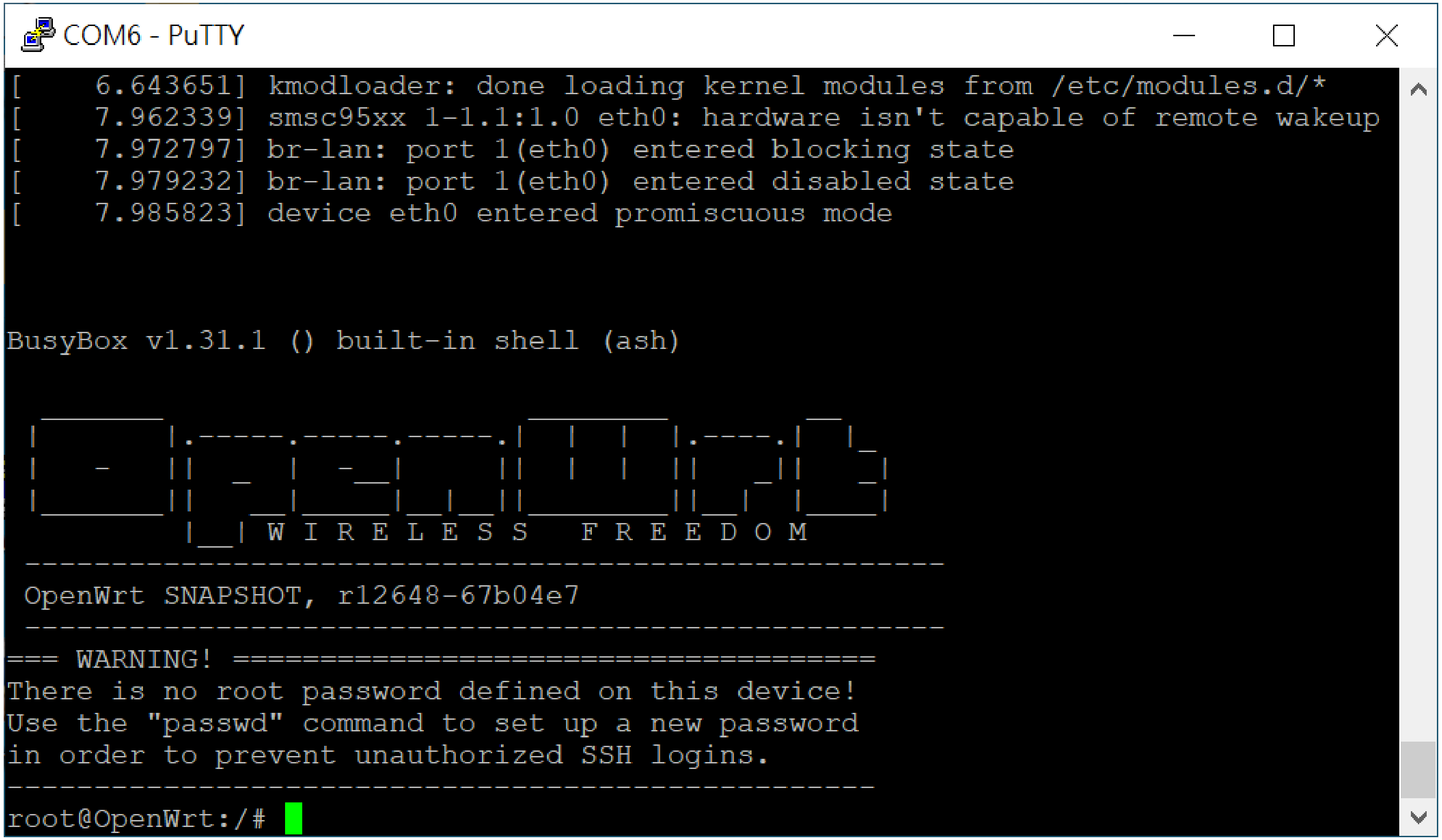
```
  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 -----------------------------------------------------
```

```
sudo apt-get install build-essential subversion libncurses5-dev zlib1g-dev gawk gcc-multilib flex git-core
gettext libssl-dev unzip

git clone git://github.com/openwrt/openwrt.git
cd openwrt/

./scripts/feeds update -a  # obtain all the latest package
./scripts/feeds install -a   # install symlinks for all obtained packages into package/feeds/

# select your preferred configuration for the toolchain, target system & firmware packages
make menuconfig

# build your firmware
make

cd bin/targets/bcm27xx/bcm2709/
```

(記得先解壓縮.gz)

```
sudo dd if=openwrt-bcm27xx-bcm2709-rpi-2-ext4-factory.img of=/dev/sdg bs=2M conv=fsync
```

OpenWrt on PI



```
[    6.643651] kmodloader: done loading kernel modules from /etc/modules.d/*
[    7.962339] smsc95xx 1-1.1:1.0 eth0: hardware isn't capable of remote wakeup
[    7.972797] br-lan: port 1(eth0) entered blocking state
[    7.979232] br-lan: port 1(eth0) entered disabled state
[    7.985823] device eth0 entered promiscuous mode



BusyBox v1.31.1 () built-in shell (ash)

  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 -----------------------------------------------------
 OpenWrt SNAPSHOT, r12648-67b04e7
 -----------------------------------------------------
=== WARNING! =====================================
There is no root password defined on this device!
Use the "passwd" command to set up a new password
in order to prevent unauthorized SSH logins.
--------------------------------------------------
root@OpenWrt:/#
```

https://wiki.dd-wrt.com/wiki/index.php/WHR-G300N_V2

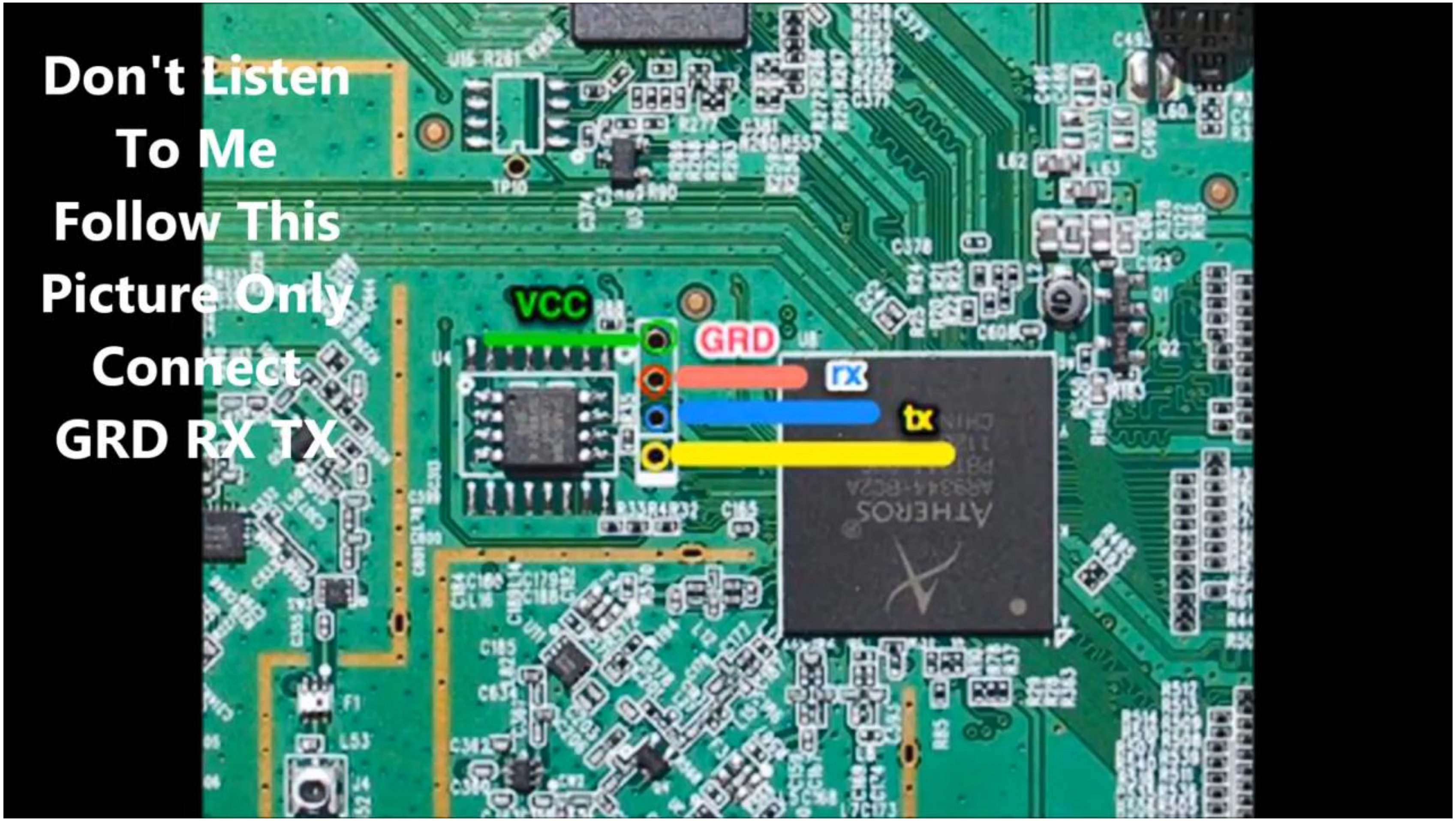
Application: Flash your AP

□ Contents

1. Installing DD-WRT
    1. Flashing from Buffalo Firmware
    2. DD-WRT Upgrade Flashes
2. Specific configuration
    1. DDNS
3. Going back to Buffalo Firmware
4. Recovery from Bricking, Semi-bricking

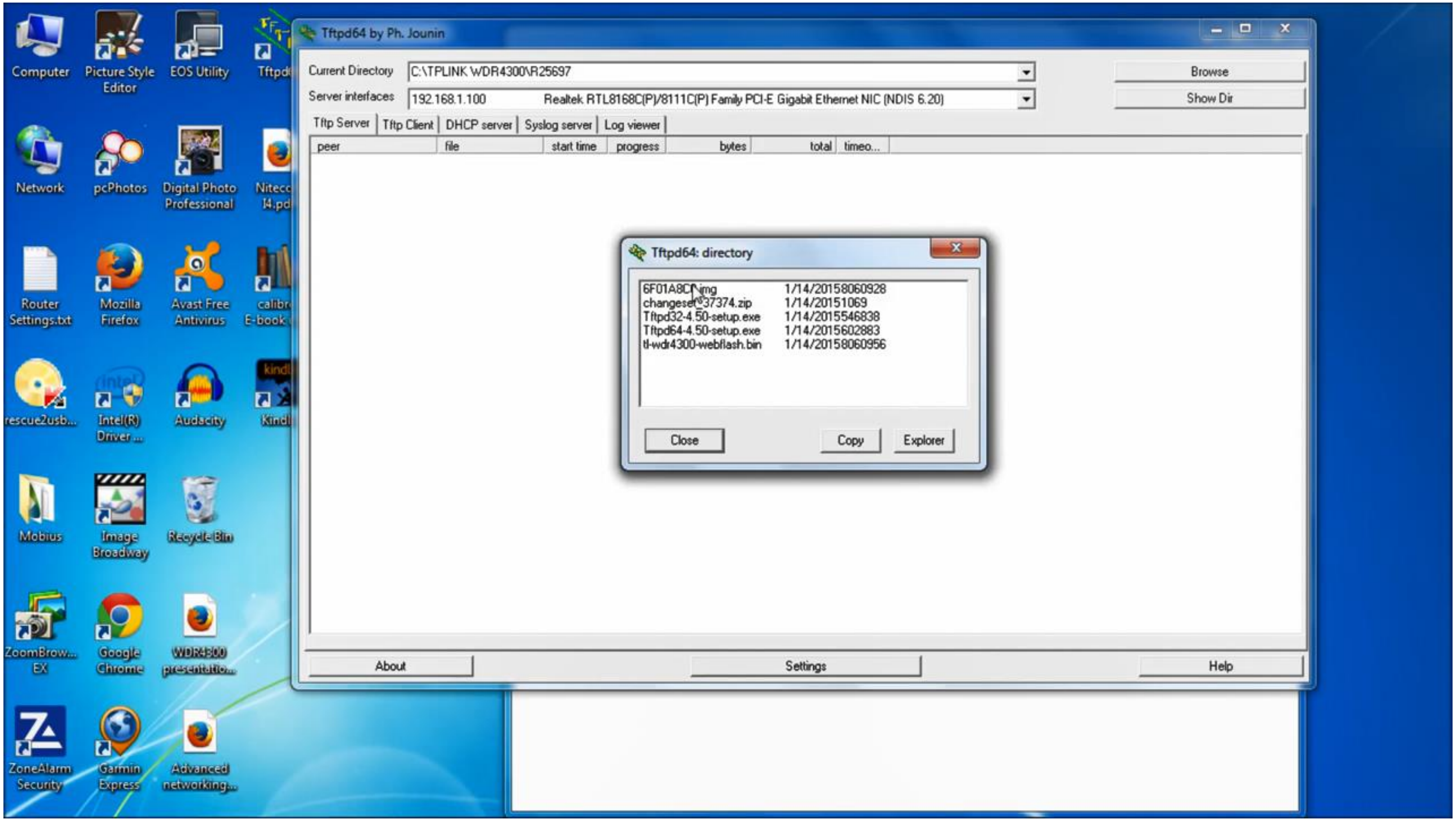
https://www.youtube.com/watch?v=t35Pei_eb6o

Bricked TP-Link WDR4300 Router Recovery Using UART Serial Converter - 1



Don't Listen To Me Follow This Picture Only Connect GRD RX TX https://www.youtube.com/watch?v=wCwBvoueBG4

Bricked TP-Link WDR4300 Router Recovery Using UART Serial Converter - 2



Current Directory C:\TPLINK WDR4300\R25697

Server interfaces 192.168.1.100 Realtek RTL8168C(P)/8111C(P) Family PCI-E Gigabit Ethernet NIC (NDIS 6.20)

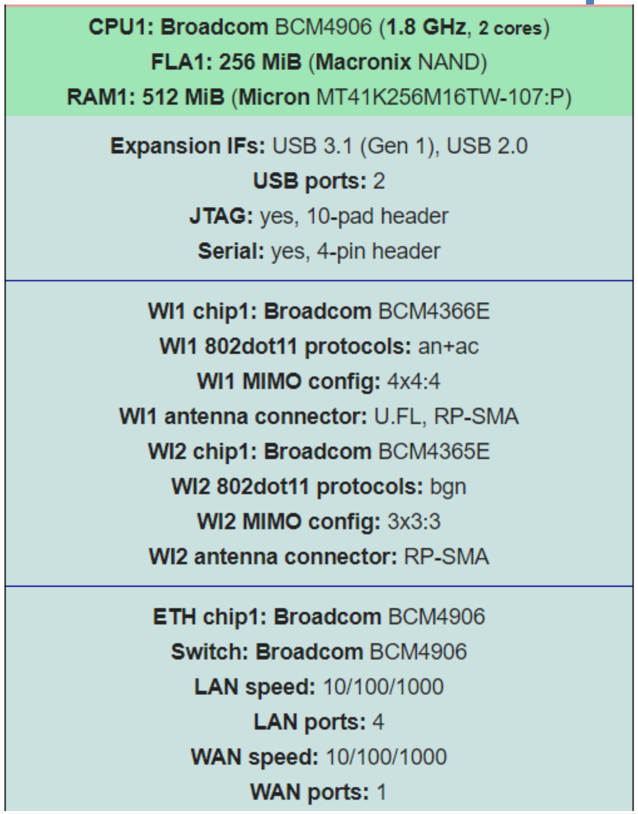
```
6F01A8C .img              1/14/20158060928
changeset_37374.zip       1/14/20151069
Tftpd32-4.50-setup.exe    1/14/2015546838
Tftpd64-4.50-setup.exe    1/14/2015602883
tl-wdr4300-webflash.bin   1/14/20158060956
```

https://openwrt.org/supported_devices

OpenWrt requirements

1. General requirements for OpenWrt support
2. SoC / target supported by OpenWrt
3. Sufficient Flash to accommodate OpenWrt firmware image
   - 4MB min (won't be able to install GUI (LuCI))
   - 8MB better (will fit GUI and some other applications)
4. Sufficient RAM for stable operation
   - 32MB min, 64MB better

- Is your device supported?
  - Go to https://wikidevi.com
  - Ex: ASUS_RT-AC86U
  - https://wikidevi.com/wiki/ASUS_RT-AC86U



CPU1: Broadcom BCM4906 (**1.8 GHz**, **2 cores**)
FLA1: 256 MiB (**Macronix** NAND)
RAM1: 512 MiB (**Micron** MT41K256M16TW-107:P)

Expansion IFs: USB 3.1 (Gen 1), USB 2.0
USB ports: 2
JTAG: yes, 10-pad header
Serial: yes, 4-pin header

WI1 chip1: Broadcom BCM4366E
WI1 802dot11 protocols: an+ac
WI1 MIMO config: 4x4:4
WI1 antenna connector: U.FL, RP-SMA
WI2 chip1: Broadcom BCM4365E
WI2 802dot11 protocols: bgn
WI2 MIMO config: 3x3:3
WI2 antenna connector: RP-SMA

ETH chip1: Broadcom BCM4906
Switch: Broadcom BCM4906
LAN speed: 10/100/1000
LAN ports: 4
WAN speed: 10/100/1000
WAN ports: 1

Summary

- Practice Lab (cross-compile, build kernel)

- Write down the answer for discussion
  - **Discussion (**Deadline: Before 6/4, 12:00)
    - **Why do we need cross compile?**

- Demonstrate **Quiz 1 and 2** to TAs
  - **Quiz1: Cross compile the code**
  - **Quiz2: Build your own kernel, put your std_ID in local version**
    - Deadline: Before 5/28, 15:10
    - Late demo: Before 6/4, 15:10