Once you've imaged your SD-card with *Win32 Disk Imager*, you end up with the two partitions.

- A vfat partition , */dev/mmcblk0p1*
- An ext4 partition, */dev/mmcblk0p2*

The vfat (or FAT32) partition contains the bootstrap files, firmware, kernel and boot arguments. This partition ***MUST*** stay on the SD-card. This is due to the hardware of the Raspberry Pi vectoring to the SD-card to fetch its bootstrap loader.

The ext4 partition contains the Raspbian Wheezy operating system files.

All of the commands below are to be run from your default user location at the shell prompt, *pi@raspberrypi~$*.

Preamble

An important step is to disable *udevd*, which is responsible for detecting new devices, **before** connecting the USB drive. This step will prevent it trying to automatically mounting the new files systems upon creation.

- **sudo service udev stop**

Step 1: Setup the USB hard drive

First, we need to set up the hard drive with the correct file system.

Another computer system is required to Secure Shell (*ssh*) into the Raspberry Pi. If you are using a PC as the remote (second) computer system I highly recommend *PuTTY* as a *ssh client*. Once a ssh connection has been made to the Raspberry Pi, presence of the USB of the hard drive can be determined with the following command.

- **ls /dev/sda**

  The command should return *''/dev/sda''*. If this is not returned insure that the USB hard drive is connected and spinning. Additionally, try increment ing the last character in *'sda'* string, in the alphabetic order -

ie 'sdb', 'sdc', etc.

After verification of the USB hard drive has been performed we can proceed to partitioning the USB hard drive. For this example we are assuming that */dev/sda* is the desired USB hard drive.

- A Swap partition of 1 GB
- A Root partition of 16 GB
- A /home partition that will consume the remainder of the drive.
- A /ntfs partition of 100 GB

Start by unmounting all partitions on */dev/sda*.

- **sudo umount /dev/sda***

The asterisk (*) after the device name is intentional so as to insure all partitions on this device are unmounted.

A convenient method to perform the partitioning is to utilize the *cfdisk* utility, which provides an easy-to-use character based user interface.

- **sudo cfdisk -z /dev/sda**

You should now see a nice text user interface that lists what's on your disk right now, with a menu on the bottom of the screen.

- Select ***New*** → [Enter].
- Select ***Primary*** → Enter the desired size the Swap partition in MB, 1000 in this case, and choose to place it on the beginning of the disk.
- Select ***Type*** → in the bottom menu → [Enter]. A list of possible types is presented. This is a numbered list, therefore to choose the *Swap* type, enter '**82**', and validate.
- Select the free space with the down arrow key. Repeat the steps above using 16000MB for the size and type *Linux* : '**83**'.
- Calculate the size of the third partition by subtracting 100 GB from the remaining space. This will be the value used for the size of third partition of type *Linux* : '**83**'. Create the partition.
- Repeat to create the fourth partition with the default proposed size and of type *HPFS/NTFS/exFAT* : '**7**'.

You should now write down what partitions the different device files refers to, just in case.

It should be :

- /dev/sda1 → swap (1 GB)
- /dev/sda2 → / (16 GB)
- /dev/sda3 → /home (Balance of the disk)
- /dev/sda4 → /ntfs (100 GB)

In order to commit the configuration to the hard drive select '**Write**' in the bottom menu and confirm by answering '**yes**' to commit the configuration changes and exit *cfdisk*.

The next step is to create an appropriate file system (fs) on each partition.

Create the *SWAP* file system.

- **sudo mkswap /dev/sda1**

Create the *Ext4* file system to host the operating system files

- **sudo mkfs.ext4 /dev/sda2**

Create the *Ext4* file system to host the user data files

- **sudo mkfs.ext4 /dev/sda3**

Create the NTFS file system to host Windows data files

- **sudo mkfs.ntfs /dev/sda4**

Display information on all detected file systems

- **df –h**

Now unmount all of the */dev/sda* partitions.

- **sudo umount /dev/sda***

This is the conclusion of Step 1. The hard drive preparation is now complete.

To verify the successful completion of this step perform the check below.

- **sudo fdisk -l /dev/sda**

  Output should resemble the following:

```
Disk /dev/sda: 320.0 GB, 320072933376 bytes
255 heads, 63 sectors/track, 30913 cylinders, total 625142448 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
Disk identifier: 0x#########

Device    Boot     Start       End        Blocks    Id  System
/dev/sda1              63     1959929     979933+  82  Linux swap/Solaris
/dev/sda2  *      1959930    33206345   15623212+  83  Linux
/dev/sda3        33206355   429835139  198314392+  83  Linux
/dev/sda4       429835140   625142447   97653654   7  HPFS/NTFS/exFAT
```

End → Step 1

Step 2 : Relocate the user and system files

In order to copy the files from the SD-card to the USB 2.0 hard drive create the directories that will hold our mounts then mount the corresponding partitions.

- **sudo mkdir /tmp/sys && sudo mount /dev/sda2 /tmp/sys**
- **sudo mkdir /tmp/home && sudo mount /dev/sda3 /tmp/home**

  Note: These directories will disappear upon reboot, as they reside in the */tmp* directory.

Since we intend to duplicate a mounted file system we are working from use the rsync to perform the file moves.

If ***rsync*** was not previously installed, install it now.

- **sudo apt-get install rsync**

Relocate the user data → */home* & */home/pi* directories.

- **sudo rsync -avh /home/pi/ /tmp/home/pi/**

We use three switches here :

-a → archive mode, preserves owner, group, permissions, links and device information, and perform recursively. This is ***required!***
-v → Verbose output of the program's progress
-h → Human readable output would be good.

For the time being unmount the */tmp/*home directory.

- **sudo umount /tmp/home**

Relocate the system files → / (Top level file system root)

Since the file system we want to copy is currently in use, we must remounting it elsewhere in order to perform a *rsync* on it without falling into a recursive loop.

Note: Do **not** perform a ***rsync*** on the ***/tmp*** folder, since this is the work folder.

Remount the */tmp/sd_sys* file system.

- **sudo mkdir /tmp/sd_sys && sudo mount --bind / /tmp/sd_sys**

Note: Perform the *rsync*, excluding */home*, */tmp*, and optionally */boot* directories.

- **sudo rsync -avh --exclude 'home/pi' --exclude 'tmp/' {--exclude 'boot/'} /tmp/sd_sys/ /tmp/sys/**

  Note: This is where the optional use of the '**-u**' switch would allow the replacement of only the modified files whenever an upgrade is performed on the system files from the SD-card.

Recreate the folders we excluded during *rsync.* Additionally create *mount points* for the /dev/mmcblk0p2 and /dev/sda4 partitions.

- **sudo mkdir {/tmp/sys/boot} /tmp/sys/tmp /tmp/sys/sd-card /tmp/sys/ntfs**

Set the attributes for the */tmp/sys/tmp* directory to what it should be, read, write, and execute for all (drwxrwxrwt), "t" is the "*sticky bit*".

- **sudo chmod –v 1777 /tmp/sys/tmp**

Note: This should eliminate problems with the x-server executing without root permissions.

Set the attributes for the */tmp/sys/ntfs mount point* (read, write, execute).

- **sudo chmod –v 777 /tmp/sys/ntfs**

Verification of the previous operations can be performed by performing the following.

- **df -h /dev/sda2**

  The ouput of *df* should yield at least a couple hundred megabytes.

Additionally the contents of the */tmp/sys* directory can be displayed by performing the following.

- **ls -lah /tmp/sys**

  The output of *ls* should appear similar to that displayed below.

```
total 89K
    drwxr-xr-x 23 root root 4.0K Jun 7 09:15 .
    drwxr-xr-x 23 root root 4.0K Jun 7 09:15 ..
    drwxr-xr-x 2 root root 4.0K Jun 7 21:38 bin
    drwxr-xr-x 2 root root 1.0K Jan 1 1970 boot
    drwxr-xr-x 12 root root 3.2K Jan 1 1970 dev
    drwxr-xr-x 76 root root 4.0K Jun 8 15:18 etc
    drwxr-xr-x 3 root root 4.0K Jun 7 00:45 home
    drwxr-xr-x 12 root root 4.0K May 30 03:25 include
    drwxr-xr-x 11 root root 4.0K Jun 7 23:18 lib
    drwx------ 2 root root 16K Jun 7 08:51 lost+found
    drwxr-xr-x 10 root root 4.0K Jun 7 00:46 media
    drwxr-xr-x 2 root root 4.0K May 7 15:28 mnt
    drwxr-xr-x 4 root root 4.0K Jun 7 09:21 opt
    dr-xr-xr-x 70 root root 0 Jan 1 1970 proc
    drwx------ 4 root root 4.0K Jun 7 22:20 root
    drwxr-xr-x 2 root root 4.0K Jun 7 00:49 sbin
    drwxr-xr-x 4 root root 4.0K Jun 7 00:49 scripts
    drwxr-xr-x 2 root root 4.0K Jul 21 2010 selinux
    drwxr-xr-x 3 root root 4.0K Jun 7 00:45 srv
    drwxr-xr-x 12 root root 0 Jan 1 1970 sys
```

```
drwxrwxrwt 6 root root 4.0K Jun 8 15:17 tmp
drwxr-xr-x 10 root root 4.0K Jun 7 00:39 usr
drwxr-xr-x 13 root root 4.0K Jun 7 00:39 var
```

End → Step 2

Step 3: Setup the new Swap file and System files

In order for Debian (Raspbian) Linux to identify and mount at system start the newly created Swap, Operating System (root), and User data (home) file systems their descriptions must be entered in */tmp/sys/etc/fstab*.

- **sudo nano /tmp/sys/etc/fstab**

Initial configuration should resemble that displayed below.

```
proc            /proc  proc    defaults          0  0
/dev/mmcblk0p1  /boot  vfat    defaults          0  2 # SD Boot partition
/dev/mmcblk0p2  /      ext4    defaults,noatime  0  1 # SD System partition
```

Edit the */etc/*fstab file to resemble that displayed below.

```
proc            /proc    proc    defaults          0  0
/dev/mmcblk0p1  /boot    vfat    defaults          0  3 # SD Boot partition
/dev/mmcblk0p2  /sd-card ext4    defaults          0  2 # SD System partition
/dev/sda1       none     swap    sw                0  0 # HD Swap partition
/dev/sda2       /        ext4    defaults,noatime  0  1 # HD System partition
/dev/sda3       /home    ext4    defaults,noatime  0  2 # HD User data partition
/dev/sda4       /ntfs    ntfs-3gumask=000           0  2 # HD NTFS partition
```

- WriteOut → **^O**
- Exit → **^X**

Unmount the */tmp/sys, /tmp/home,* and */tmp/sd_sys* file systems.

- **sudo umount /tmp/sys /tmp/home /tmp/sd_sys**

An error related to */tmp/home* not being mounted may be displayed. It should already be unmounted, thus the error message is just informational.

End → Step 3.

Step 4: Prepare the Raspberry Pi for the new location of the operating system.

The location of the partition containing the operating systems files for the Raspberry Pi is within the */boot/cmdline.txt* file. Edit this file now to redirect the location of the operating system directory from the SD-card to the USB hard drive.

- **sudo nano /boot/cmdline.txt**

  Replace → ***root=/dev/mcblk0p2*** with ***root=/dev/sda2***

  - WriteOut → **^O**
  - Exit → **^X**

End → Step 4.

Step 5: Reboot the Raspberry Pi to implement the changes.

- **sudo reboot**

Verify that the new file systems from the USB hard drive are now utilized.

- **df -h**

  The output should resemble the output below.

Filesystem	Size	Used	Avail	Use%	Mounted on
rootfs	7.4G	1.6G	5.5G	23%	/
/dev/root	7.4G	1.6G	5.5G	23%	/
devtmpfs	212M	0	212M	0%	/dev
tmpfs	44M	424K	44M	1%	/run
tmpfs	5.0M	0	189M	0%	/run/lock
tmpfs	189M	0	189M	0%	/run/shm
/dev/mmcblk0p1	56M	19M	38M	34%	/boot
/dev/mmcblk0p2	1.8G	1.5G	201M	88%	/sd-card
/dev/sda3	194G	191M	184G	1%	/home

```
/dev/sda4            94G   89M  94G     1% /ntfs
```

Verify that the swap file in use is the partition on the hard drive.

- **sudo swapon -s**

  The output from the query should resemble the display below.

```
Filename        Type        Size        Used  Priority
/dev/sda1       partition   514044      0     -1
/var/swap       file        102396      0     -2
```

==========

***** Important:** If ***startx*** will only function for *root*, perform the following on the /dev/sda2 partition serving as the root (/) of the file system. Additionally, it is also serving as the operating system partition. The leading "1" in the permissions is the "*sticky bit*", "t".

- **sudo chmod –v 1777 /tmp**

==========

End → Step 5

[How-to] → Complete