```
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
```

sera plutôt /dev/sdx où "x" est le n° représentant le périphérique le plus élevé de ce type.
Lancez la commande fdisk -l. La fin de l'affichage est reproduite au haut de cette page. Je peux reconnaitre la carte grâce à sa taille, 8 Go (ou 7948 Mo ci-dessus), car c'est le seul périphérique à avoir cette taille. Sont aussi indiquées les deux partitions, une petite en FAT32 (LBA) W95 et une Linux, ce qui confirme que c'est bien ma carte SD.

Plus haut, les suffixes "p1" et "p2" dans la colonne Device Boot désignent les partitions. La carte elle-même est nommée /dev/mmcblk0 et c'est le nom dont nous avons besoin, pas celui des partitions.

Faire une sauvegarde

Allez dans votre répertoire de sauvegarde, où vous voulez placer la copie de votre carte, et tapez la commande suivante sur une seule ligne :

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

Le résultat ressemblera à ceci :

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

Voilà. La carte SD 8 Go a été (lentement) copiée en entier sur mon ordinateur. Comment savoir si ça a marché ? Tapez ls -l -h et vous verrez le nom du nouveau fichier et sa taille.

Compression de la sauvegarde

Une fois créée et vérifiée, l'image peut être compressée pour gagner de la place. Il suffit d'un simple appel à la commande gzip :

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

Celle-ci réalisera une compression maximale. Le CPU sera très sollicité mais le fichier généré sera le plus compact possible. Vous pouvez trouver un compromis entre CPU et taille de fichier en changeant l'option "-9" par un nombre inférieur. Une compression rapide sera obtenue avec "-1" au détriment d'une taille de fichier plus importante.

Utiliser gzip de cette façon oblige à avoir assez d'espace pour stocker à la fois l'image complète et l'image compressée pendant l'exécution de la commande. Une fois la compression terminée, le fichier d'origine sera supprimé pour ne laisser que la version compressée avec le suffixe ".gz". Dans cet exemple, mon image compressée s'appelle Rpi_8gb_backup.img.gz.

Compression à la volée

Si vous ne voulez faire que la sauvegarde ou si vous n'avez pas assez d'espace libre pour les deux fichiers, alors vous pourriez être intéressés par la compression à la volée.

Sauf indication contraire, dd envoie par défaut sa sortie sur la console. Vous pouvez utiliser cette caractéristique pour faire un tube vers gzip et créer un fichier compressé en une seule étape. Vous n'aurez pas besoin d'autant d'espace disque car l'image de taille complète ne sera jamais créée, il n'y aura que la plus petite compressée. La commande est :

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

Le "\" final doit être en fin de ligne, juste avant l'appui sur Entrée. Il indique au shell que la commande n'est pas finie et qu'elle se poursuit ligne suivante.

Suite sur la page suivante...