```
Disk /dev/mmcblk0: 7948 MB, 7948206080 bytes
4 heads, 16 sectors/track, 242560 cylinders, total 15523840 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: 0x000dbfc6

      Device Boot          Start         End      Blocks   Id  System
/dev/mmcblk0p1            8192      122879       57344    c  W95 FAT32 (LBA)
/dev/mmcblk0p2          122880    15523839     7700480   83  Linux
```

USB adaptor of some kind, it's most likely going to be /dev/sdx where 'x' is the digit representing the highest device of this kind.
To find out, run the fdisk -l command. The tail end of the output from this command can be seen at the top of this page. I can recognise the card by the size, 8Gb (or 7948Mb as listed above) which is the only device or partition that I have of this size. Also shown are the two partitions on the device and seeing that one of them is small and W95 FAT32 (LBA) and the other is Linux, then I am certain that this is my SD card.

In the above, the suffixes 'p1' and 'p2' in the Device Boot column indicate the partitions on the card. The card itself is the device named /dev/mmcblk0 and we need the device name, not the partition names.

Make a Backup

To make a backup switch to your backup directory, where you intend to keep the copy of your SD card, and run the following command, typed all on one line:

```
$ dd if=/dev/mmcblk0 of=Rpi_8gb_backup.
img bs=2M
```

The output from the above command will be similar to the following:

```
3790+0 records in
3790+0 records out
7948206080 bytes (7.9 GB) copied,
1369.42 s, 5.8 MB/s
```

That's it. The whole 8Gb SD card has been (slowly) copied to my computer. How do I know it worked? Type ls -l -h and you will see the new file name and size.

Compressing the Backup

Once created and checked, the backup image can be compressed to save space. This is a simple matter of using the gzip command:

```
$ gzip -9 Rpi_8gb_backup.img
```

This command attempts to perform maximum compression. This uses a lot of CPU but will result in the smallest output file. You may adjust the trade off between CPU and file sizes by changing the '-9' option to a smaller number. The quickest compression will be obtained with the '-1' option at the expense of the file size being larger.

The use of gzip in this manner requires that you have enough space to save both the full size image file and the compressed image for as long as the gzip command is running. Once complete, the full sized file will be deleted leaving only a compressed file, named as per the original file name but with an additional '.gz' extension. In this example my compressed image file would be Rpi_8gb_backup.img.gz.

Compression on the Fly

If you only wish to make a backup or if you don't have enough available space to hold both the full sized image and the compressed image, then you may wish to consider compressing on the fly.

Unless otherwise instructed, the dd command defaults its output file to be the console. You can use this feature to pipe the output through gzip and create a compressed image file in a single step. You will not need as much disc space because the full sized image file is never created, only the smaller compressed one. The command to do this is:

```
$ dd if=/dev/mmcblk0 bs=2M | \
gzip -9 - > Rpi_8gb_backup.img.gz
```

The trailing "\" must be at the very end of the