http://clx.asso.fr/spip/?Recuperer-les-donnees-d-une-clef

Récupérer les données d'une clef USB

- Documentations - Installation / Administration de base de Linux -

Date de mise en ligne : lundi 28 juin 2004

Ma clef USB (contenant 384 Mo de données précieuses) a fait un vol plané. Linux ou Windows ont eu la même réaction : partition non formatée. C'est toujours à ce moment là qu'on se rend compte que la derniÚre sauvegarde a plus d'un mois...

Instant idéal pour prendre le temps d'étudier les commandes linux de sauvegarde, copie, exploration et tuttiquanti.

Cet article n'a pas pour ambition de vous donner une méthode infaillible ou efficace pour récupérer les données perdues mais de vous présenter par un exemple vécu quelques outils simples et utiles pour sonder sa clef.

Tout d'abord la première chose à faire est de ne pas jouer les apprentis sorciers : ne pas détruire encore plus les données qui s'y trouveraient encore avec des logiciels spécialisés dont j'ignore les actions et performances.

Nous travaillerons donc sur une copie de la clef USB.

dd

« dd » est un outil de copie.Dans la mandrake il est installé d'office. Sa syntaxe est assez simple :

```
dd if=fichier_d'entrée of=fichier-de-sortie
```

Les options les plus courantes sont :
bs=taille_d'un_bloc-secteur
count=nombre_de_blocs_à_lire

Par exemple, la commande suivante crée une copie des 512 premiers octets de la partition 1 du disque 1 sur la première nappe IDE. C'est le "Master Boot Record" (MBR) ou "secteur de démarrage" :

```
dd if=/dev/hda of=./lesecteurmbr.dd bs=512 count=1
```

Notre clef étant reconnue comme un périphérique "sda", nous copierons donc sa première partition dans un fichier.
Notre commande sera donc ici :

```
dd if=/dev/sda1 of=./maclef.dd
```

Ce fichier maclef.dd fait quasiment 488 Mo, ça tombe bien puisqu'il s'agit presque de la taille de ma clef, donc de la partition unique qui s'y trouve.
Attention, j'ai voulu servir l'option bs de façon très réduite (bs=1) car j'ignorais la taille réelle des secteurs qui s'y trouvaient. Quelle mauvais idée : la copie était d'une lenteur... Finalement après plusieurs essais (512, 2048...), j'ai oté ce paramêtre car cela ne changeait rien sur le fichier résultant.

mc

mc [1] est le couteau suisse du mode console. Il permet entre autres fonctions de naviguer parmi les répertoires, visualiser les fichiers textes, changer les droits, copier, effacer, détruire...

Nous naviguons donc dans la liste des fichiers pour sélectionner maclef.dd. Un appui sur la touche F3 permet d'en visualiser le début et de me rendre compte qu'un octet sur deux est un point. Un appui sur la touche F4 permet de visualiser en mode hexadécimal. Je me rends alors compte qu'un octet sur deux est à zéro.

A force de regarder ces caractères bizarres, j'arrive à repérer N.N.M. . . . précédé de F.T.2 puis r.s. .a.y. .k.y. Un cruciverbiste aurait probablement complété plus vite que moi les lettres manquantes : "NONAME", "FAT32" ainsi que "press any key". A l'aide de la touche F6, je recherche le texte complet et le trouve un peu plus loin. Je venais de redécouvrir que la FAT est inscrite deux fois sur un disque ! La même séquence commence sur la ligne C00 mais au complet. Plus de pointillé.

"C00", Kcalc [2] (et son bouton "base") me transforme cette valeur en nombre décimal : 3072 octets.

Visiblement, seuls les 512 premiers octets de ma clef ont ce syndrôme du pointillé (ce caractère nul, une fois sur deux), je vais donc extraire 512 octets à partir du 3072 et les remettre à la place des 512 premiers.

Encore une fois dd va venir à mon secours. Un "man dd" me donnera les informations nécessaires.
3072 c'est 6 fois 512 octets donc je dois extraire le 7ème bloc d'octets :

```
dd if=./maclef.dd of=./fatok.dd bs=512 count=1 skip=6
```

Détail de la commande :

– if (Input File = fichier duquel j'extrais)
– of (Output File = résultat de ma copie)
– bs=512 (BlockSize = nombre d'octets à lire d'un coup)
– count=1 (nombre de "tranches" de 512 octets à lire)
– skip=6 (mais en laissant de côté les 6 premières tranches de 512 octets soit 3072)

Avec mc, je vérifie que j'ai bien récupéré le bloc voulu puis on passe à l'intégration dans le fichier de départ mais cette fois-ci on le positionne au début du fichier :

```
dd if=./fatok.dd of=./maclef.dd bs=512 count=1 conv=notrunc
```

L'option supplémentaire *conv=notrunc* précise à dd de laisser maclef.dd à sa taille d'origine donc de ne pas le réduire à ce qu'on vient d'écrire. En d'autres termes, on écrase les 512 premiers octets mais on laisse intact tout ce qui est derrière. Sans cette option, maclef.dd ne ferait plus que 512 octets.

Arrivé là, j'ai procédé à la copie du fichier sur ma clef et j'ai croisé les doigts pour que ça marche... C'est faisable mais risqué ! Ensuite j'ai trouvé comment j'aurais pu éviter ce risque inconsidéré : mount et loop.

mount

C'est une commande que les mandrakephiles n'utilisent que rarement : les confortables supermount, hotplug et diskdrake se chargement généralement de maîtriser mount à notre place. Cette commande permet de "monter" un périphérique de type bloc, c'est-à-dire de passer d'une lecture bloc par bloc (octet par octet) à une lecture utilisant le système de fichier qui lui est associé (c'est-à-dire la reconnaissance des fichiers, dossiers ou répertoires, débuts et fins de fichiers, attributs...).

J'ai souvent entendu dire que "*tout est fichier dans linux*" donc pourquoi ne pas utiliser notre fichier maclef.dd comme s'il s'agissait d'une vraie clef USB ?

après quelques tâtonnements, je trouve la formule qui permet à *mount* de faire de mon fichier, une pseudo clef USB.

D'abord créer un point de montage car mount m'a dit que le point de montage n'existait pas :

```
mkdir /mnt/mapseudoclef
```

Ensuite mount me dira que ce n'est pas un périphérique "bloc" et qu'il me faut utiliser l'option -o loop. Vu comme ça c'est pas dur linux ! On essaye, et quand ça ne va pas, on suit la proposition !

```
mount -t vfat /home/beuz/maclef.dd /mnt/mapseudoclef -o loop
```

Les options sont :
– *t vfat* pour signifier que nous sommes en présence d'un système de fichier FAT (iso9660 permettrait ainsi d'explorer le contenu d'une image ISO)
– origine de mon montage (ici mon fichier)
– endroit où le système de fichier est intégré à l'arborescence existante (ici /mnt/mapseudoclef)
– *o loop* qui permet d'utiliser ce fichier comme un périphérique "bloc"

Plus de message d'erreur, je vais voir avec mc ce qu'il y a dans le répertoire /mnt/mapseudoclef et bingo ! Voilà mes fichiers. Je regarde l'intérieur d'un fichier et tout va bien.

Ensuite il ne restera plus qu'à recopier maclef.dd sur /dev/sda1 ou bien formater la clef et recopier les fichiers avec un gestionnaire de fichiers.

Post-scriptum :

Je vous vois venir : "Moi, ça ne marche pas avec ma clef qui est passée dans la machine à laver !" Oui, bon... Je n'avais pas la prétention de fournir la solution ultime mais simplement de donner quelques pistes en s'appropriant par cette occasion quelques commandes trÚs pratiques de linux.

---

[1] il est installé par défaut avec la mandrake 10 mais pour la 9.2, taper "*urpmi mc*". Il existe probablement aussi en deb, tar.gz,etc. Plus d'info : http://www.ibiblio.org/mc/

[2] calculatrice sous kde mais n'importe quelle calculatrice capable de travailler en base 16 convient

