/dev/loop, /dev/loop-control

périphériques loop

Le périphérique loop est un périphérique loop qui map ses blocks de données non pas sur un périphérique physique comme un disque dur, mais aux blocks d'un fichier régulier dans un système de fichier ou dans un autre périphérique block. Cela peut être utile par exemple pour fournir un périphérique block pour une image de système de fichier stocké dans un fichier, pour qu'il puisse être monté avec la commande mount.

On peut faire :

```
dd if=/dev/zero of=file.img bs=1MiB count=10
sudo losetup /dev/loop4 file.img
sudo mkfs -t ext4 /dev/loop4
sudo mkdir /myloopdev
sudo mount /dev/loop4 /myloopdev
```

Un fonction tranfert peut être spécifiée pour chaque périphérique loop pour chiffrer/déchiffrer.

Depuis Linux 3.1, le kernel fournis le périphérique /dev/loop-control, qui permet à une application de trouver dynamiquement un périphérique libre, et d'ajouter et supprimer des périphériques du système.

Le programme ci-dessous utilise le périphérique /dev/loop-control pour trouver un périphérique loop, l'ouvre, ouvre un fichie à utiliser comme stockage sous-jacent, puis l'associe avec le périphérique loop. La session shell suivante démontre l'utilisation du programme :

```
> dd if=/dev/zero of=file.img bs=1MiB count=10
10+0 records in
10+0 records out
10485760 bytes (10 MB) copied, 0.00609385 s, 1.7 GB/s
> sudo ./mnt_loop file.img
loopname = /dev/loop5
```

Programme source :

```
#include <fcntl.h>
#include <linux/loop.h>
#include <sys/ioctl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#define errExit(msg) do { perror(msg); exit(EXIT_FAILURE); \
    } while (0)

int
main(int argc, char *argv[])
{
 int loopctlfd, loopfd, backingfile;
 long devnr;
 char loopname[4096];

 if (argc != 2) {
  fprintf(stderr, "Usage: %s backing-file\n", argv[0]);
  exit(EXIT_FAILURE);
 }
```

```
 loopctlfd = open("/dev/loop-control", O_RDWR);
 if (loopctlfd == -1)
  errExit("open: /dev/loop-control");

 devnr = ioctl(loopctlfd, LOOP_CTL_GET_FREE);
 if (devnr == -1)
  errExit("ioctl-LOOP_CTL_GET_FREE");

 sprintf(loopname, "/dev/loop%ld", devnr);
 printf("loopname = %s\n", loopname);

 loopfd = open(loopname, O_RDWR);
 if (loopfd == -1)
  errExit("open: loopname");

 backingfile = open(argv[1], O_RDWR);
 if (backingfile == -1)
  errExit("open: backing-file");

 if (ioctl(loopfd, LOOP_SET_FD, backingfile) == -1)
  errExit("ioctl-LOOP_SET_FD");

 exit(EXIT_SUCCESS);
}
```