Module SY5 – Systèmes d'Exploitation

Dominique Poulalhon
dominique.poulalhon@irif.fr

Université Paris Cité
L3 Informatique & DL Bio-Info, Jap-Info, Math-Info
Année universitaire 2023-2024

Organisation du système de fichiers

Où et comment stocker les fichiers ?

un fichier, c'est...

Où et comment stocker les fichiers ?

un fichier, c'est...

- du contenu

Où et comment stocker les fichiers ?

un fichier, c'est...

- du contenu
- des attributs ou méta-données (type, permissions, dates...)

Où et comment stocker les fichiers ?

un fichier, c'est...

- du contenu
- des attributs ou méta-données (type, permissions, dates...)

plusieurs solutions :

Où et comment stocker les fichiers ?

un fichier, c'est...

- du contenu
- des attributs ou méta-données (type, permissions, dates...)

plusieurs solutions :

- stockage contigu (CD-ROM)

Où et comment stocker les fichiers ?

un fichier, c'est...

- du contenu
- des attributs ou méta-données (type, permissions, dates...)

plusieurs solutions :

- stockage contigu (CD-ROM)
- liste chaînée de blocs

Où et comment stocker les fichiers ?

un fichier, c'est...

- du contenu
- des attributs ou méta-données (type, permissions, dates...)

plusieurs solutions :

- stockage contigu (CD-ROM)
- liste chaînée de blocs
- liste chaînée de numéros de blocs (*File Allocation Table*)

Où et comment stocker les fichiers ?

un fichier, c'est...

- du contenu
- des attributs ou méta-données (type, permissions, dates...)

plusieurs solutions :

- stockage contigu (CD-ROM)
- liste chaînée de blocs
- liste chaînée de numéros de blocs (*File Allocation Table*)
- table d'i-nœuds, regroupant les attributs et les adresses des blocs (mais pas les noms des fichiers)

Où et comment stocker les fichiers ?

un fichier, c'est...

- du contenu
- des attributs ou méta-données (type, permissions, dates...)

plusieurs solutions :

- stockage contigu (CD-ROM)
- liste chaînée de blocs
- liste chaînée de numéros de blocs (*File Allocation Table*)
- table d'i-nœuds, regroupant les attributs et les adresses des blocs (mais pas les noms des fichiers)
- *Master File Table* de NTFS : suite d'enregistrements décrivant les attributs ou donnant l'adresse des blocs où ils se trouvent (*y compris le nom et les données*)

Où et comment stocker les fichiers ?

un fichier, c'est...

- du contenu
- des attributs ou méta-données (type, permissions, dates...)

plusieurs solutions :

- stockage contigu (CD-ROM)
- liste chaînée de blocs
- liste chaînée de numéros de blocs (*File Allocation Table*)
- table d'i-nœuds, regroupant les attributs et les adresses des blocs (mais pas les noms des fichiers)
- *Master File Table* de NTFS : suite d'enregistrements décrivant les attributs ou donnant l'adresse des blocs où ils se trouvent (*y compris le nom et les données*)
- ...

Consultation des i-nœuds

```
int stat(const char *pathname, struct stat *statbuf);
int lstat(const char *pathname, struct stat *statbuf);
int fstat(int fd, struct stat *statbuf);
```

remplissent une `struct stat` avec les caractéristiques de l'i-nœud et retournent 0, ou -1 en cas d'erreur, précisée par `errno` (`ENOENT` ou `EACCESS` par exemple)

le type `struct stat` contient (entre autres) les champs suivants :

```
struct stat {
  dev_t     st_dev;       /* ID of device containing file */
  ino_t     st_ino;       /* Inode number */
  mode_t    st_mode;      /* File type and mode */
  uid_t     st_uid;       /* User ID of owner */
  off_t     st_size;      /* Total size, in bytes */
  blksize_t st_blksize;   /* Block size for filesystem I/O */
  blkcnt_t st_blocks;     /* Number of 512B blocks allocated */
  /* ... */
```

Consultation des i-nœuds

Par exemple, pour déterminer un numéro d'i-nœud :

```
struct stat st;  /* déclaration préalable d'une struct stat */
if (stat("toto", &st)==-1) perror("stat␣toto");
else printf("inoeud␣numéro␣:␣%ld\n", st.st_ino);
```

Consultation des i-nœuds

Par exemple, pour déterminer un numéro d'i-nœud :

```
struct stat st;  /* déclaration préalable d'une struct stat */
if (stat("toto", &st)==-1) perror("stat␣toto");
else printf("inoeud␣numéro␣:␣%ld\n", st.st_ino);
```

Mais d'autres champs sont plus compliqués à manipuler :

- `st.st_atime`, `st.st_ctime`, `st.st_mtime` sont des `struct timespec`

Consultation des i-nœuds

Par exemple, pour déterminer un numéro d'i-nœud :

```
struct stat st;  /* déclaration préalable d'une struct stat */
if (stat("toto", &st)==-1) perror("stat␣toto");
else printf("inoeud␣numéro␣:␣%ld\n", st.st_ino);
```

Mais d'autres champs sont plus compliqués à manipuler :

- `st.st_atime`, `st.st_ctime`, `st.st_mtime` sont des `struct timespec`
- `st.st_uid` et `st.st_gid` sont les identifiants de l'utilisateur et du groupe propriétaires ; pour déterminer leurs noms, il faut se référer au fichier des mots de passe, par exemple à l'aide de :
  `struct passwd *getpwuid(uid_t uid);`

Consultation des i-nœuds

Par exemple, pour déterminer un numéro d'i-nœud :

```
struct stat st;  /* déclaration préalable d'une struct stat */
if (stat("toto", &st)==-1) perror("stat␣toto");
else printf("inoeud␣numéro␣:␣%ld\n", st.st_ino);
```

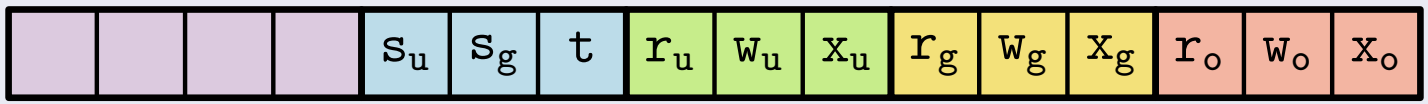
Mais d'autres champs sont plus compliqués à manipuler :

- `st.st_atime`, `st.st_ctime`, `st.st_mtime` sont des `struct timespec`
- `st.st_uid` et `st.st_gid` sont les identifiants de l'utilisateur et du groupe propriétaires ; pour déterminer leurs noms, il faut se référer au fichier des mots de passe, par exemple à l'aide de :
  `struct passwd *getpwuid(uid_t uid);`
- `st.st_mode` est un entier qui agrège deux informations : le type et les droits du fichier

Interprétation du champ st_mode

le champ st_mode est constitué de 2 octets, soit 16 bits :

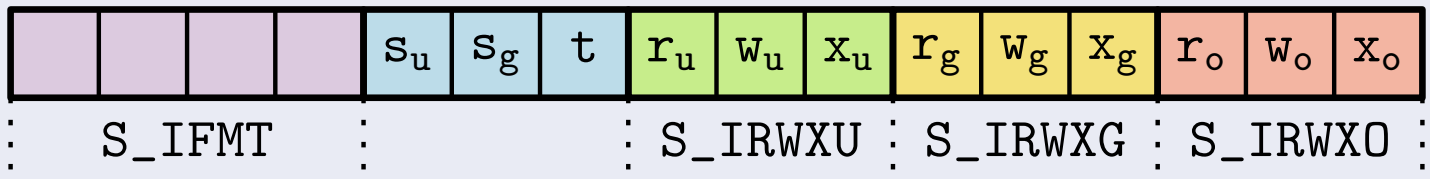
4 pour le type, puis 4 fois 3 pour les droits :



Interprétation du champ st_mode

le champ st_mode est constitué de 2 octets, soit 16 bits :

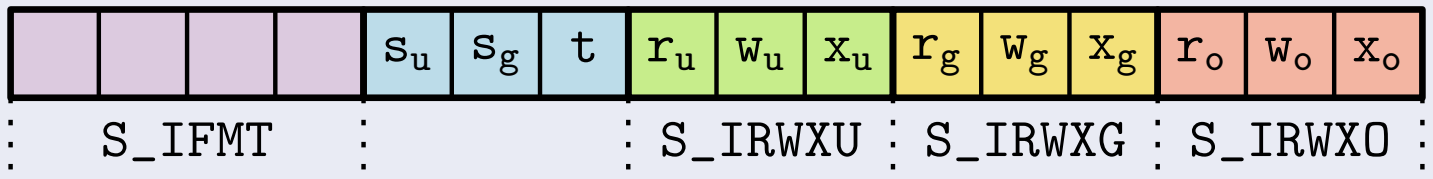
4 pour le type, puis 4 fois 3 pour les droits :



l'interprétation passe par des combinaisons logiques bit à bit avec des masques, comme par exemple S_IFMT=0170000, c'est-à-dire 4 bits 1 suivis de 12 bits 0, ou S_IWOTH=02, c'est-à-dire un unique 1 en avant-dernière position

Interprétation du champ st_mode

le champ st_mode est constitué de 2 octets, soit 16 bits :
4 pour le type, puis 4 fois 3 pour les droits :



l'interprétation passe par des combinaisons logiques bit à bit avec des masques, comme par exemple S_IFMT=0170000, c'est-à-dire 4 bits 1 suivis de 12 bits 0, ou S_IWOTH=02, c'est-à-dire un unique 1 en avant-dernière position

Exemple, pour tester si un fichier est un répertoire :

```
struct stat st;
if (stat("toto", &st)==-1) perror("stat␣toto");
if ((st.st_mode & S_IFMT) == S_IFDIR) { /* ... */ }
```

ou de manière équivalente :

```
if (S_ISDIR(st.st_mode)) { /* ... */ }
```