

SYS

Espace disque

File system

Métadonnées qui définissent le formatage.

C'est en quelque sorte la façon dont on organise les fichiers au sein d'un disque.

ex: **EXT**(linux), **NTFS**(windows), **FAT**

Partition

Avant que son espace puisse être utilisé, un disque doit être divisé en partitions.

Ce sont en quelque sorte des régions définies du disques.

Partition	Nom	Système de fichiers	Point de montage	Étiquette	Taille	Utilisé	Inutilisé	Drapeaux
/dev/nvme0n1p1	Basic data partition	ntfs		Récupération	529.00 Mio	449.87 Mio	79.13 Mio	hidden, diag
/dev/nvme0n1p2	EFI system partition	fat32	/boot/efi		100.00 Mio	36.19 Mio	63.81 Mio	boot, esp
/dev/nvme0n1p3	Microsoft reserved partition	inconnu			16.00 Mio	---	---	msftres
/dev/nvme0n1p4	Basic data partition	ntfs			155.38 Gio	132.26 Gio	23.13 Gio	msftdata
/dev/nvme0n1p5		ntfs			596.00 Mio	506.47 Mio	89.53 Mio	hidden, diag
non alloué		non alloué			1.00 Mio	---	---	
/dev/nvme0n1p6		ntfs			546.00 Mio	457.59 Mio	88.41 Mio	hidden, diag
/dev/nvme0n1p7		ext4	/		81.34 Gio	20.52 Gio	60.83 Gio	

disque SSD et ses partitions

On y voit principalement:

- Le disque SSD **/etc/nvme0n1** de **238.47 Gio**
- Une partition **nvme0np2** type **fat32** de **100 Mio** possédant le flag **boot**
- Une partition **nvme0np4** type **ntfs** de **155.38 Gio** sur laquelle est installé Windows
- Une partition **nvme0np7** type **ext4** de **81.34 Gio** sur laquelle est installé Linux

Partition DOS

La partition DOS permet de placer plusieurs systèmes de fichiers ou systèmes d'exploitation.

MBR

LE MBR compose la première partie d'un système de partition DOS.

Le MBR est lui-même divisé en 3 parties:

- Un programme d'amorçage (sur lequel on boot) -> **446 bytes**

- La table de partitions (il peut y avoir 4 partitions) -> 4x16 bytes
- Un code magique de boot mis par le BIOS et chargé en RAM (0x55AA) -> 2 bytes

Table de partitions

Indique si est bootable (1)	Adresse CHS du premier sector (2)	Type de Système de Fichiers (FAT, NTFS..) (3)	Adresse CHS du dernier sector (4)	Adresse LBA du premier sector (5)	Taille en Secteurs de la partition (6)
1 byte (0x80 si oui, 0x00 sinon)	3 bytes	1 Byte	3 Bytes	4 bytes	4 bytes

schéma d'une table de partitions

- (1) Marque de boot
- (2) (4) Il peut y avoir des bytes appelés CHS, mais à priori c'est obsolète
- (3) Informations sur le système de fichiers
- (5) LBA du premier secteur absolu de cette partition
- (6) Nombre de secteurs dans la partition

Chaque partition possède un **BR** (boot record), premier secteur qui peut également contenir un programme d'ammorçage.

LBA

Logical Bloc Address: Numéro de secteur codé sur 4 bytes.

Partition étendue

Le MBR est limité à 4 partitions, ce qui n'est pas pratique pour certaines personnes qui en désirent plus. On a donc créé la partition étendue pour contourner ce problème.

Rappels

echo

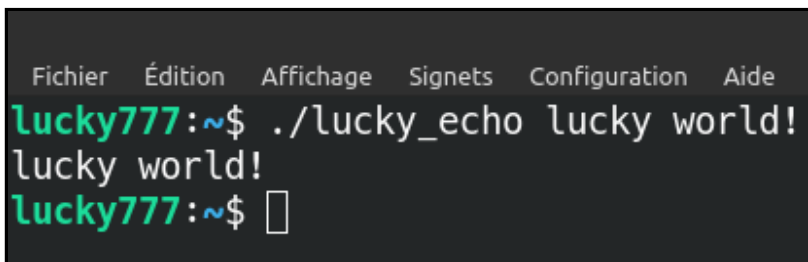
La commande **echo** affiche à l'écran.

```
Fichier  Édition  Affichage  Signets  Configura
lucky777:~$ echo lucky world!
lucky world!
lucky777:~$
```

echo

Réécrivons la commande `echo` en `c` dans le fichier `lucky_echo.c`:

```
//File: lucky_echo.c
int main(int argc, char* argv[]) {
    for(int i=1; i<argc; i++) {
        printf("%s ", argv[i]);
    }
    printf("\n");
}
```

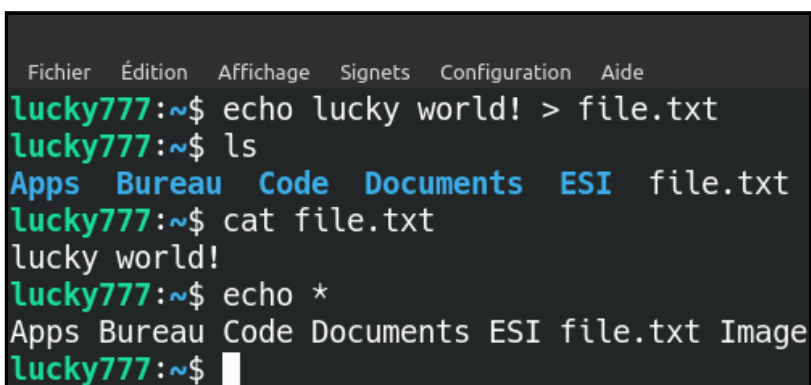


```
Fichier  Édition  Affichage  Signets  Configuration  Aide
lucky777:~$ ./lucky_echo lucky world!
lucky world!
lucky777:~$
```

lucky_echo

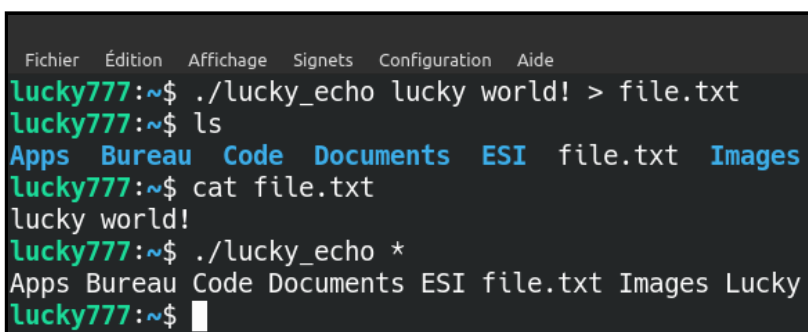
Essayons d'enregistrer le résultat dans un fichier au lieu d'afficher dans le terminal:

```
echo lucky world! > file.txt #On enregistre dans file.txt
ls #On affiche les fichiers du répertoire courant
cat file.txt #On affiche le contenu du fichier file.txt
echo * #On affiche les fichiers du répertoire courant
```



```
Fichier  Édition  Affichage  Signets  Configuration  Aide
lucky777:~$ echo lucky world! > file.txt
lucky777:~$ ls
Apps  Bureau  Code  Documents  ESI  file.txt
lucky777:~$ cat file.txt
lucky world!
lucky777:~$ echo *
Apps Bureau Code Documents ESI file.txt Image
lucky777:~$
```

echo > file.txt



```
Fichier  Édition  Affichage  Signets  Configuration  Aide
lucky777:~$ ./lucky_echo lucky world! > file.txt
lucky777:~$ ls
Apps  Bureau  Code  Documents  ESI  file.txt  Images
lucky777:~$ cat file.txt
lucky world!
lucky777:~$ ./lucky_echo *
Apps Bureau Code Documents ESI file.txt Images Lucky
lucky777:~$
```

`lucky_echo > file.txt`

On remarque que `>` et `*` fonctionnent dans `echo` ET `lucky_echo`, pourtant on ne l'a pas codé dans notre fichier `lucky_echo.c`!

En effet, c'est le terminal lui même qui effectue ces actions avec notre commande.

`echo lucky world! > f` redirige le résultat dans un fichier

`lucky_echo lucky world! > f` enregistre le stdout du shell dans un fichier

Niveaux du manuel

- `man 1`: Command User
- `man 2`: Appels Système
- `man 3`: Librairies C
- `man 4`: Fichier
- `man 7`: Livre
- `man 8`: Command Admin

Variables en C

```
static int s2; //Variable stockée dans le .bss
int a; //Variable globale NON déclarée stockée dans le .bss
int b = 4; //Variable globale déclarée stockée dans le .data
{
    int i; //Variable locale stockée sur la pile
    static int s1; //Variable stockée dans le .bss
    char c* = malloc(1); //c stocké sur la pile
                        //contenu de c stocké sur le tas
}
```