Les fichiers

Problèmes imposés à un processus

- Enregistrer une grande quantité d'informations
- Persistance des informations en mémoire
- Accès simultané par d'autres processus

Types de fichiers

- Fichiers ASCII
  - Lignes de textes
  - Possibilité de les lire et de les modifier
- Fichiers Binaires

Noms des fichiers

- Abstration des données techniques qui permettent de référencer les données sur le disque.
- Unix
  - Extension indicative mais pas imposée
- MS-DOS
  - Extension définie par le système, par exemple
    - *fichier.c* est un fichier qui contient du code C
    - *fichier.jpg* est un fichier qui contient une image
    - *fichier.zip* est un fichier qui contient une archive compressée

Accès aux fichiers

- Accès séquentiel
  - Lecture du début vers la fin (ex : cassette, bande magnétique)
- Accès aléatoire
  - Accès direct à n'importe quel fichier

Aujourd'hui quasiment tous les systemes sont en accès aléatoire.

Attributs des fichiers

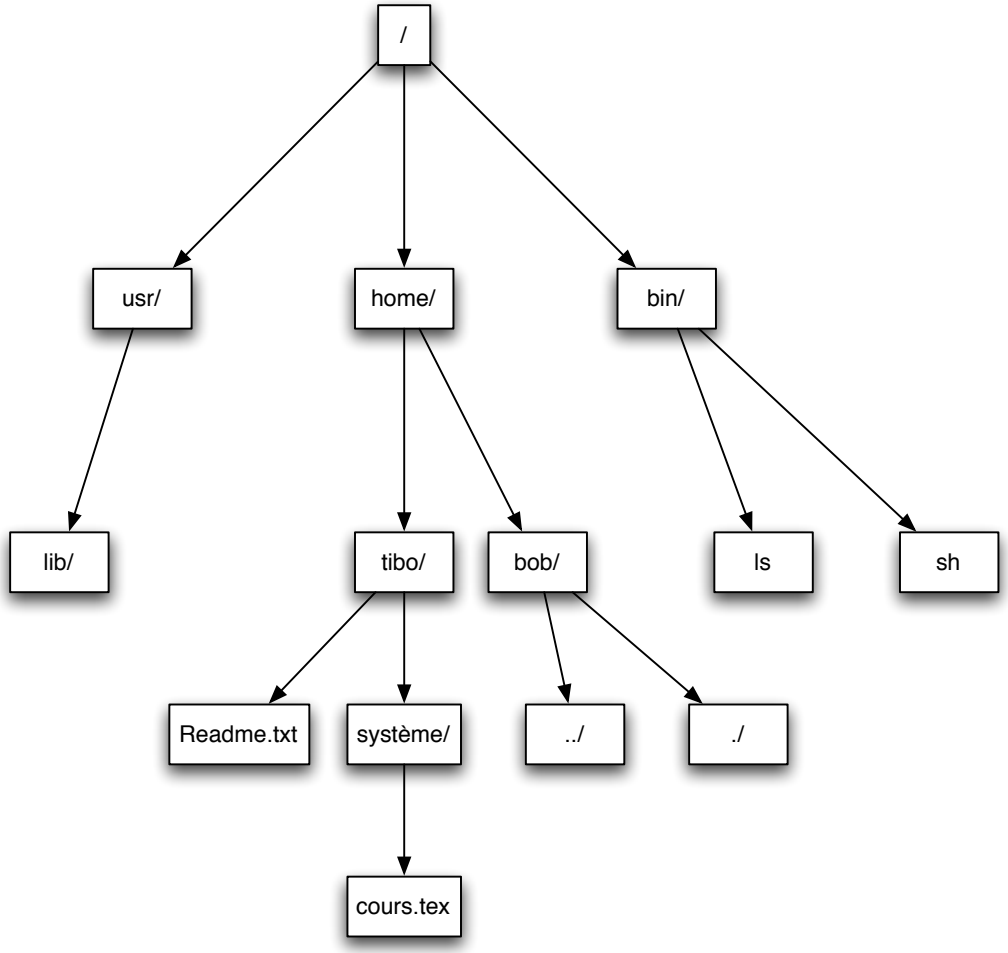
Informations complémentaires concernant un fichier (dépendemment du système)

- Protection
- Créateur
- Propriétaire
- Date de création
- Taille
- Date modification
- ...

Intérets des répertoires

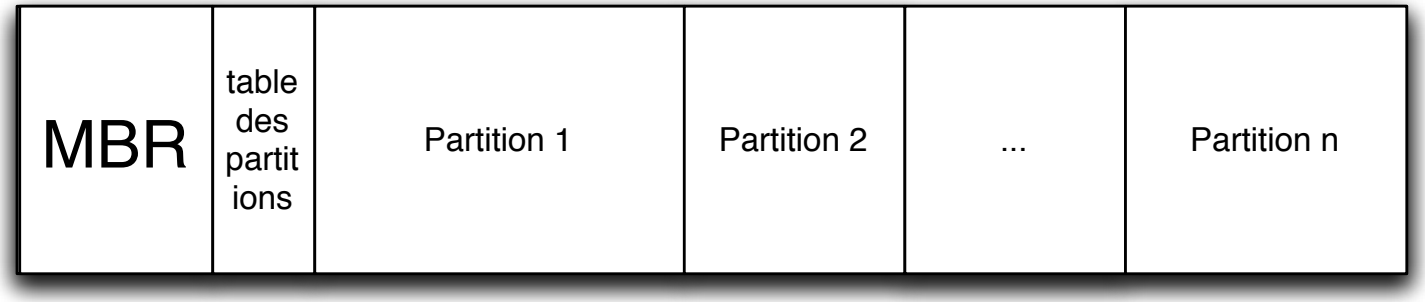
- Hiérarchiser
- Organiser

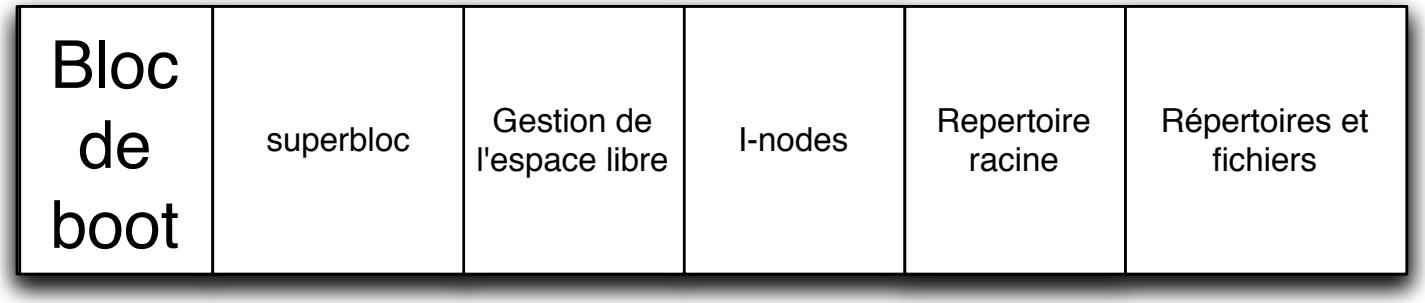
Arborescence



Chemins relatifs/absolus

Partitionnement et partition





Implantation des fichiers

- Contigüe
  - Perte d'espace lors de supression
  - Perte de temps à l'accès
- Aléatoire (solution utilisée)
- Bloc de taille fixée: quelle taille utilisée?

Mémorisationdes blocs libres

Comme pour la mémoire

- Liste chainée
- Tableaux de bits

Ouverture d'un fichier

$$int\ open(const\ char * path, int\ flags, mode\_t\ mode)$$

- *flags* :
  - O_RDONLY lecture seule
  - O_WRONLY écriture seule
  - O_RDWR lecture écriture
  - O_APPEND concaténation
- *mode* droits sur le fichier

Fermeture d'un fichier

$$int\ close(int\ d)$$

Lecture sur un fichier

$$ssize\_t\ read(int\ d,\ void\ *\ buf,\ size\_t\ nbytes)$$

- retourne le nombre d'octets effectivement lus
- *nbytes* nombre d'octets que la fonction tente de lire
- *buf* pointeur vers la zone mémoire tampon

Ecriture sur un fichier

$$ssize\_t\ write(int\ d,\ const\ void\ *\ buf,\ size\_t\ nbytes)$$

- retourne le nombre d'octets effectivement écrits
- *nbytes* nombre d'octets que la fonction tente d'écrire
- *buf* pointeur vers la zone mémoire contenant les informations à écrire

Se positionner dans un fichier

$$off\_t\ lseek(int\ fildes,\ off\_t\ offset,\ int\ whence)$$

- retourne le nombre d'octets effectivement écrits
- *offset* ampleur du déplacement
- *whence* point de départ du déplacement :
  - SEEK_SET Debut du fichier
  - SEEK_CUR Position courante
  - SEEK_END Fin du fichier

Supprimer un fichier

```
int unlink(const char *pathname);
```

Informations sur un fichier

```
int stat(const char *file_name, struct stat *buf);
```

struct stat
dev_t st_dev; /* Périphérique */
ino_t st_ino; /* Numéro i-node */
mode_t st_mode; /* Protection */
nlink_t st_nlink; /* Nb liens matériels */
uid_t st_uid; /* UID propriétaire */
gid_t st_gid; /* GID propriétaire */
dev_t st_rdev; /* Type périphérique */
off_t st_size; /* Taille totale en octets */
blksize_t st_blksize; /* Taille de bloc pour E/S */
blkcnt_t st_blocks; /* Nombre de blocs alloués */
time_t st_atime; /* Heure dernier accès */
time_t st_mtime; /* Heure dernière modification */
time_t st_ctime; /* Heure dernier changement */

Manipulation des répertoires

- `DIR *opendir(const char *pathname);`
- `int closedir(DIR *dirp);`
- `struct dirent *readdir(DIR *dirp);`
- `void rewinddir(DIR *dirp);`

```
struct dirent {
/* etc */ /* Les autres membres sont spécifiques à
l'implémentation */
char d_name [256] ; /* Longueur maxi du nom de 256
octets pour POSIX */
}
```

Tests sur les fichiers

- `S_ISREG()`
- `S_ISDIR()`
- `S_ISLINK()`
- `S_ISCHR(m)`
- `S_ISBLK()`
- `S_ISFIFO()`
- `S_ISLNK()`
- `S_ISSOCK()`

Opérations sur les répertoires

- `char * getcwd(char *buf, size_t size);`
- `int chdir(const char *pathname);`
- `int rmdir(const char *pathname);`