Image mentale fichiers

Il s'agit d'un exercice corrigé, ne tournez la page que lorsque vous avez répondu vous-même à la question sur une feuille.

Question

On branche une clef USB et le fichier `/dev/sdq` apparaît dans l'arborescence. Puis on tape les commandes suivantes :

```
# mkdir /mnt/{a,b,c}
# echo a > /mnt/a/aa
# parted /dev/sdq mklabel gpt
# parted /dev/sdq mkpart primary 2048s 2GiB
# parted /dev/sdq mkpart primary 2GiB 4GiB
# mkfs -t ext4 /dev/sdq1
# mkfs -t ext4 /dev/sdq2
# mount /dev/sdq1 /mnt/a
# mount /dev/sdq2 /mnt/b
# mkdir /mnt/a/{rep,rap}
# touch /mnt/a/rep/vide
# ln -s /mnt/a/rep/vide /mnt/b/plouf
# cp -l /mnt/a/rep/vide /mnt/a/rep/revide
# cp /mnt/a/rep/vide /mnt/a/rap/encorevide
```

Sans exécuter les commandes précédentes, donnez une représentation graphique de la situation obtenue. Dessinez au moins les block devices touchés, les filesystems créés, l'arborescence (en particulier la partie contenant les fichiers crées), et leurs relations.

Réponse possible

Faites votre propre dessin avant de tourner la page.

Réponse possible

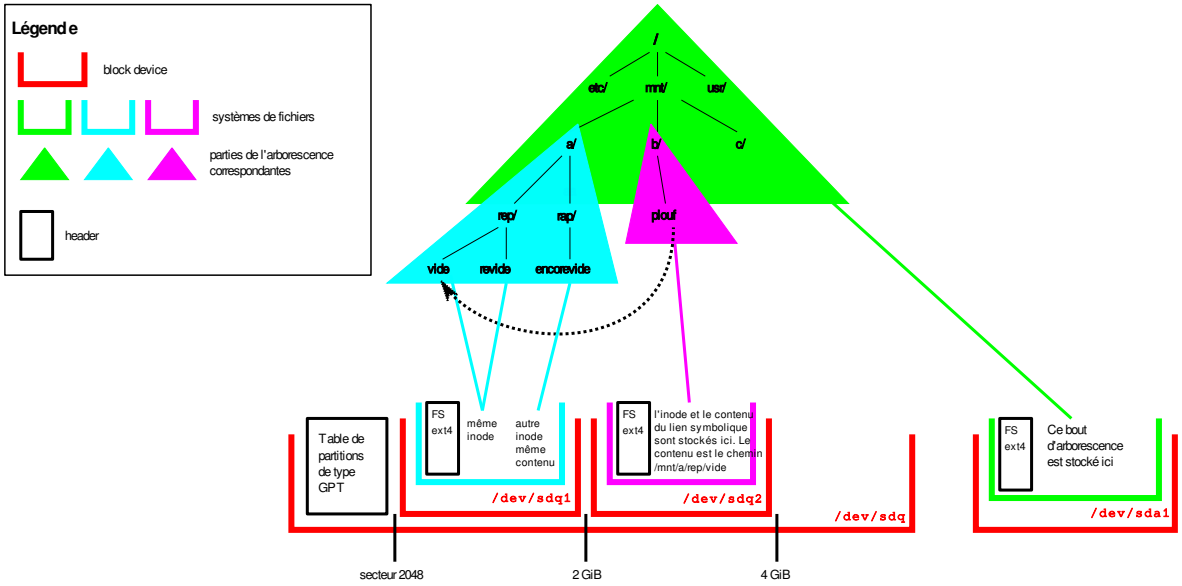
J'insiste. Faites votre propre dessin avant de tourner la page.

Réponse possible

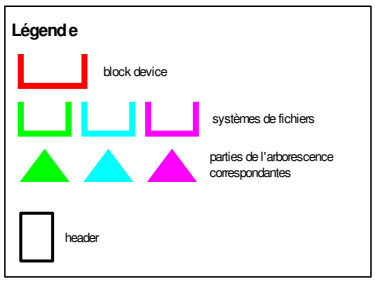
Chaque page montre le résultat d'une commande, pour voir les dessins bien alignés, page après page (comme une animation), utilisez

- la touche "flèche droite" si vous utilisez `atril`, `evince`, `okular` ou `qpdfview`
- la touche "page suivante" si vous utilisez `mupdf` ou `zathura`
- la touche "n" si vous utilisez `xpdf`

Réponse possible (il ne s'agit que d'une proposition)

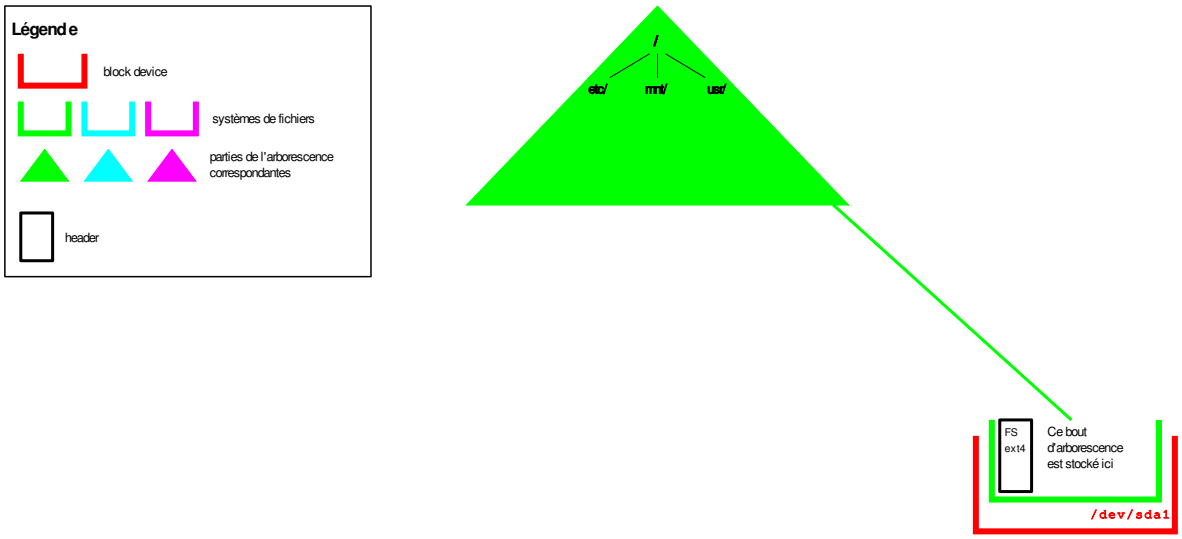


On va décomposer commande par commande



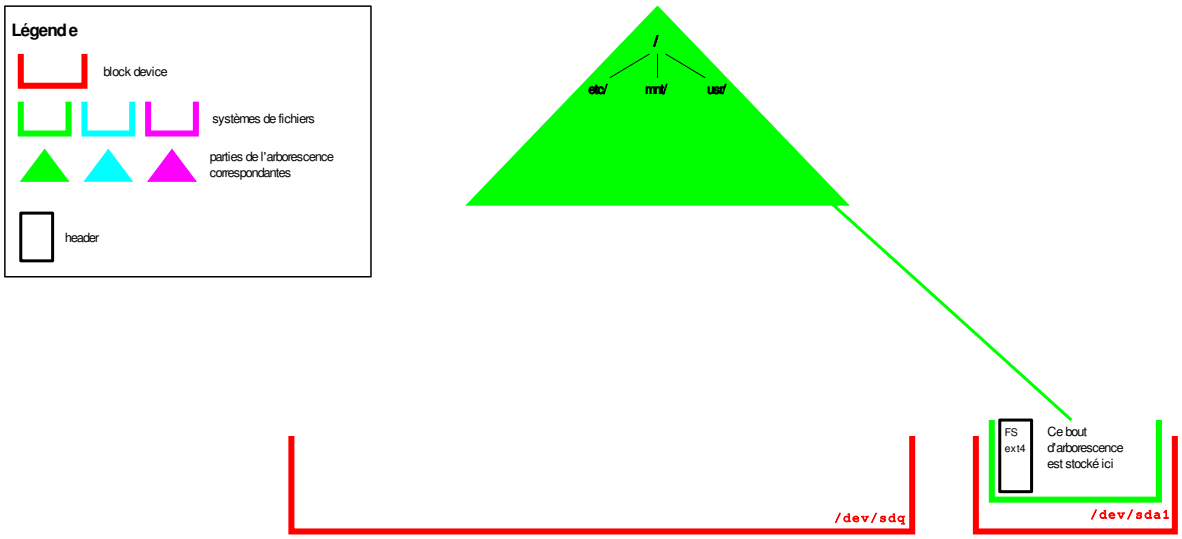
On commence par la légende. On va décrire la situation selon les 3 couches vues en cours : périphériques en mode bloc, systèmes de fichiers, arborescence.

On ne part pas de rien



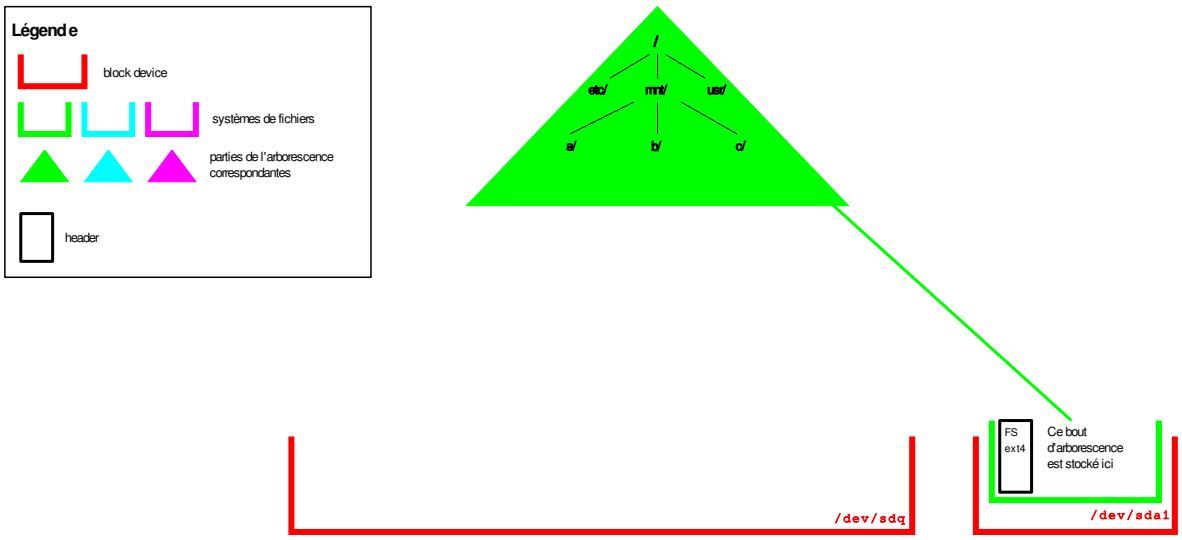
La première chose à comprendre, c'est que lorsqu'on branche la clef USB, on a un système en cours de fonctionnement, en particulier l'arborescence contient déjà de nombreux fichiers. En particulier, un système de fichiers est déjà monté sur la racine `/` pour nous offrir un système de base. Dans notre exemple, nous avons considéré que c'était le système de fichiers installé dans la permière partition du premier disque dur `/dev/sda1` qui était monté sur le répertoire racine `/` (représenté en vert).

On branche la clef USB



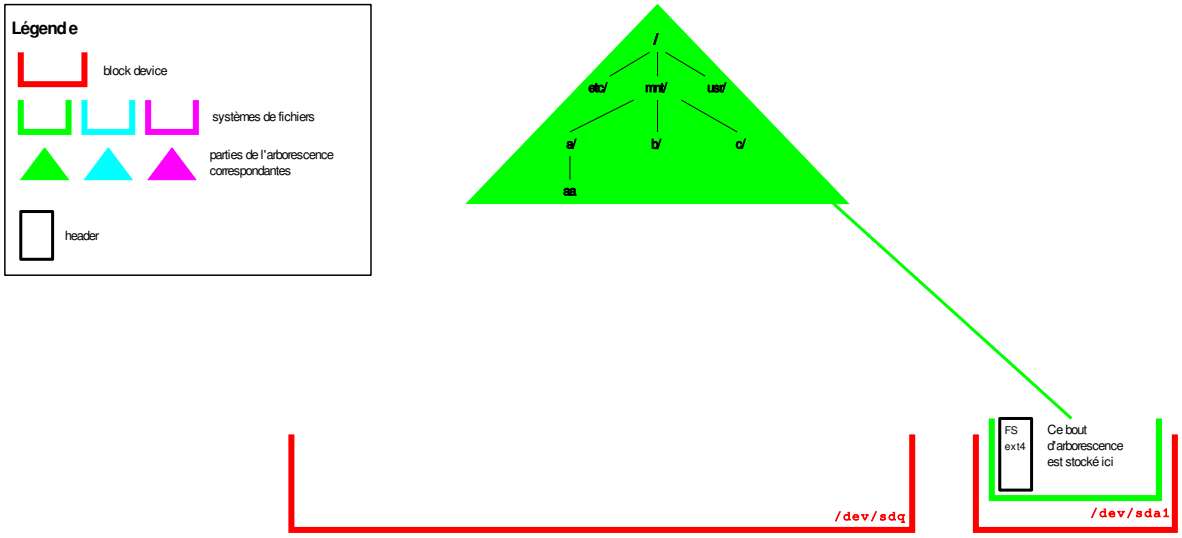
Au moment où on branche la clef USB, le noyau, les drivers et `udev` font en sorte de nous donner accès au contenu de la clef via un fichier périphérique en mode bloc `/dev/sdq` (la lettre `q` a été choisie pour éviter les ennuis aux personnes qui seraient tentées de copier les commandes dans un terminal).

```
# mkdir /mnt/{a,b,c}
```



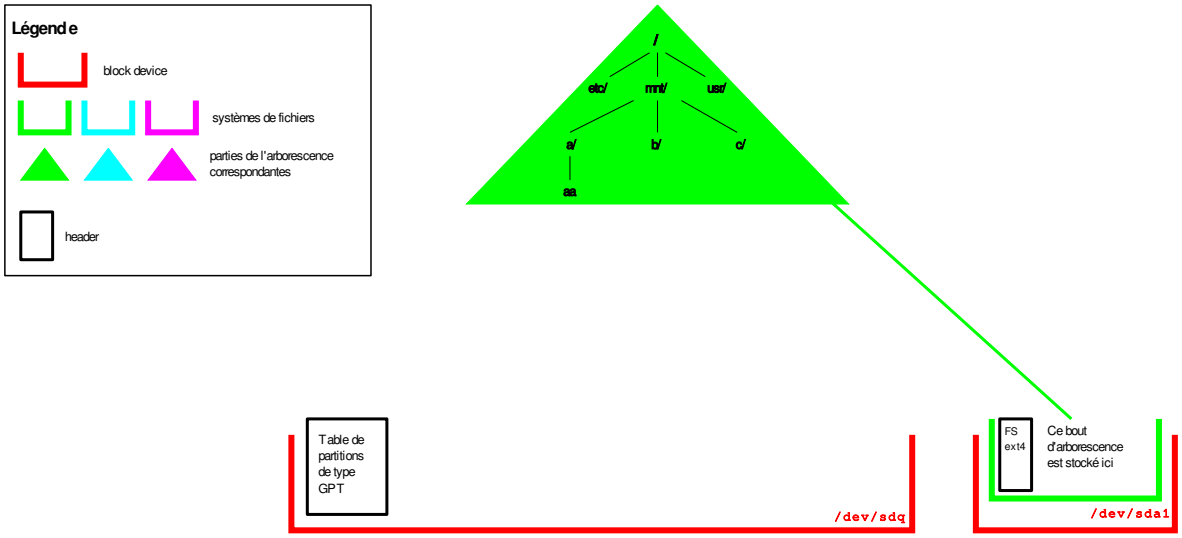
On crée trois répertoires `/mnt/a`, `/mnt/b` et `/mnt/c`. Les informations relatives à ces répertoires (inode et contenu) sont stockés dans le système de fichiers de couleur verte.

```
# echo a > /mnt/a/aa
```



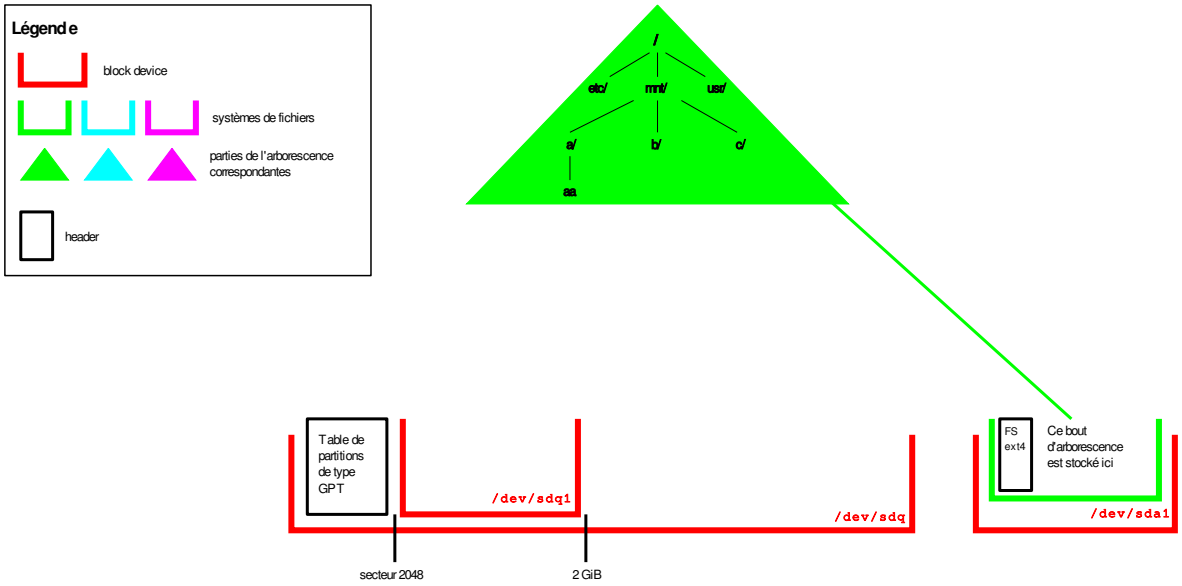
On crée un fichier `/mnt/a/aa` ayant pour contenu la chaine de caractères `"a"` (en toute rigueur `"a\n"`). Les informations relatives à ce fichier (inode et contenu) sont stockées dans le système de fichiers de couleur verte.

```
# parted /dev/sdq mklabel gpt
```



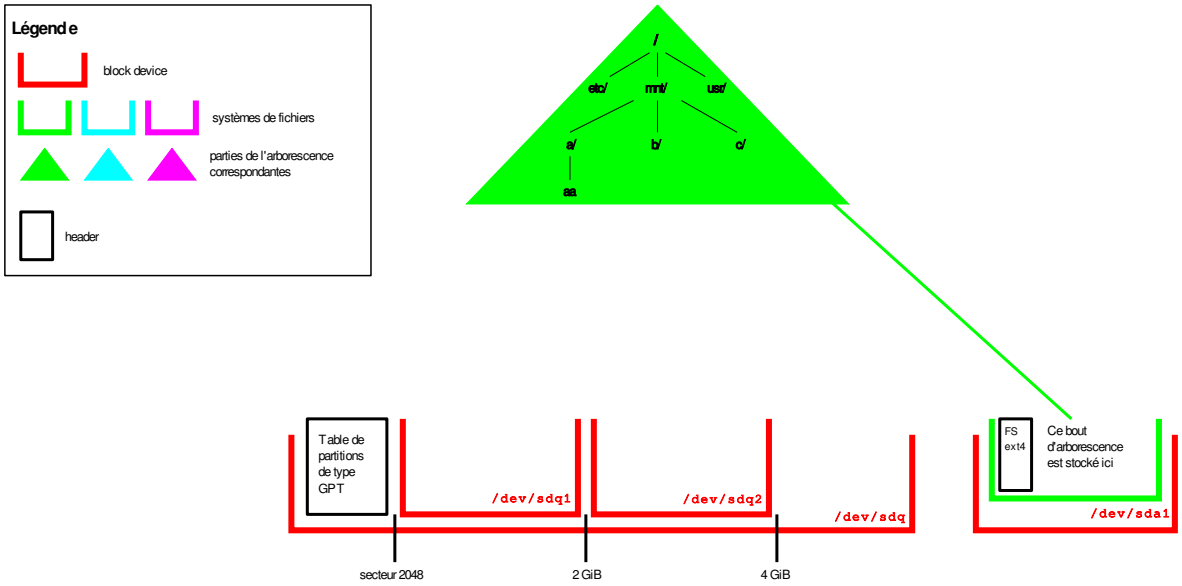
On crée une table de partitions de type `GPT` sur la clef USB (block device `/dev/sdq`). C'est dans cette table que seront inscrites les informations sur les partitions.

```
# parted /dev/sdq mkpart primary 2048s 2GiB
```



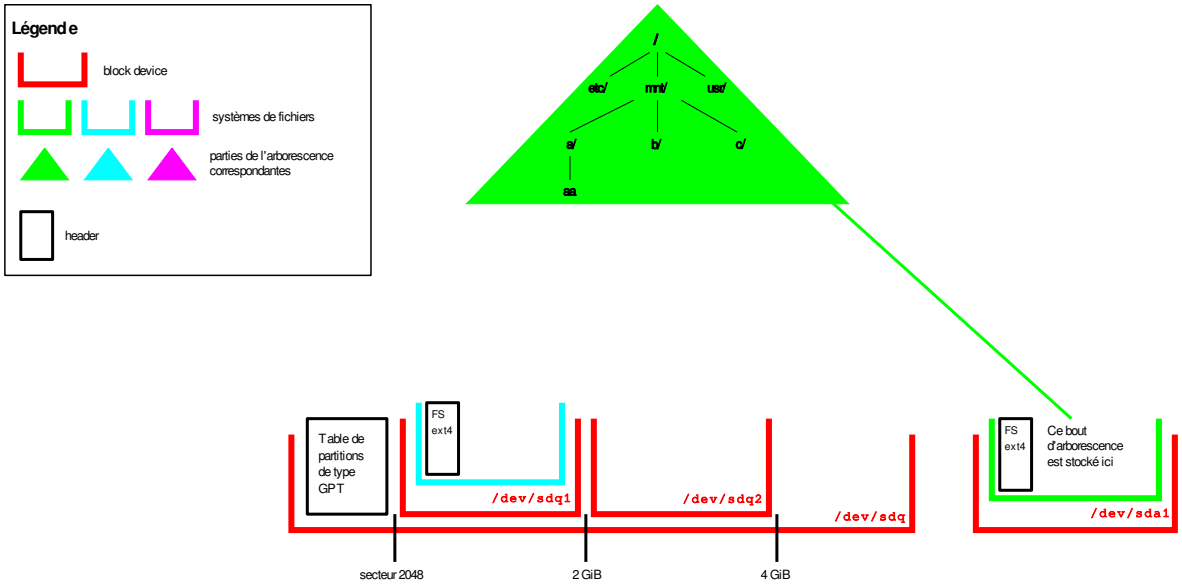
On crée une première partition (block device `/dev/sdq1`) sur la clef USB, elle s'étend du secteur `2048` à `2 GiB`.

```
# parted /dev/sdq mkpart primary 2GiB 4GiB
```



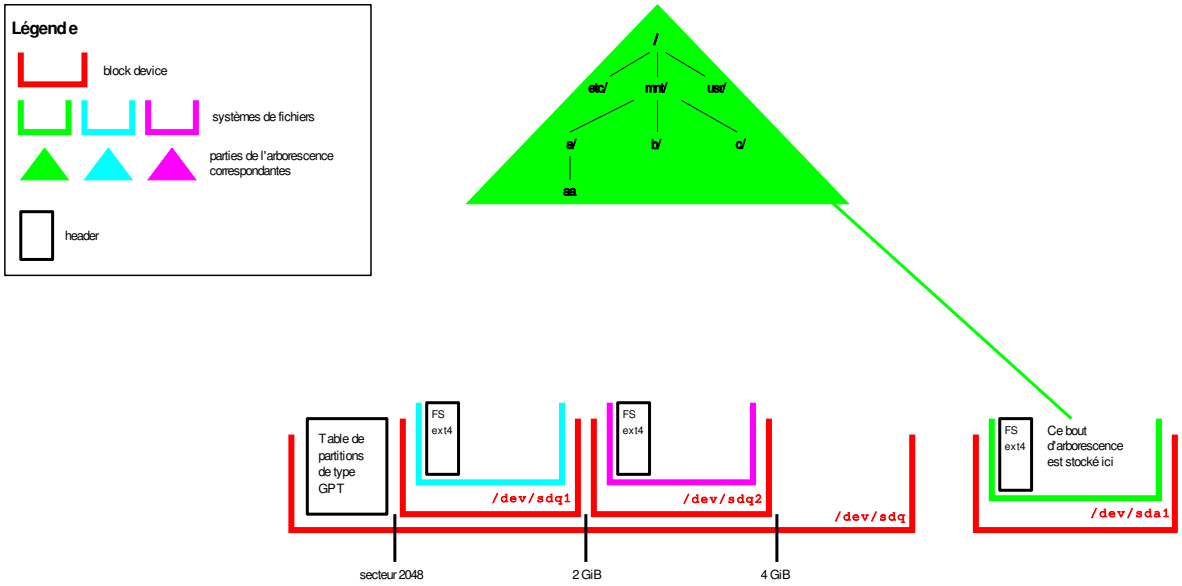
On crée une seconde partition (block device `/dev/sdq2`) sur la clef USB, elle s'étend de `2 GiB` à `4 GiB`.

```
# mkfs -t ext4 /dev/sdq1
```



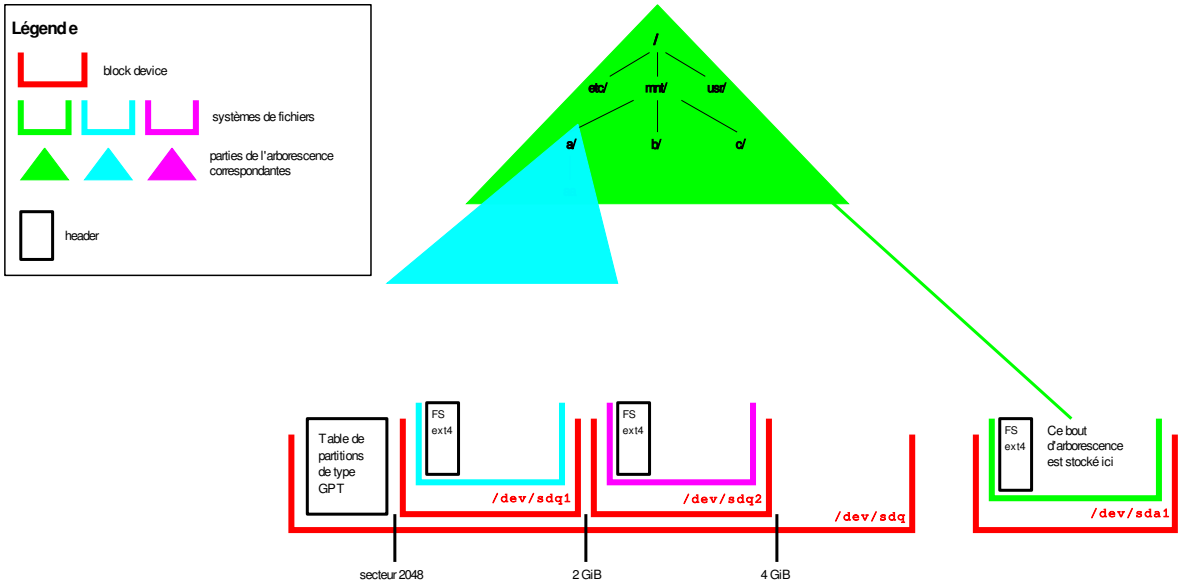
On installe un système de fichiers (représenté en couleur cyan) de type `ext4` dans le block device `/dev/sdq1`.

```
# mkfs -t ext4 /dev/sdq2
```



On installe un système de fichiers (représenté en couleur magenta) de type `ext4` dans le block device `/dev/sdq2`.

```
# mount /dev/sdq1 /mnt/a
```

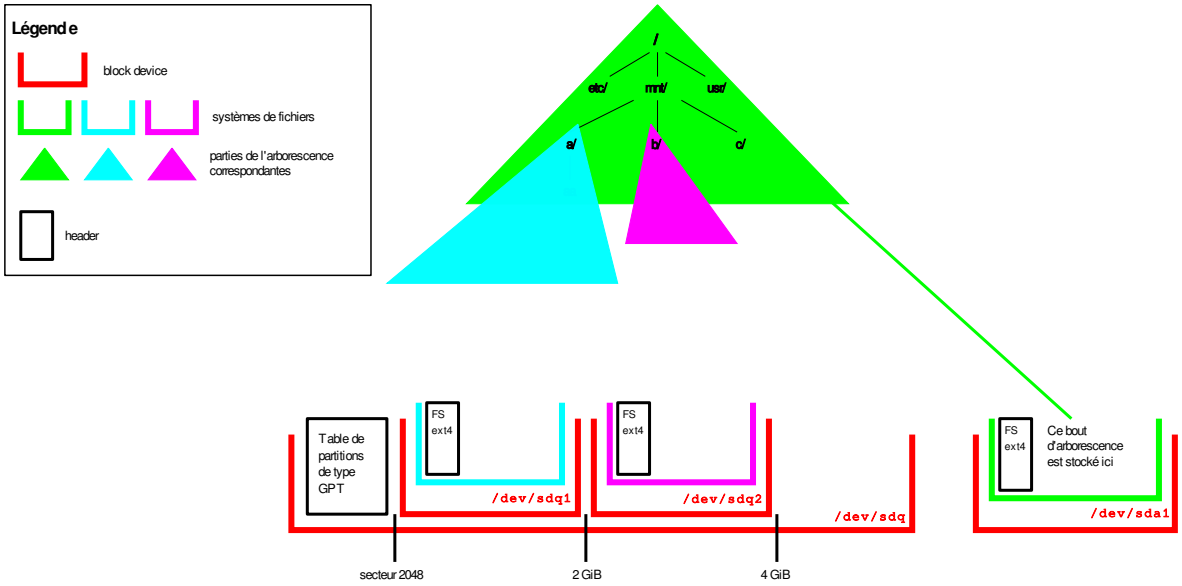


On monte le système de fichiers de couleur cyan installé dans `/dev/sdq1` sur le répertoire `/mnt/a`. Les informations relatives aux fichiers qui se trouvent dans la sous-arborescence dont l'apex est `/mnt/a/` seront stockées dans le système de fichiers installé dans `/dev/sdq1`.

En particulier, le fichier dont le chemin était `/mnt/a/aa` n'est plus accessible via ce chemin.

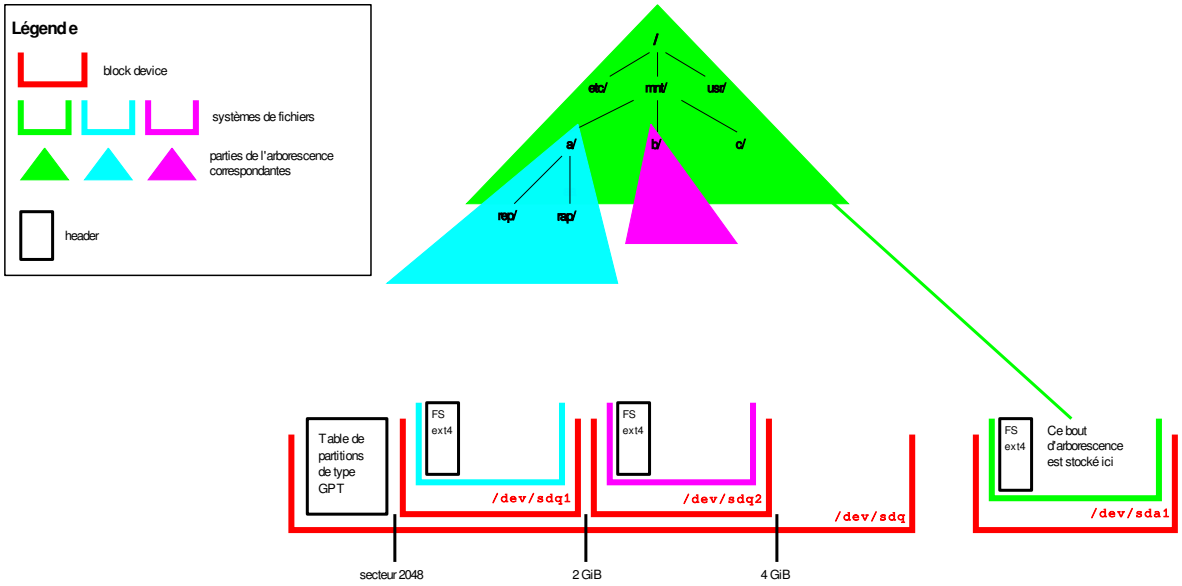
Bien sur, les informations concernant le fichier qui était accessible via le chemin `/mnt/a/aa` n'est pas effacé du système de fichiers installé sur `/dev/sda1`, il n'est simplement plus accessible par le chemin `/mnt/a/aa`. Il serait à nouveau accessible si on démontait le système de fichiers installé dans `/dev/sdq1` (avec la commande `umount /mnt/a/`).

```
# mount /dev/sdq2 /mnt/b
```



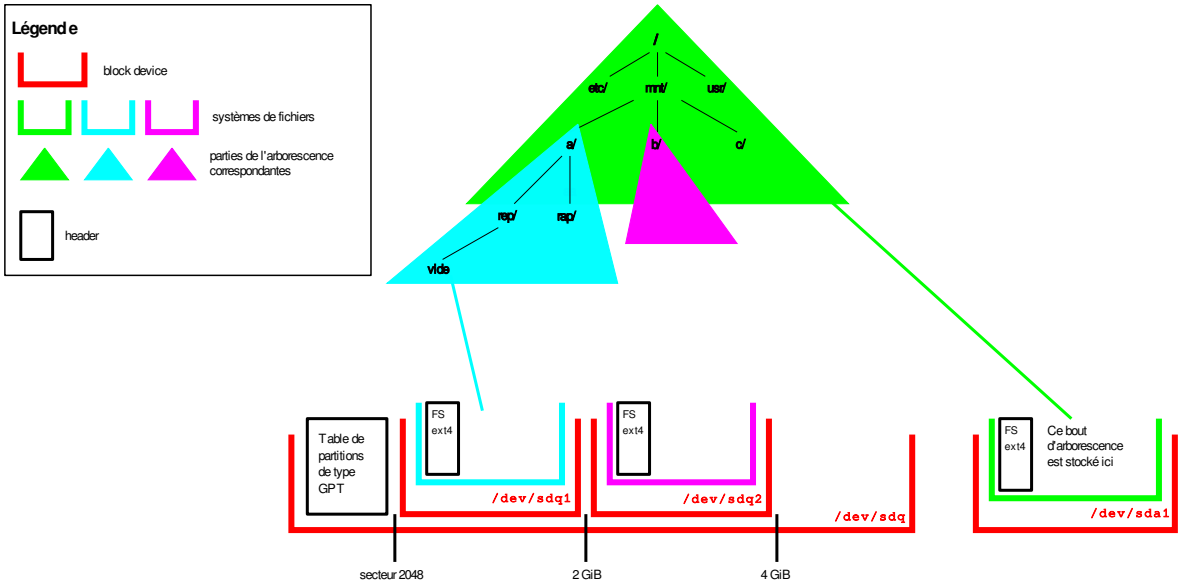
On monte le système de fichiers de couleur magenta installé dans `/dev/sdq2` sur le répertoire `/mnt/b`.

```
# mkdir /mnt/a/{rep,rap}
```



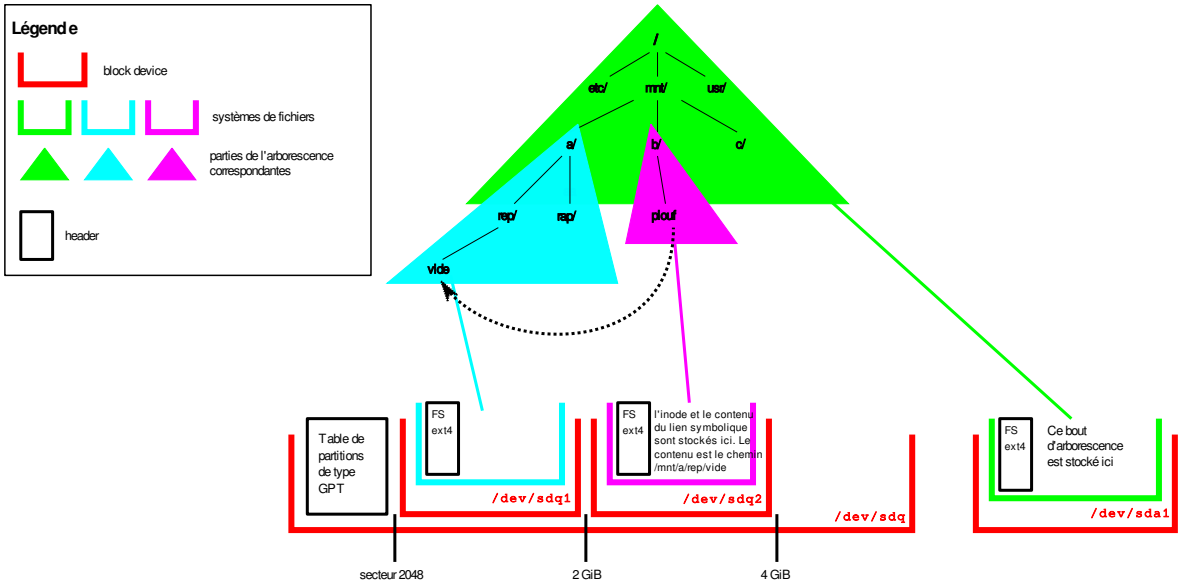
On crée deux répertoires `/mnt/a/rep` et `/mnt/a/rap`. Les informations relatives à ces répertoires (inode et contenu) sont stockées dans le système de fichiers de couleur cyan.

```
# touch /mnt/a/rep/vide
```



On crée un fichier vide nommé `vide` dans le répertoire `/mnt/a/rep/` (les informations de ce fichier se trouvent dans le système de fichier de couleur cyan).

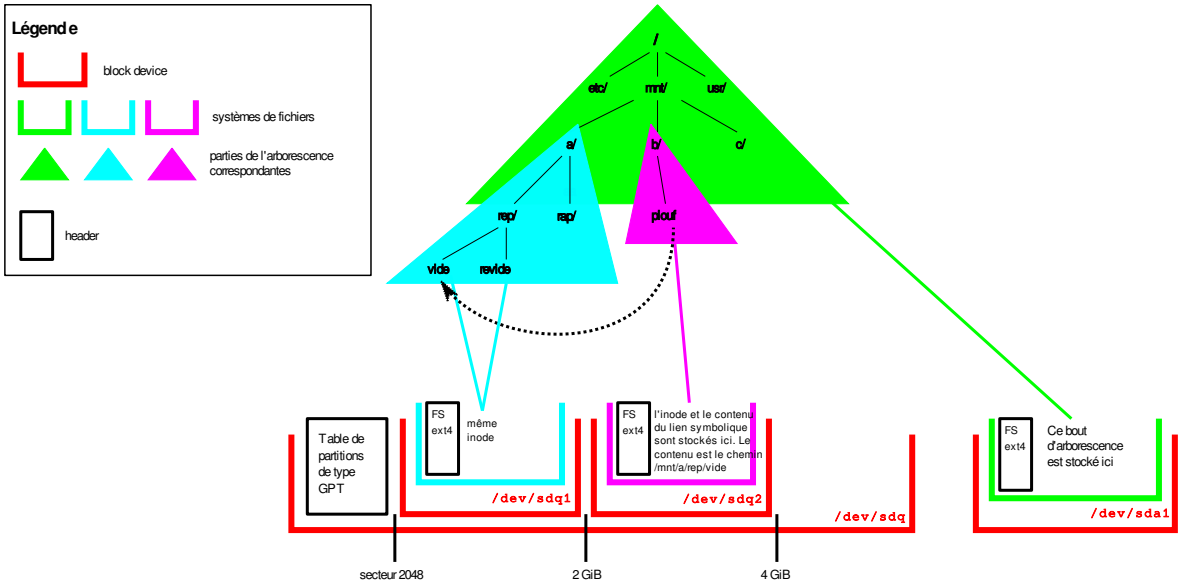
```
# ln -s /mnt/a/rep/vide /mnt/b/plouf
```



On crée un lien symbolique nommé `plouf` dans le répertoire `/mnt/b/`. Il pointe vers le chemin `/mnt/a/rep/vide`.

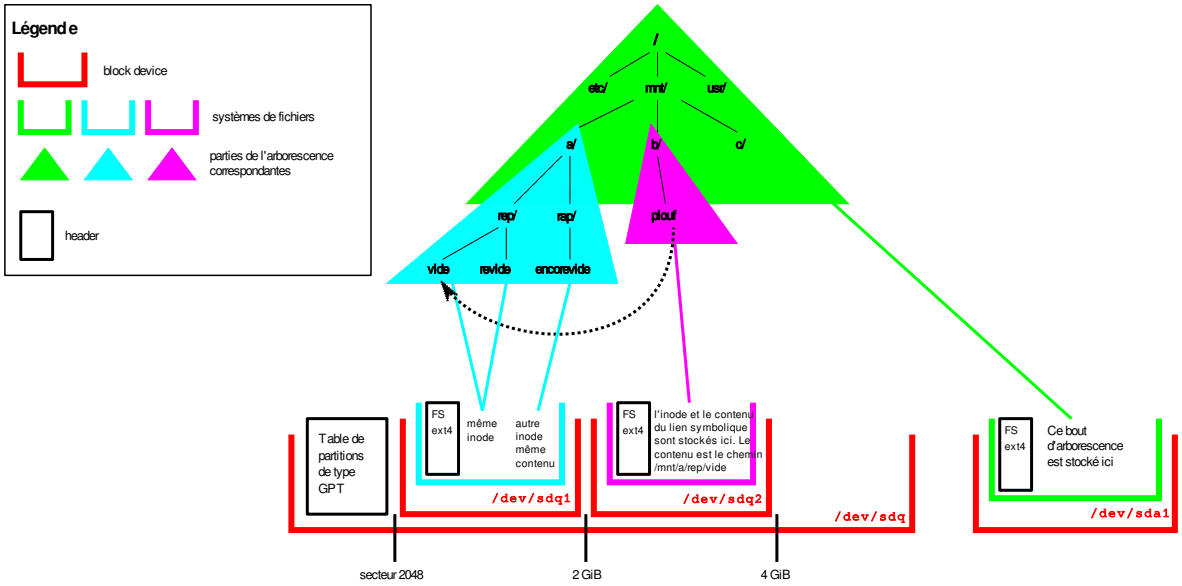
Les informations sur ce lien symbolique (inode et contenu) sont stockées dans le système de fichier de couleur magenta. Le contenu de ce lien symbolique est le chemin `/mnt/a/rep/vide`.

```
# cp -l /mnt/a/rep/vide /mnt/a/rep/revide
```



On crée un lien physique nommé `/mnt/a/rep/revide`. Les fichiers identifiés par `/mnt/a/rep/vide` et `/mnt/a/rep/revide` sont un seul et même fichier. Il s'agit seulement de deux noms différents donnés au même fichier. Ils ont le même numéro d'inode.

```
# cp /mnt/a/rep/vide /mnt/a/rap/encorevide
```



On copie le fichier `/mnt/a/rep/vide` vers `/mnt/a/rap/encorevide`. Ces deux fichiers sont différents. Le contenu de `/mnt/a/rep/vide` a été entièrement recopié.