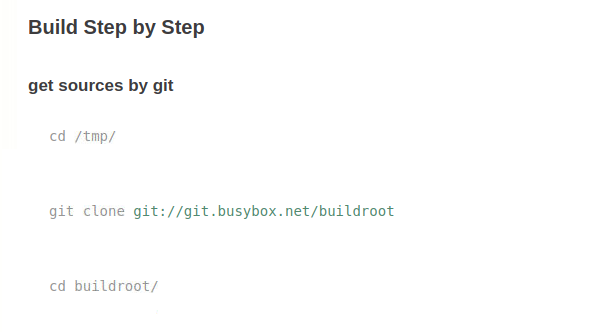


這個步驟主要就是讓你的docker可以和host互相有個共用的資料夾(有點像是雲端同步) 待會複製東西的時候就要記好目前的存放位置，在這裡為例就是/tmp/package(是docker container的資料夾)和 ~/mnt/mnt\_docker(host的共享資料夾)





在下載所有資源之前要先建立組態

**make list-defconfigs**

找到樹梅派1

**make raspberrypi\_defconfig**

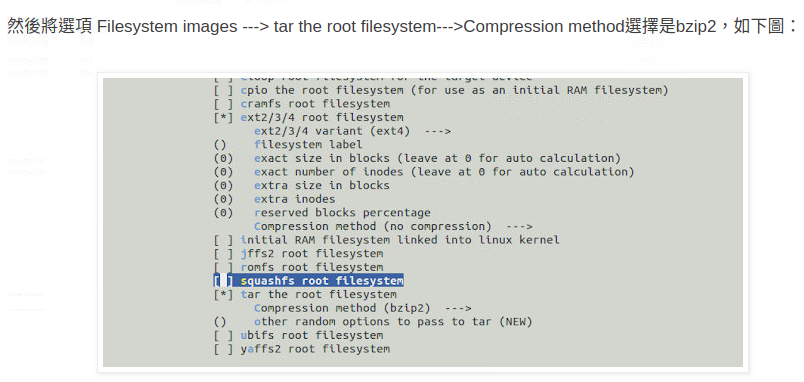
然後只要下這個指令就會下載所有個資源：

**make source**

如果缺少**cpio**和**rsync**的話直接執行

**apt-get install cpio && apt-get install rsync**即可





這樣我們待回的root filesytem就會以tar ball 的型式。

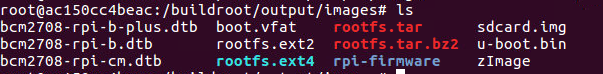
然後我們也想要自己build個u-boot來玩玩。

所以選項 Bootloaders --> 選擇U-boot選項，Board defconfig 輸入「rpi\_1」。



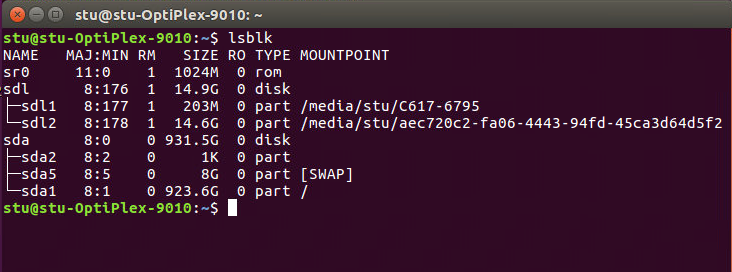
也可以直接make





切割的步驟 (要另外開一個終端機來執行)

使用參考網站，裡面有切割的標題( Partition SD Card)會教你怎麼切，如果想要看有沒有切好，使用lsblk 指令即可，像我的sd卡名稱是sdl，所以切完會像下面一樣(不過有時候可能要重新插入再輸入lsblk去查看)



記得要掛載SD卡!如果拔掉後發現他掛載不上去，先umount你的sd卡在重新mount一次即可。

**sudo umount /dev/sdl1**

**sudo umount /dev/sdl2**

**sudo mount /dev/sdl1 ~/mnt/fat32**

**sudo mount /dev/sdl2 ~/mnt/rtfs**

1. Kernel Image

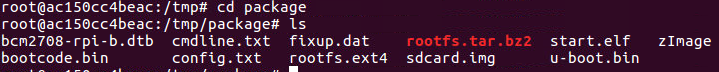
我們將kernel的Image複製到boot section，但是首先要先將東西複製到tmp/package讓/mnt/mnt\_docker 有zImage，再將其複製到~/mnt/fat32

所以步驟是

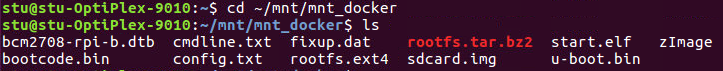
**cd /buildroot/output/images**

**cp zImage /tmp/package**

然後**cd /tmp/package**查看

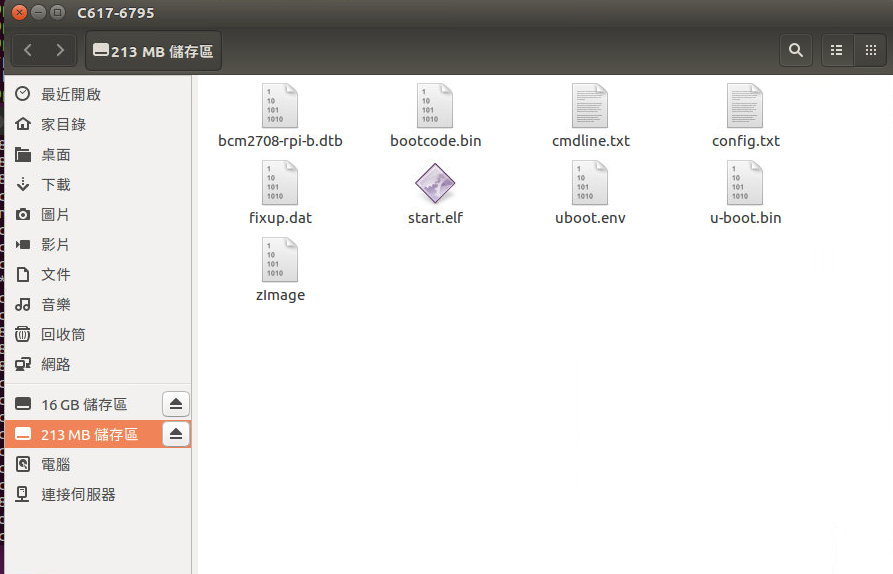


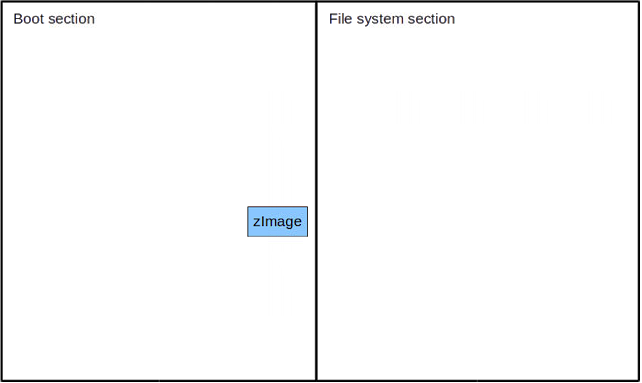
再開另一個終端機輸入 **cp /mnt/mnt\_docker** 進入**mnt\_docker中**



最後將” zImage”複製到sd卡 (在mnt/mnt\_docker 執行)

sudo cp zImage ~/mnt/fat32





1. Root file system

將root file system的image複製到第二個partition底下然後再解壓縮。步驟依序是先從**/buildroot/output/images**的資料夾裡將**images**複製到**tmp/package**的資料夾上，再從另一個終端機**(mnt/mnt\_docker)**複製到**~/mnt/rtfs。**

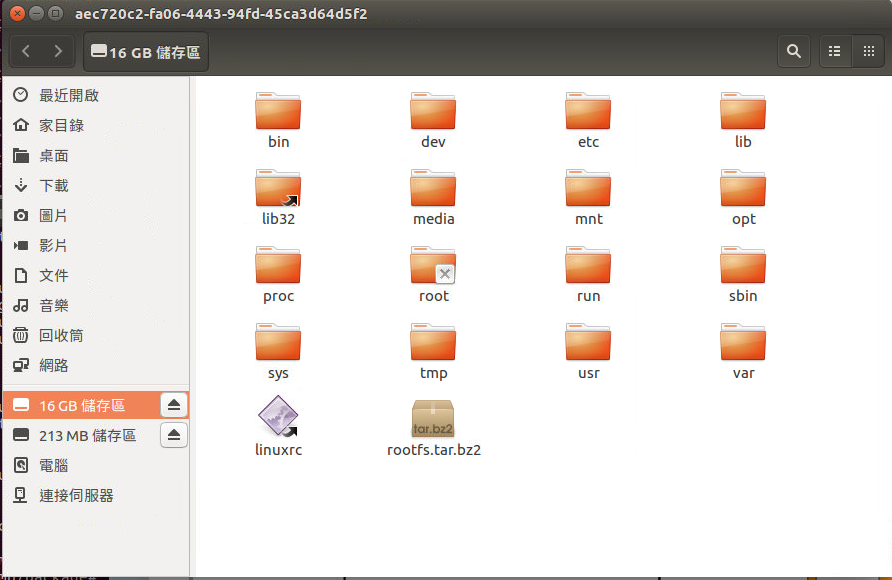
**cd /buildroot/output/images**

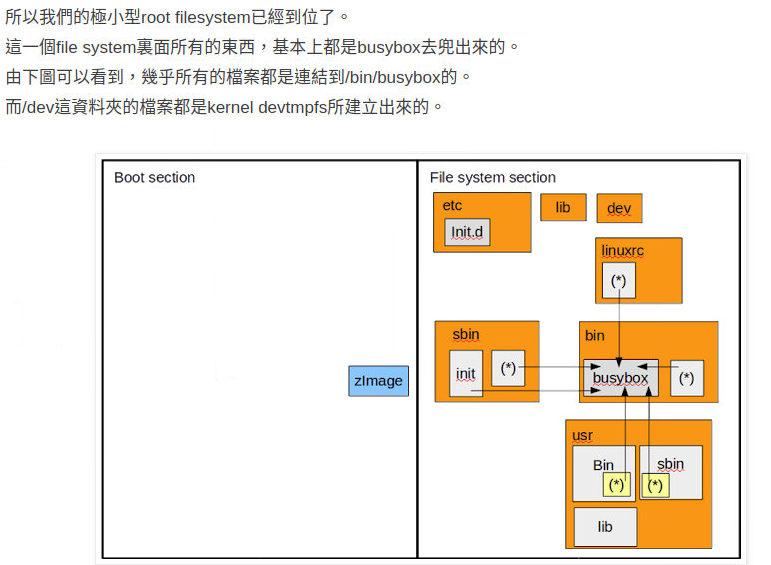
**cp rootfs.tar.bz2 /tmp/package**

之後就在你之前開mnt/mnt\_docker的終端機裡面執行cd ~/mnt/rtfs然後作解壓縮的動作

**sudo tar -jvxf rootfs.tar.bz2**

壓縮完後就可在你的SD卡上看到下圖的東西





1. Device tree blob

將裝置樹檔複製到**boot section**。

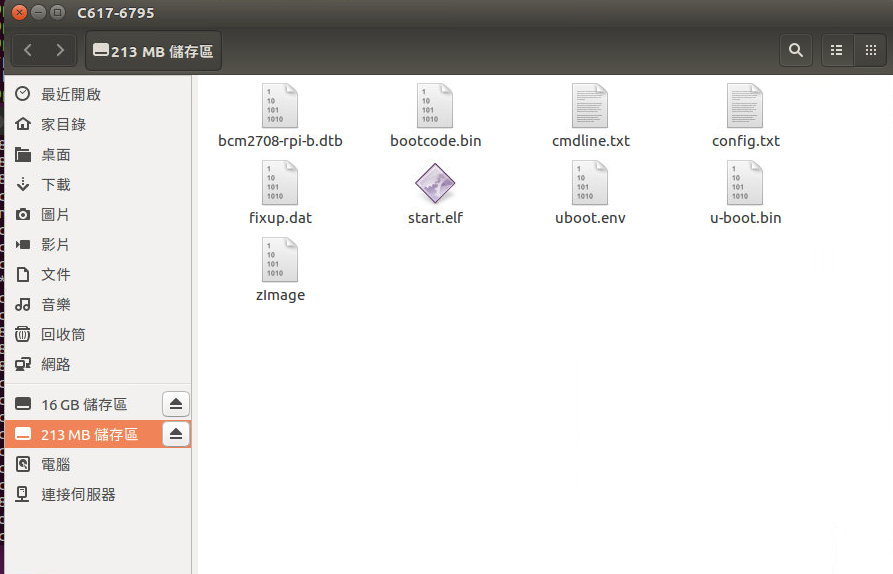
**cd /buildroot/output/images**

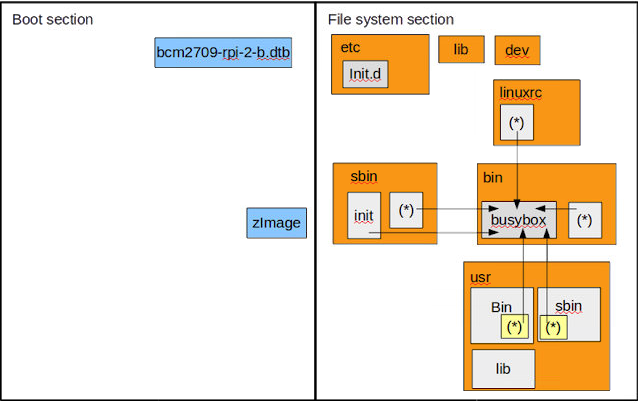
**cp bcm2708-rpi-b.dtb /tmp/package**

之後就在你之前開**mnt/mnt\_docker**的終端機執行

**cd /mnt/mnt\_docker**

**cp bcm2708-rpi-b.dtb ~/mnt/fat32/**





1. Raspberry pi firmware

在firmware裡，我們需要的檔案有：

**start.elf （GPU frimware）**

**bootcode.bin (bootloaders)**

**config.txt** (裏面也是有一些組態)

**cmdline.txt** (這個檔案裏面的文字都會當作參數傳遞給Kernel)

**fixup.dat** (用來組態GPU和CPU之間的SDRAM partition)

(就是把**rpi-firmware**資料夾底下的東西都複製到**boot section)**

執行**cd /buildroot/output/images/rpi-firmware)**



再來一個一個複製到 /tmp/package

**cp start.elf** **/tmp/package**

**cp bootcode.bin /tmp/package**

**cp config.txt /tmp/package**

**cp cmdline.txt /tmp/package**

**cp fixup.dat /tmp/package**

再到另一個終端機(/mnt/mnt\_docker)執行

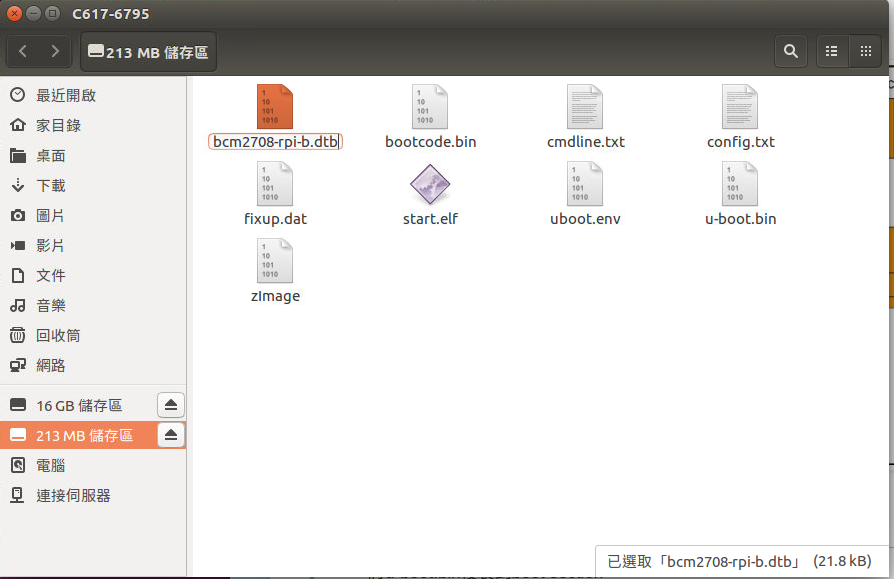
**sudo cp start.elf ~/mnt/fat32**

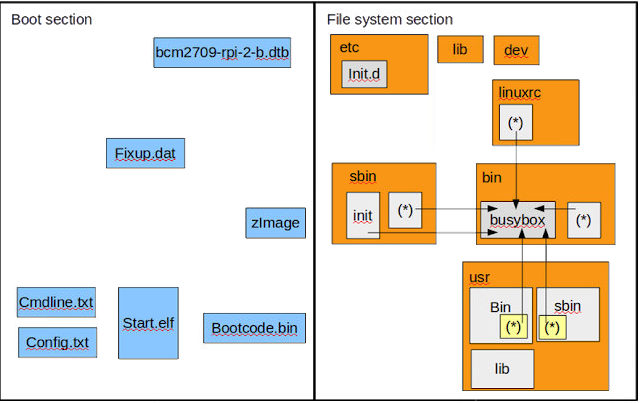
**sudo cp bootcode.bin ~/mnt/fat32**

**sudo cp config.txt ~/mnt/fat32**

**sudo cp cmdline.txt ~/mnt/fat32**

**sudo cp fixup.dat ~/mnt/fat32**





1. U-boot (針對u-boot開機)

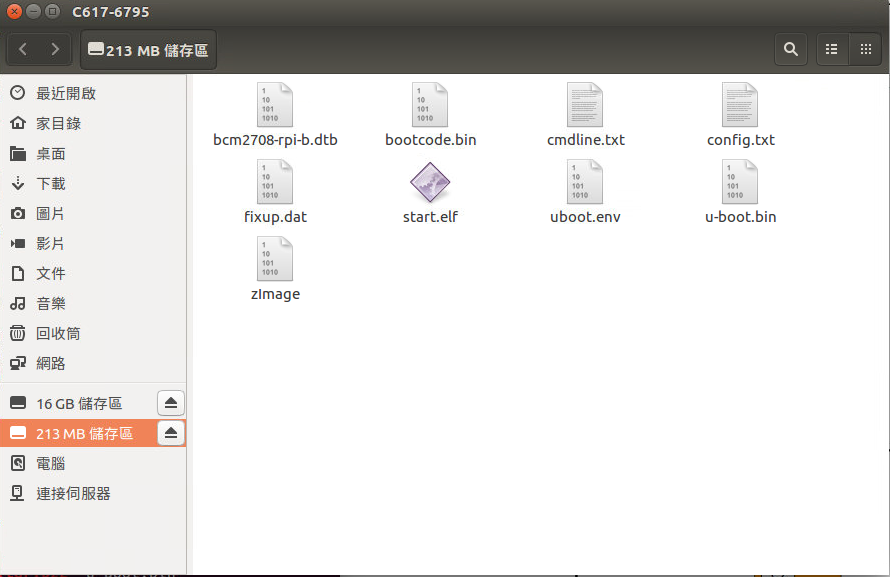
將u-boot.bin複製到boot section，

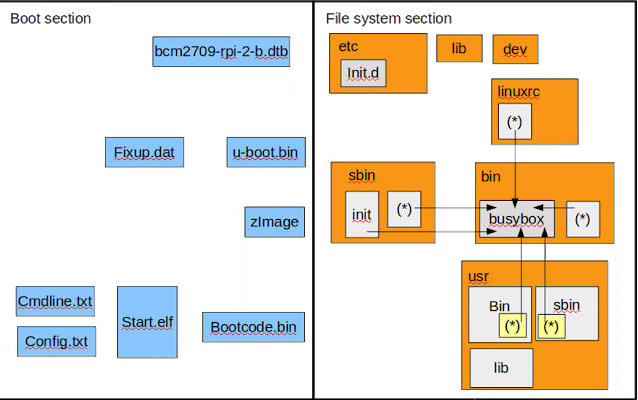
**cd /buildroot/output/images/**

**cp u-boot.bin /tmp/package**

再到另一個終端機**(/mnt/mnt\_docker)**執行

**sudo** **cp** **u-boot.bin ~/mnt/fat32**





然後修改config.txt，

kernel=zImage

改成

kernel=u-boot.bin

這樣在final-stage bootloader時，就會去讀我們的u-boot.bin而不是直接進入kernel。

如果你沒有要針對u-boot開機的話，將uboot.bin複製到SD卡後就可以開機了

參考網站: <http://fichugh.blogspot.com/2016/02/buildroot-study.html>