Systèmes d'exploitation, 2ème année

Système de gestion de fichiers - Symbolique programmeur

Yves Stadler

Université Paul Verlaine - Metz

1er décembre 2011

Agenda

Plan du cours

- Pourquoi les fichiers
- Un système de gestion
- Représentation symbolique externe
- Représentation symbolique noyau

Objectifs et motivations

Motivation

- La mémoire vive disparait après usage
- Mémoire vive non persistante (données à durée de vie = vie du processus)
- Faible capacité
- Non partageable en dehors de l'ordinateur
- Aucune organisation personnelle

Objectifs

- Mémoire persistante
- Très grande capacités
- Partage possible
- Organisation des données

Le Système de Gestion des Fichiers

À quoi sert le SGF

- Définition d'un fichier
- Manipulation d'un fichier
- Gérer l'organisation logique
- Lien entre le symbolique et le physique
- Gestion méta-logique (sécurité, robustesse, ...)

Représentation symbolique externe

Plan du cours

- Flux linéaire d'octets
- Le programmeur n'a besoin que d'un nombre minimun d'information
  - nom
  - taille
  - type
  - propriétaire
  - dates
  - protections

Flux linéaire d'octets

- Tubes, Sockets
- Tout ce qui peut entrer dans `open(2)`

Représentation symbolique externe

Contenus

- Sources, traitement de texte, ...
- Scripts, exécutables, ...
- Spéciaux : fichier null, périphérique
- Répertoires

Formats

- Fichiers ordinaires non-exécutables / exécutable
- Fichiers spéciaux
- Répertoires

Attention

- L’extension est juste un visuel et n’a rien a voir avec le format !

Représentation symbolique externe

Champs de protection : ls -l

U	G	O
r x w	r x w	r x w

- - rwx rwx rwx

Légende

- User Group Others
- Read Write eXecute
- Absence de droit : -

Représentation symbolique externe

Distinction fichier répertoire

- `-rwxrwxrwx` : fichier
- `drwxrwxrwx` : répertoire

Distinction du format de fichier

```
ystadler@stage-habbas:~$ file lock
lock: ASCII text
ystadler@stage-habbas:~$ file vimconf.tar.gz
vimconf.tar.gz: gzip compressed data, from Unix, last
modified: Fri Oct 21 13:49:32 2011
ystadler@stage-habbas:~$
```

Représentation symbolique externe

Les permissions en C

- `S_IRUSR S_IWUSR S_IXUSR`
- `S_IRGRP S_IWGRP S_IXGRP`
- `S_IROTH S_IWOTH S_IXOTH`

Consultation et modification des droits

- Appel système `access(2)` permet de lire ces informations
- Appel système `chmod(2)` permet de modifier ces informations
- Il faut que le progamme ait les droits de le faire.

Représentation symolique interne

Associer le nom de fichier à un élément système

- Un fichier c’est un emplacement de début sur le disque et une taille
- Comment dire que "readme.txt" est à cet endroit

Philosophie

- On a pas besoin d’avoir en permanence un lien vers tous les fichiers
- Notion de fichiers ouverts et fermés (`open(2)` `close(2)`)
- On doit conserver un lien avec les fichiers ouverts (efficacité)
- Pourquoi ne pas numéroter les fichiers ouverts ?
- On utilise une table des fichiers ouverts

Représentation symolique interne

Table des fichiers ouverts - kernel

- C’est un cache, recherche physique effectuée une seule fois.
- Chaque fichiers dispose d’un numéro d’entrée dans la table de fichiers ouverts
- Ce numéro est le descripteur de ce fichiers
- La table gère ensuite les permissions, liens physique, ...
- On stocke dans cette table la position de lecture dans le fichier

Représentation symolique interne

On se souvient

- `STDIN_FILENO` vaut toujours 0 pour tous les processus ?

Table des fichiers - processus

- Table à deux niveaux
- Chaque processus gère ses descripteurs de fichiers (dont `STDIN_FILENO` ...
- Chaque élément de la table des fichiers ouverts pointent sur un élément de la table des fichiers ouverts du système
- On ne peut pas partager de descripteur de fichier entre processus (sauf avec fork car duplication).

Représentation symolique interne

Protections

- opération de fichiers :
  - open(2)
  - close(2)
  - lseek(2)
  - read(2)
  - write(2)

Standard library

Que sont les fonctions usuelles ?

- fichier = FILE* : "wrapper" autour d'un descripteur de fichier
- structure sous jacente _IO_FILE (dépent des systèmes)
- fopen(3), fclose(3), fread(3), fscanf(3) : ajout d'un buffer

Standard library

```
typedef struct  {
  int           level;      /* fill/empty level of buffer */
  unsigned      flags;      /* File status flags          */
  char          fd;         /* File descriptor            */
  unsigned char hold;       /* Ungetc char if no buffer   */
  int           bsize;      /* Buffer size                */
  unsigned char *buffer;    /* Data transfer buffer       */
  unsigned char *curp;      /* Current active pointer     */
  unsigned      istemp;     /* Temporary file indicator   */
  short         token;      /* Used for validity checking */
 } FILE;
```

Standard library

Différences

- open(2), close(2), lseek(2), read(2), write(2)
- fopen(3), fclose(3), fread(3), fscanf(3)

man man

```
1   Executable programs or shell commands
2   System calls (functions provided by the kernel)
3   Library calls (functions within program libraries)
4   Special files (usually found in /dev)
5   File formats and conventions eg /etc/passwd
6   Games
7   Miscellaneous (including macro  packages  and  conven-
    tions), e.g. man(7), groff(7)
8   System administration commands (usually only for root) ..
```