Système de Fichiers

Complément sur les structures et méta-données

Olivier Dalle
Université de Nice - Sophia Antipolis
http://deptinfo.unice.fr/
\

Stockage brut des données

- **Périphériques physique = réservoir de blocs**
  - Collection de blocs de taille fixe
  - Vue indépendante du type de support
    - Disque dur, CD/DVD
  - Aucune gestion interne
- **Adressage pas ou peu hiérarchique (géométrie)**
  - CHS (Cylinder / Head / Sector)
  - LBA (Logical Bloc Addressing)
  - Unix fournit l'illusion d'un unique fichier / periphérique
    - Fichier special de type bloc (exemple: /dev/sda)
- **Problèmes à résoudre par le système**
  - Permissions? Propriétaire?
  - Espace libre/occupé?

Stockage structuré

- **Entièrement construit à partir du stockage brut**
  - Les données de gestion (meta-data) sont stockées avec les données des utilisateurs (data)
  - Fournit l'illusion d'une organisation arborescente
    - Fichiers, répertoires
    - Taille variable
  - Méta-données
    - propriétaire, horodatage, permissions...

Stockage structuré (suite)

- **Organisé selon des règles très précises**
  - De complexité parfois très élevée
  - Plusieurs modèles d'organisation (types de SF)
    - Ext2,3,4, Reiserfs, XFS, AFS, ...
  - Chaque modèle possède des avantages et inconvénients
    - Rapidité, flexibilité, compacité, robustesse, extensions...
    - Un vaste sujet de recherche!
  - Choix du modèle lors du formatage de la partition
    - Création de la structure "vide"... parfois très longue
      - Initialisation de listes chainées/tables de blocs vides, compteurs, copies de sauvegarde, ...
      - Recherche de blocs défectueux (optionnel)
    - Un bloc très spécial (et précieux): le superbloc
      - Informations de gestion globales
      - Point de passage obligatoire = centre nerveux du SGF

Système de Fichier Unix

- Fournit une vue indépendante du type de système de fichiers
- Principale abstractions
  - Fichiers
    - ordinaires, répertoires, liens symboliques, fifos, spéciaux de périphérique, ...
    - Interface standard: open, read, write, lseek, ...
  - Répertoires
    - Fichier special *catalogue*
    - *Interface spécifique*
  - Chemin
    - Suite de répertoires à traverser pour atteindre un fichier
    - Plusieurs possibles pour chaque fichier (liens), minimum = 0 (!)
  - I-node (Information node)
    - Contient toutes les meta-données d'un fichier
      - Taille, proprietaire, horodatage, permissions, type, adresse des bocs de données
    - Un seul par fichier

Récupération des meta-données d'un fichier

- **fonctions stat/lstat/fstat**
  - stat: information a propos d'un fichier designé par l'un de ses chemins
  - lstat: idem sauf si lien symbolique (info sur le lien au lieu de la cible)
  - fstat: information a propos d'un fichier ouvert (désigné par son numéro de fd)
- **Fonctionnement**
  - En C: retournent une struct stat
    - man stat ou cat /usr/include/sys/stat.h
  - En python: os.stat(path, ...) retourne un objet dont les attributs correspondent (plus ou moins) aux champs de la structure stat :
    - mode (permission), inode, device, nb liens, uid, gid, taille, dates
    - Infos additionnelles selon l'OS

Récupération du contenu d'un répertoire

- **En C, plusieurs méthodes**
  - Standard: grâce à la libc (section 3 du man)
    - opendir, readdir, closedir
    - Itératif: chaque readdir retourne l'entrée suivante dans le reepertoire
  - Bas niveau, selon système
    - Exemple MACOS: getdirentries
      - utilise un descripteur de fichier ouvert (car repertoire = fichier)
- **En python:**
  - os.listdir(chemin)
    - retourne la liste des fichiers
    - Ouverture inutile