Gestion des fichiers

Rappels

Abdelkader Gouaïch[1]

[1]Département d'Informatique
IUT de Montpellier
Université de Montpellier

Cours M3101 Systèmes d'exploitation, 2014

Sommaire

La structure des fichiers Linux

Le concept de fichier

- Le concept de fichier dans le système Linux est un concept fondamental.
- Les fichiers offrent une interface simple et cohérente pour :
  - les services du système d'exploitation
  - les ressources matérielles.

Les fichiers dans Linux

Linux et les fichiers

- Linux considère presque tout comme un fichier :
  - Cela veut dire que les programmes utilisent des disques durs, les ports série, les imprimantes, et les autres ressources matérielles comme des fichiers.
  - Il y'a évidemment quelques exceptions à cela.

Les fichiers dans Linux

Linux et les fichiers

- Linux considère presque tout comme un fichier :
  - Cela veut dire également qu'un programmeur va utiliser seulement cinq fonctions élémentaires : **open**, **close**, **read**, **write**, et **ioctl**

Les fichiers dans Linux

Linux et les fichiers

- Linux considère presque tout comme un fichier :
  - Les répertoires sont aussi considérés comme des fichiers spéciaux.
  - On y accède en utilisant les fonctions : **opendir** et **readdir** pour respectivement ouvrir et lire le contenu d'un répertoire sans connaître les détails d'implémentation.

Concepts fondamentaux d'un système de fichiers

- Chaque système de fichier Linux implémente un ensemble de concepts fondamentaux communs à tous les systèmes Unix-like :
  - Les fichiers sont représentés par des inodes
  - Les répertoire sont simplement des fichiers contenant une liste d'entrées
  - L'interaction avec les équipements (device) se fait par l'intermédiaire de requêtes E/S (I/O) sur des fichiers spéciaux

Lignes directrices

Concepts fondamentaux d'un système de fichiers

Inode (Index Node)

- Chaque fichier est représenté par une structure : inode (index node)
- Un inode contient la description complète d'un fichier :
  - Type de fichier
  - Les droits d'accès
  - Le propriétaire
  - Compteur de liens
  - Timestamps
  - Taille
  - Tableau de pointeurs vers les blocs de données

Structure de l'inode

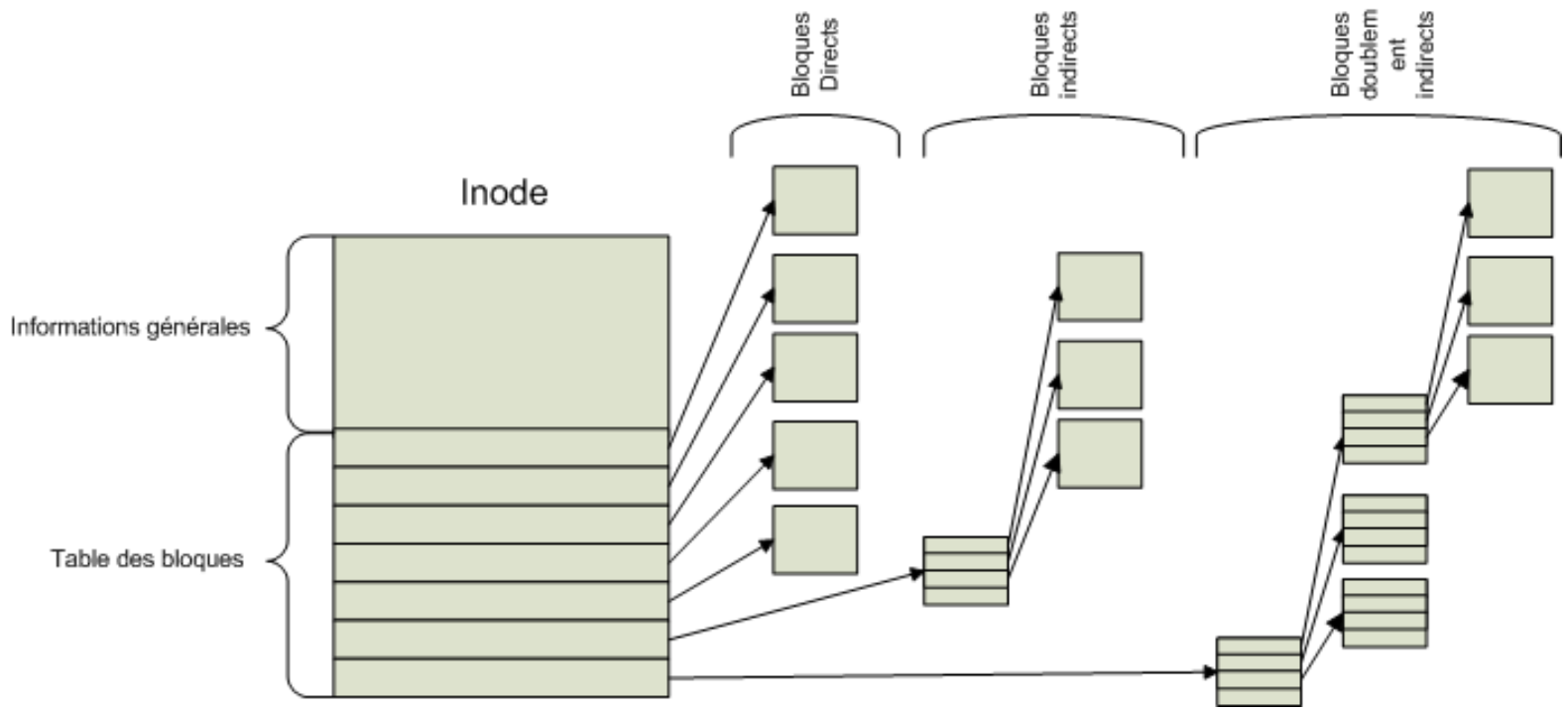


Figure : La figure illustre la structure de l'inode et des bloques

Inodes

Blocs de données

- Les adresses des blocs de données alloués pour le fichier sont stockés dans la structure du inode.
- Quand un utilisateur effectue une requête I/O sur un fichier, le noyau (OS) :
  - convertit cette valeur du pallier (offset) en un numéro de bloc
  - ce numéro est utilisé comme index dans la table des adresses de blocs
  - lit ou écrit sur les blocs physiques

Lignes directrices

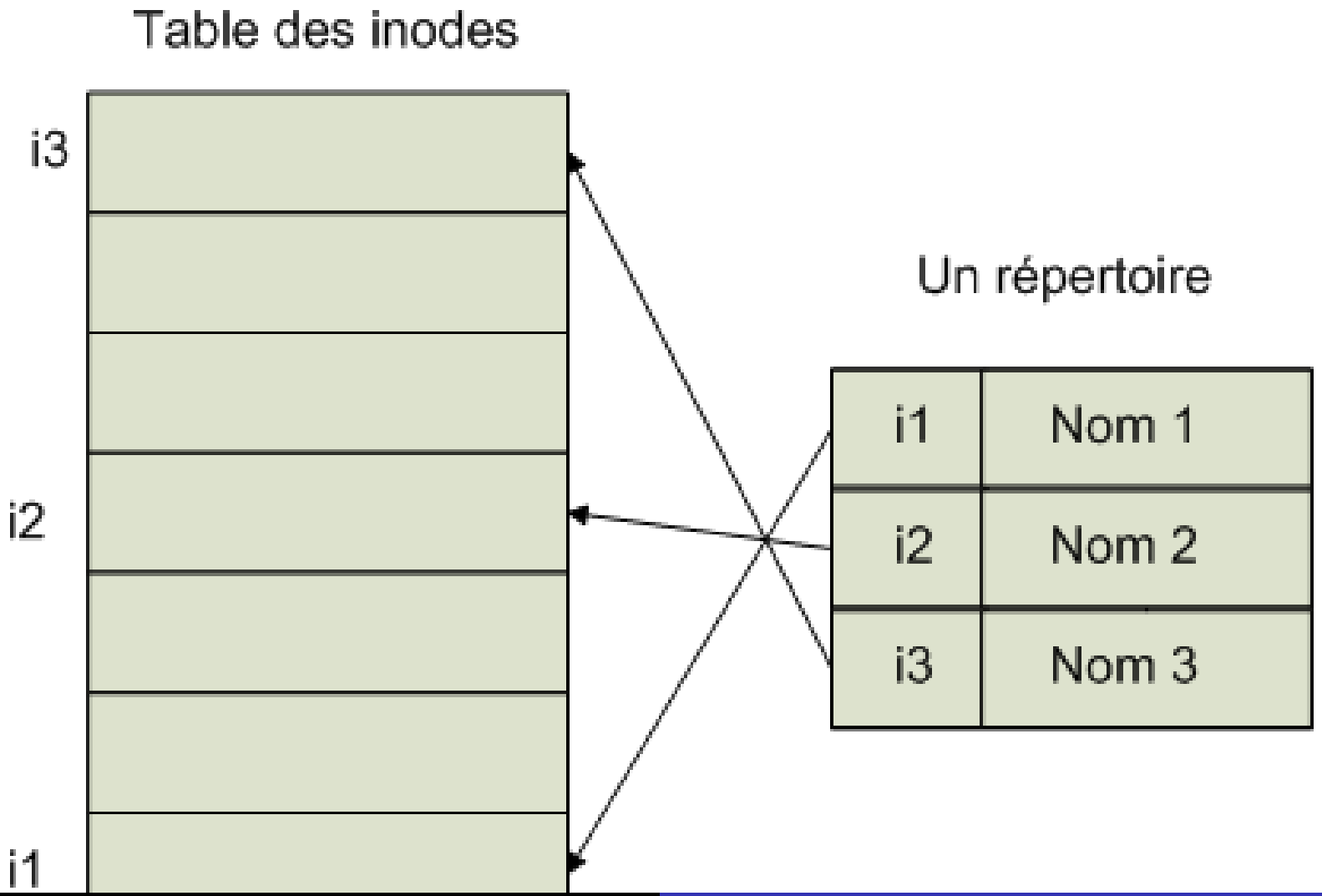
Répertoire

- Les répertoires sont organisés dans une hiérarchie arborescente
- Un répertoire contient des fichiers et des sous-répertoires

Répertoire

- Un répertoire est réalisé comme un fichier spécial :
  - Ce fichier contient une liste d'entrées
  - Chaque entrée est composée d'un numéro d'inode et d'un nom de fichier
  - Le noyau du système prend en charge la conversion d'un chemin de répertoire vers le numéro d'inode qui correspond au fichier spécial représentant le répertoire

Exemple de répertoire



Lignes directrices

Le concept de lien

- Dans le système de fichier de Linux on trouve le concept de lien
- Ajouter un lien revient à :
  - Créer une entrée dans le répertoire
  - Dans cette entrée, le numéro d'inode correspond à l'inode du ficher cible
  - Il faut également incrémenter le compteur de liens dans l'inode cible
- Pour effacer un lien (avec la commande rm sur un lien) :
  - Le noyau décrémente le compteur de liens
  - Si le compteur devient zéro, l'inode est libéré
- Ce type de lien est appelé un hard link

Le concept de lien

- Dans le système de fichier de Linux on trouve le concept de lien
- Ajouter un lien revient à :
  - Créer une entrée dans le répertoire
  - Dans cette entrée, le numéro d'inode correspond à l'inode du ficher cible
  - Il faut également incrémenter le compteur de liens dans l'inode cible
- Pour effacer un lien (avec la commande rm sur un lien) :
  - Le noyau décrémente le compteur de liens
  - Si le compteur devient zéro, l'inode est libéré
- Ce type de lien est appelé un hard link

Le concept de lien

- Dans le système de fichier de Linux on trouve le concept de lien
- Ajouter un lien revient à :
  - Créer une entrée dans le répertoire
  - Dans cette entrée, le numéro d'inode correspond à l'inode du ficher cible
  - Il faut également incrémenter le compteur de liens dans l'inode cible
- Pour effacer un lien (avec la commande rm sur un lien) :
  - Le noyau décrémente le compteur de liens
  - Si le compteur devient zéro, l'inode est libéré
- Ce type de lien est appelé un hard link

Le concept de lien

- Dans le système de fichier de Linux on trouve le concept de lien
- Ajouter un lien revient à :
  - Créer une entrée dans le répertoire
  - Dans cette entrée, le numéro d'inode correspond à l'inode du ficher cible
  - Il faut également incrémenter le compteur de liens dans l'inode cible
- Pour effacer un lien (avec la commande rm sur un lien) :
  - Le noyau décrémente le compteur de liens
  - Si le compteur devient zéro, l'inode est libéré
- Ce type de lien est appelé un hard link

Restriction sur les liens durs

- Les liens durs ne peuvent exister que dans un même système de fichier
- Les liens durs ne concernent que les fichiers
- On ne peut pas créer un lien dur qui pointe vers un répertoire pour éviter l'apparition de cycles dans l'arborescence des répertoires

Lien symbolique

- Un lien symbolique est simplement un fichier qui a comme contenu un nom de fichier.
- Durant la phase de conversion d'un chemin vers un numéro d'inode, quand le système rencontre un lien symbolique :
  - Il lit le contenu du fichier
  - Remplace le nom du lien avec ce contenu
  - Continue l'interprétation et la résolution du chemin
- Il est possible de créer des liens symboliques entre des fichiers entre des systèmes de fichiers différents.

Lien symbolique

- Un lien symbolique est simplement un fichier qui a comme contenu un nom de fichier.
- Durant la phase de conversion d'un chemin vers un numéro d'inode, quand le système rencontre un lien symbolique :
  - Il lit le contenu du fichier
  - Remplace le nom du lien avec ce contenu
  - Continue l'interprétation et la résolution du chemin
- Il est possible de créer des liens symboliques entre des fichiers entre des systèmes de fichiers différents.

Lignes directrices

Les équipements (device)

- Dans les systèmes Unix-like les devices sont utilisés à travers des fichiers spéciaux
- Un fichier spécial de device n'utilise pas de place mémoire sur le disque, il représente seulement un point d'accès vers le driver du device
- Deux types de fichiers existent : caractère et bloque

Les types de device

Caractère

Autorise les opérations IO dans le mode caractère

Bloque

Les données sont écrites dans une mémoire tampon par l'intermédiaire de fonction tampon

Identification de fichier device

- Un fichier spécial est référencé avec un couple :
  - numéro majeur identifiant le type de device
  - un numéro mineur identifiant l'unité.

Lignes directrices

Virtual File System (VFS)

- Le noyau Linux contient une couche d'abstraction qui offre une vue unifiée d'un système de fichiers virtuel.
- Cette vue sera utilisée par les programmes qui veulent interagir avec les fichiers.

Virtual File System (VFS)

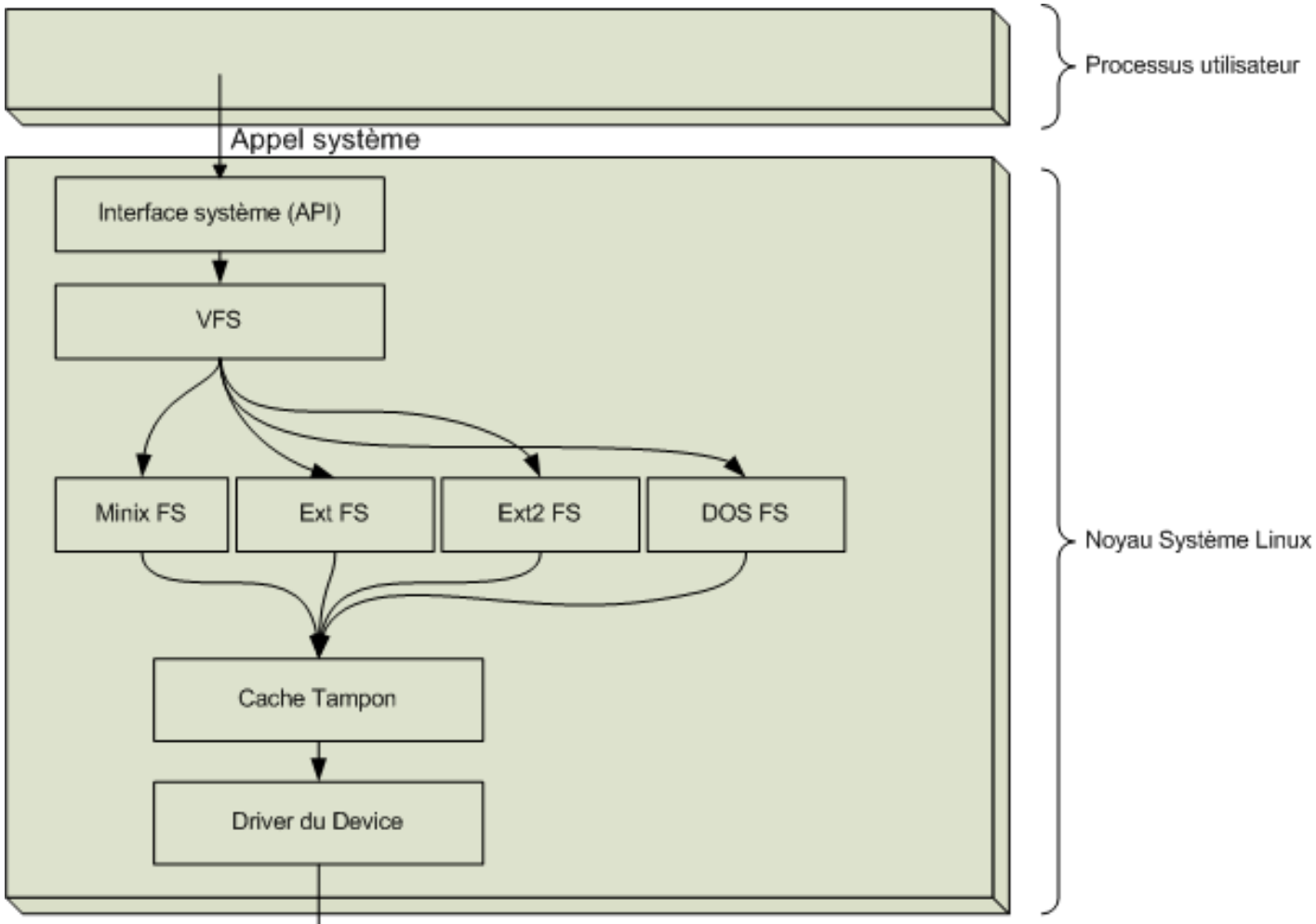
- Le système de fichiers virtuel (VFS) est une couche qui prend en charge les appels systèmes et exécute les 'vraies' routines définies par le système de fichiers réel pour effectuer les opérations E/S
- Ce mécanisme d'indirection permet une généricité et l'intégration facile de plusieurs système de fichiers dans un même système

Virtual File System (VFS)

1. Quand il y'a un appel système concernant le système de fichiers, le noyau fait appel à une routine (fonction) contenue dans le VFS.
2. Cette routine gère tous les aspects génériques et indépendant d'une implémentation particulière d'un système de fichier
3. Une redirection est ensuite effectuer vers le code de la routine du système de fichier réel.
4. Cette routine sera responsable de gérer les aspects spécifiques à cette implémentation particulière du système de fichier.

Virtual File System (VFS)

Ce mécanisme est illustré par le schéma suivant :



Utilisateurs et Groupes

- Pour utiliser le système chaque utilisateur doit être déclaré
- Un utilisateur déclaré possède un unique identifiant : User ID (UID)
- Un utilisateur peut appartenir à un ou plusieurs groupes
- Chaque groupe est identifié par un unique identifiant : Group ID (GID)

Gestion des droits

- Avec le système de fichier d'Unix on peut affecter des droits aux fichiers
- Ces droits concernent des actions de type : lecture, écriture et exécution
- En anglais : Read, Write et eXecute
- Ces droits sont attribués à l'Utilisateur, son Groupe, les autres (Others)

Gestion des droits

On construit ainsi une sorte de matrice des droits pour un fichier :

test.sh	Read	Write	Execute
User	1	1	1
Group	1	1	0
Others	0	0	0

- Pour voir cette matrice des droits on peut essayer la commande **ls -l test.sh**
- Elle montre l'écriture de cette matrice sur une ligne :
- **-rwxrw-r— test.sh**

- **-rwxrw-r— test.sh**
- Le premier caractère indique s'il s'agit d'un répertoire ou d'un fichier standard (d)
- Les caractères suivants sont regroupés en groupe de trois
- Chaque groupe indique les autorisations (R, W et X) pour respectivement :
  - User : **-rwxrw-r— test.sh**
  - Group : **-rwxrw-r— test.sh**
  - Others : **-rwxrw-r— test.sh**

Changer les droits d'un fichier

- La commande **chmod** permet de changer les droits d'un fichier
- **chmod <qui ?><opération ?><droit ?>**
- Exemples :
  - donner le droit de lecture au groupe : **chmod g+r test.sh**
  - Donner le droit d'exécution l'utilisateur et retirer le droit d'écriture au groupe et aux autres :
  - **chmod u+x,g-rw,o-rw test.sh**

chmod

- On peut utiliser également une écriture en octale pour la commande **chmod**
- On met 1 pour accorder le droit, 0 pour retirer le droit
- On fait 3 regroupements de 3 bits : octale
- Chaque groupe de 3 bits désigne les droits de : User Group Others

chmod

Exemple **chmod 640 test.txt**

- $6 = 110 \mapsto$ R+,W+,X-
- $4 = 100 \mapsto$ R+,W-,X-
- $0 = 000 \mapsto$ R-,W-,X-

Droits par défaut

- Il existe des droits par défaut utilisés à la création de nouveaux fichiers
- Droits maximaux pour :
  - Fichier : 666
  - Répertoire : 777

Droits par défaut

- La commande **umask** permet de modifier ces droits
- On tape **umask XYZ**
  - Les droits des fichiers seront 666 - XYZ (en octale)
  - Les droits des répertoires seront 777 - XYZ (en octale)
- Exemple **umask 022**
  - Les droits des fichiers seront 644
  - Les droits des répertoires seront 755

Changer de propriétaire et de groupe

- La commande **chown** permet de changer de propriétaire de fichier
- La commande **chgrp** permet de changer de groupe pour un fichier

Droits particuliers

- le drapeau setuid : exécution avec les droits du propriétaire du fichier et non de celui qui exécute le fichier
- **chmod u+s test.sh**
- le drapeau setgid : exécution avec les droits du groupe propriétaire et non du groupe de l'exécutant

Droits particuliers

- sticky bit pour un répertoire :
- droit de création à tous
- suppression et modification uniquement aux propriétaires
- **chmod o+t montemprep**

Descripteur de fichier

Descripteur c'est quoi ?

- Toutes les opérations E/S utilisent un identifiant pour désigner un fichier ouvert
- Cet identifiant est appelé le descripteur
- En général c'est un entier $\geqslant 0$

Universalité du descripteur

- Un descripteur désigne différents types de fichiers :
- Tubes (Pipes)
  - FIFOs
  - Sockets
  - Devices
  - Fichier standard

Descripteurs de fichier pour un processus

- Chaque processus possède son propre ensemble de descripteurs de fichiers
- Deux processus peuvent avoir deux valeurs différentes de descripteurs qui désignent le même fichier (inode)
- Par convention, il existe 3 descripteurs particuliers

Descripteur	Nom	Posix	Stream
0	entrée standard	STDIN_FILENO	stdin
1	sortie standard	STDOUT_FILENO	stdout
2	sortie des erreurs	STDERR_FILENO	stderr

Les opérations d'E/S

- L'API sur les E/S se résume en 4 opérations élémentaires :
  1. `fd = open(pathname, flags, mode)`
  2. `numread = read(fd, buffer, count)`
  3. `numwritten = write(fd, buffer, count)`
  4. `status = close(fd)`

Ouverture d'un fichier

- `fd = open(const char* pathname, int flags, mode_`
- Ouvre le fichier désigné par le chemin pathname
- retourne le descripteur de fichier qui sera utilisé par la suite
- retourne -1 en cas d'erreur

Lecture dans un fichier

- `size_t read(int fd, void *buffer, size_t count);`
- Lecture du fichier fd de count octets
- Range les données lues à l'addresse buffer
- Retourne le nombre de données effectivement lues

Ecriture dans un fichier

- `numwritten = write(int fd, void* buffer, size_t`
- Ecriture de count données à partir de l'addresse buffer
- Retroune le nombre de données écrites

Fermeture du fichier

- `status = close(fd)`

Lignes directrices

lseek

```
#include <unistd.h> off_t lseek
(int fd,
off_t offset,
int whence);
```

- Retourne le nouveau décalage ou -1 en cas d'erreur
- Le noyau maintient un offset (décalage) pour chaque fichier ouvert
- Le premier octet du fichier est à l'offset 0
- Après un open, l'offset vaut 0
- Après un read ou write l'offset est mis à jour pour prendre en compte les octets lus/écrits

Trous dans un fichiers !

- Avec la fonction lseek nous pouvons déplacer le curseur
- Nous pouvons écrire dans où se trouve le curseur
- Nous pouvons créer des trous dans un fichier !
- Ce sont des trous logiques et la lecture à ces addresses retourne null (0)