Systèmes de Gestion de Fichiers

Plan

- Désignation des fichiers
- Organisation arborescente
- Montages de systèmes de fichiers
- Virtual File System
- Exemple SGF : Unix File System

Noms Internes & Externes

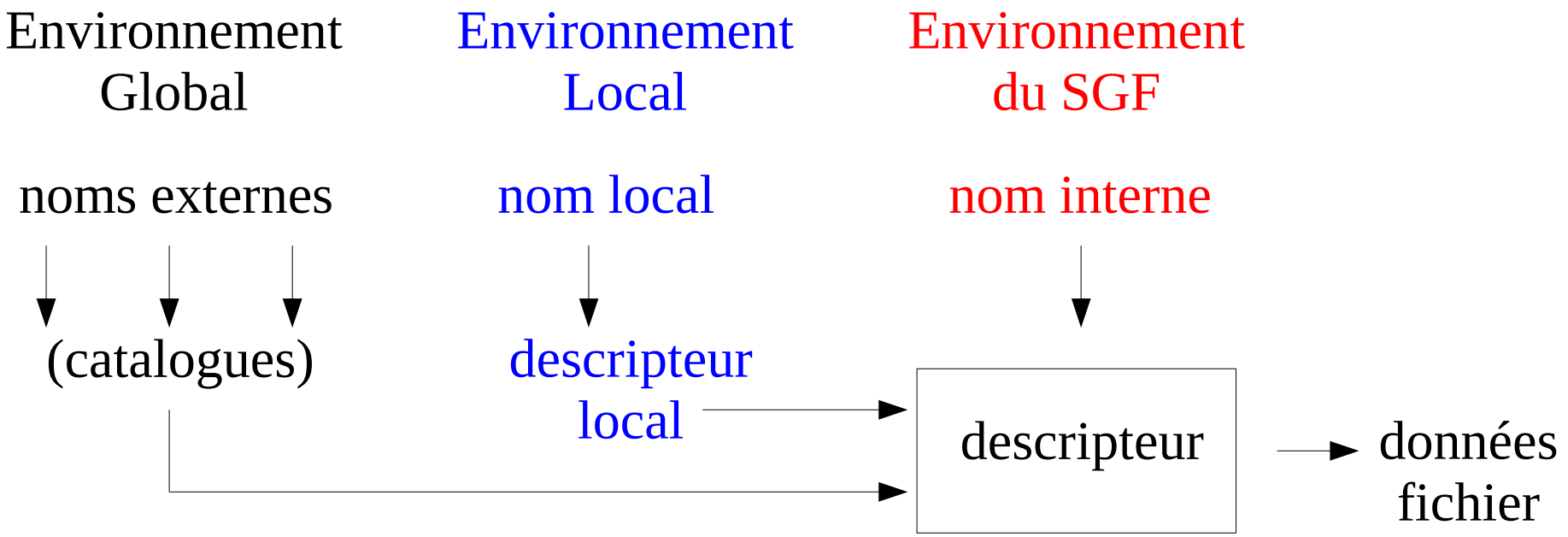
- Fichier est un objet composé
  - Descripteur (localisation physique, droits, etc...)

    désigné dans OS par un *nom interne*
  - Contenu du fichier (données utiles)
- Fichier désigné globalement par un *nom externe*
  - Structuré par l'intermédiaire de répertoires

    (« directory »)
  - Représentant chemin d'accès (« pathname »)

Nom Local d'un fichier

- Nom local = nom temporaire défini dans l'environnement (local) d'un processus
  - Entier >= 0 (« file descriptor ») dans Unix



Avantages nom local fichier

- Efficacité : interprétation plus rapide que nom externe
  - Interprétation nom externe et parcours des répertoires 1 seule fois à l'ouverture du fichier
  - Noms locaux utilisés dans opérations d'entrées/sorties
- Commodité : noms locaux prédéfinis pouvant désigner des fichiers différents à des instants différents, sans avoir à modifier les programmes (redirection E/S standards)

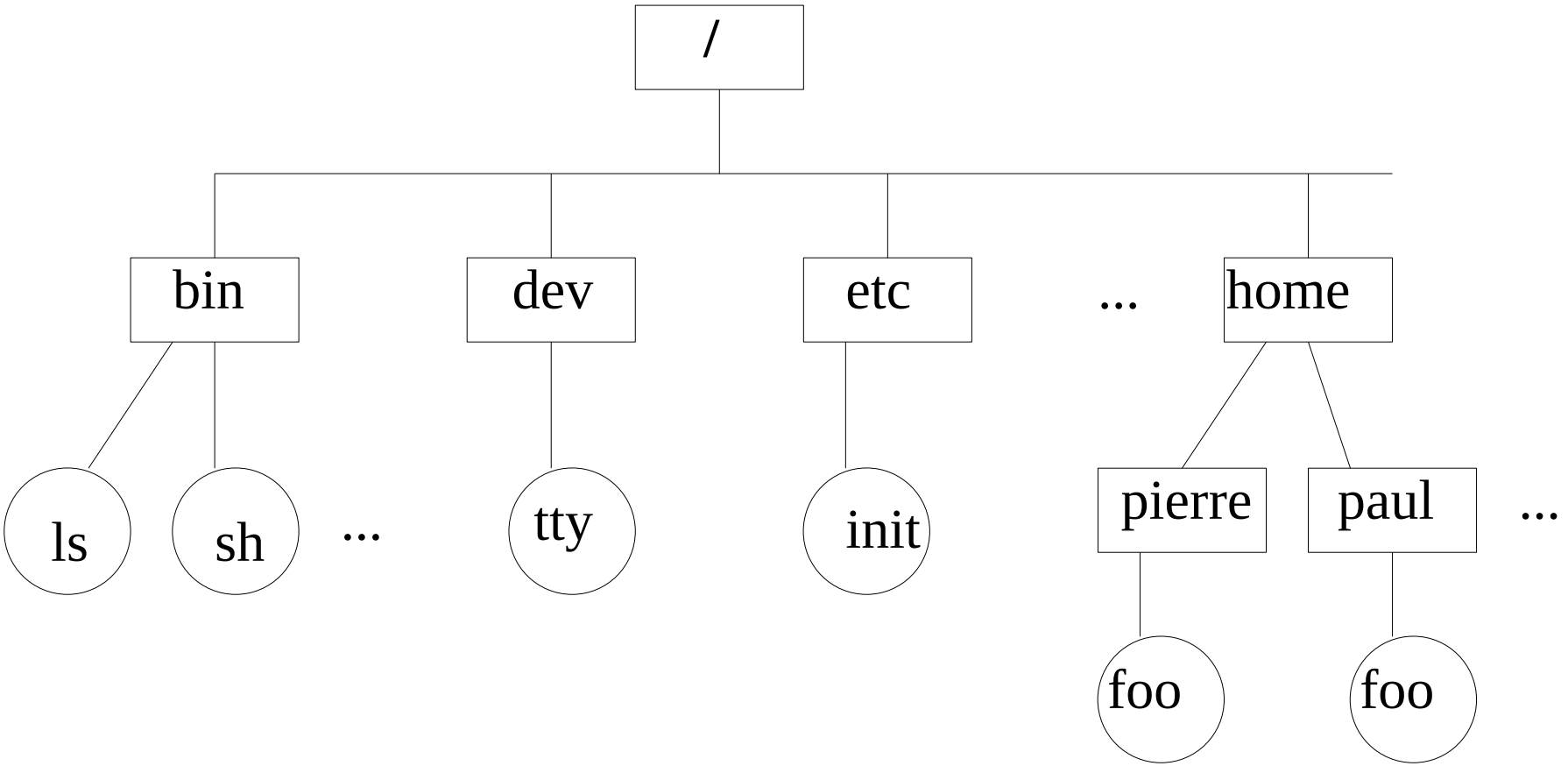
Plan

- Désignation des fichiers
- Organisation arborescente
- Montages de systèmes de fichiers
- Virtual File System
- Exemple SGF : Unix File System

Organisation Arborescente

- Organisation hiérarchique arborescente
  - Répertoire contenant des répertoires et des fichiers
  - Relation de filiation entre répertoires pères et fils
  - Fichiers = feuilles de l'arbre
- Fichiers et répertoires identifiés par un *nom simple* dans leur répertoire père
- 1 seul arbre & répertoire racine (Multics, Unix)
- Plusieurs arbres & racines (Windows)

Arboresence Unix



Noms Composés

- Chemin unique depuis un répertoire de l'arboresence pour atteindre chaque descendant
- Nom descendant = concaténation des noms simples de tous les répertoires ancêtres + nom simple du descendant
- Nom externe est un *nom composé* avec un symbole spécial séparant les noms simples
  - « > » dans Multics
  - « / » dans Unix
  - « \ » dans Windows

Environnement de Désignation

- Répertoire racine désigné par le symbole du séparateur (« / » dans Unix)
- Nom absolu = nom composé commençant par symbole désignant la racine
  - « /bin/ls » , « /home/pierre/foo »
- Environnement de désignation
  - ensemble des noms (simples et composés) des répertoires et fichiers descendants d'un répertoire
- Environnement Universel = environnement de désignation depuis la racine du système

Contexte de Désignation (1)

- Environnement de désignation par processus
  - restreindre à un sous-arbre de l'environnement universel
- Contexte système de chaque processus inclut répertoire racine (défaut = racine du système)
- Appel système spécial pour changer répertoire racine du processus courant
  - `chroot ("/home/pierre")`
  - => `/foo` désigne `/home/pierre/foo` dans processus

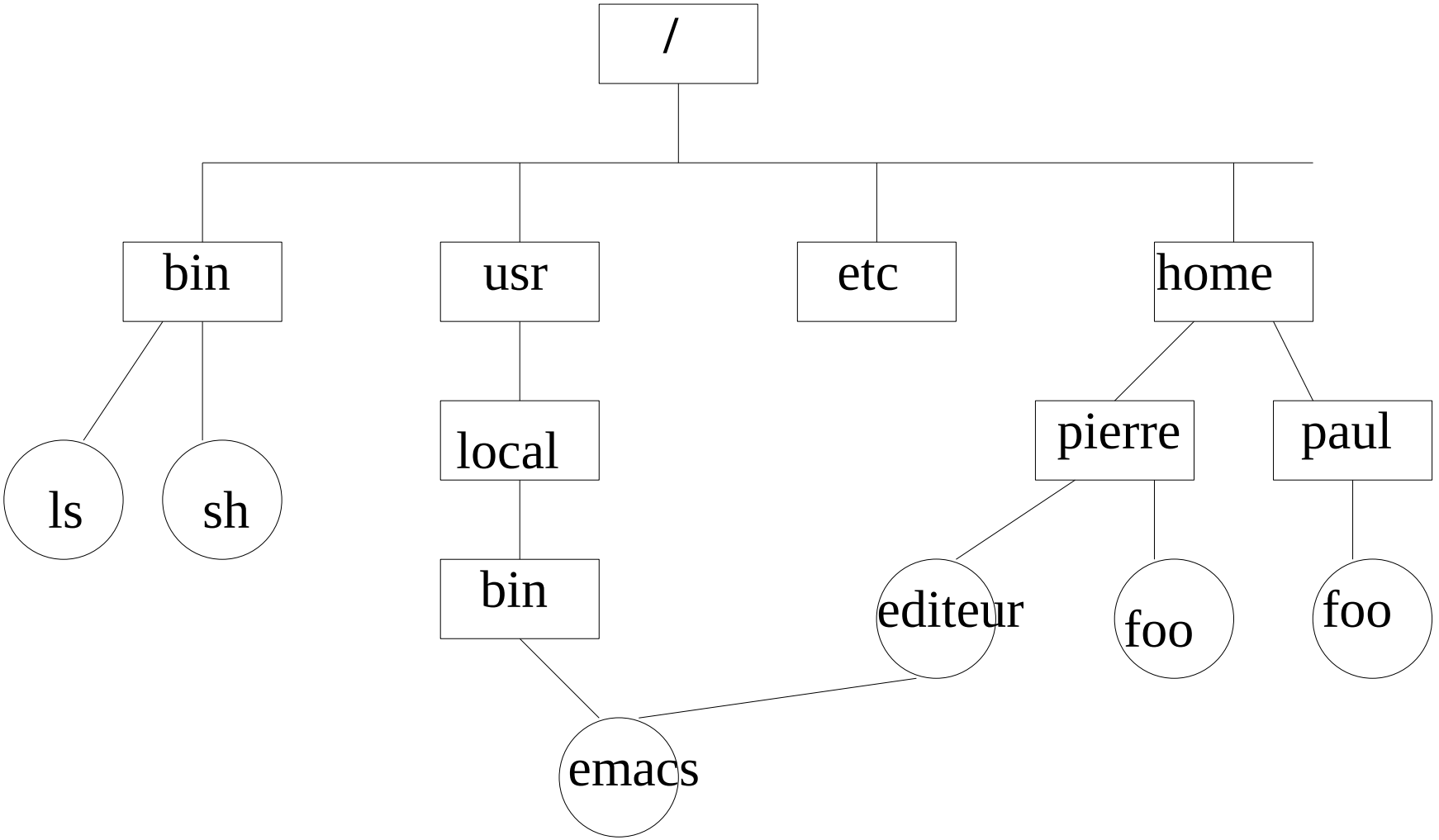
Contexte de Désignation (2)

- Contexte système de chaque processus inclut répertoire courant (défaut = répertoire racine)
- *Nom relatif* : nom composé commençant par un nom simple du répertoire courant du processus
  - « pierre/foo », « foo »,
- Désignation du répertoire père d'un catalogue par un symbole spécial (« .. » dans Unix)
  - « ../paul », « ../../foo », « /home/paul/../pierre »
- Désignation du répertoire courant par symbole spécial (« . » dans Unix)

Liens Physiques

- Noms absolus différents désignant un même fichier
- Nom cible interprété à la création du lien
  - Doit désigner un fichier existant
- Nombre liens physiques dans descriptif du fichier
  - Tous les liens physiques ont statut identique
  - Fichier détruit quand dernier lien physique détruit

Lien Physique - exemple



Liens Symboliques

- Fichier particulier contenant un nom absolu ou relatif d'un fichier, répertoire ou autre lien cible
- Nom cible interprété à chaque utilisation du lien
  - création lien symbolique vers fichier cible inexistant
  - destruction fichier sans effet sur liens symboliques
  - Destruction d'un lien symbolique ne permet pas de détruire le fichier cible
- Nom cible inséré avant partie restante du nom source analysé
- Nombre max de liens traversés par chaîne d'accès => protection contre circuits

Conventions de Nommage

- Suffixes par types de fichiers
  - `.c` pour fichiers source en langage C
  - `.o` pour fichiers objets produits par compilateurs
- Permettent classement de fichiers associés dans un même répertoire
- Facilitent traitements automatiques

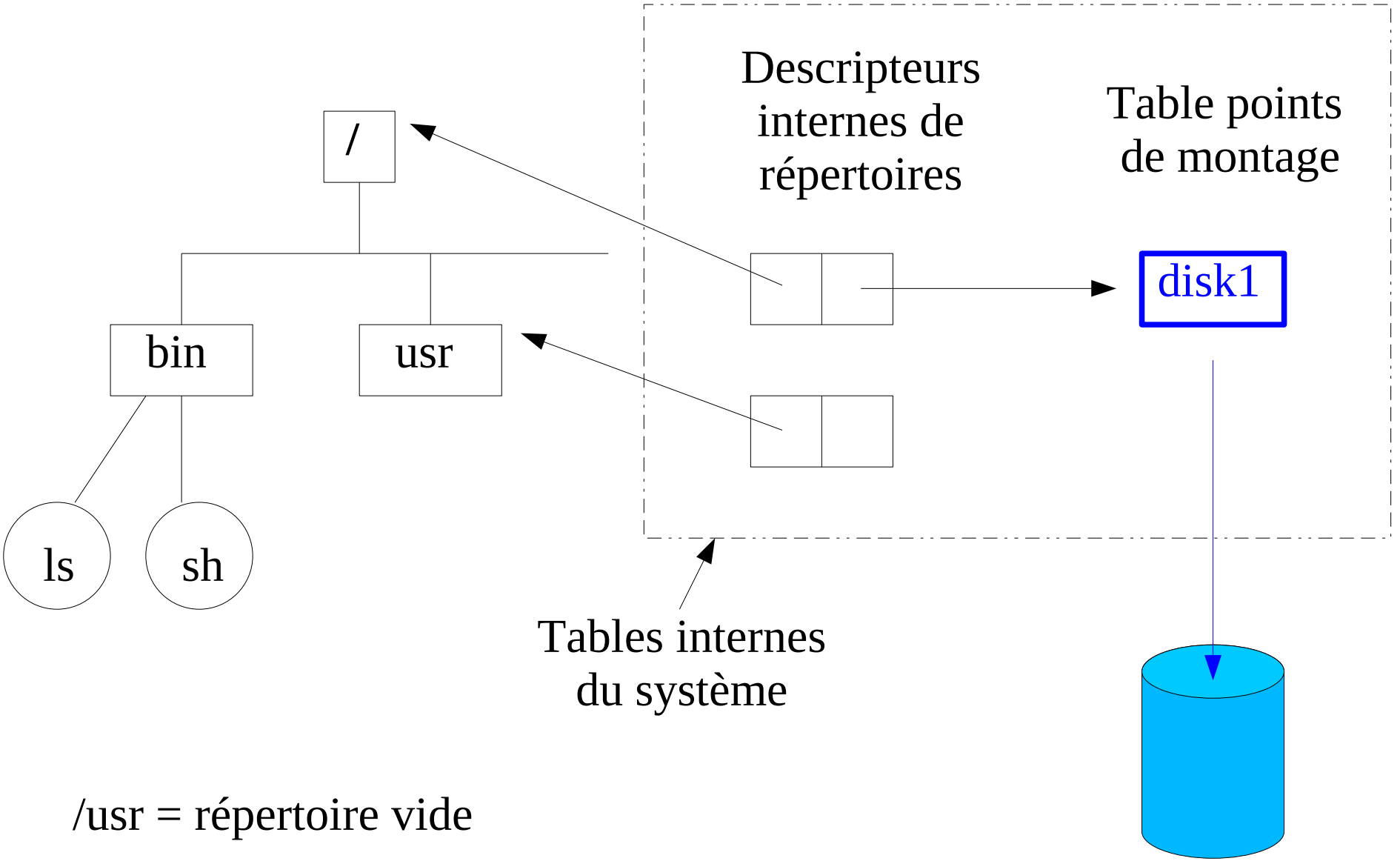
Plan

- Désignation des fichiers
- Organisation arborescente
- Montages de systèmes de fichiers
- Virtual File System
- Exemple SGF : Unix File System

Montage de Systèmes de Fichiers

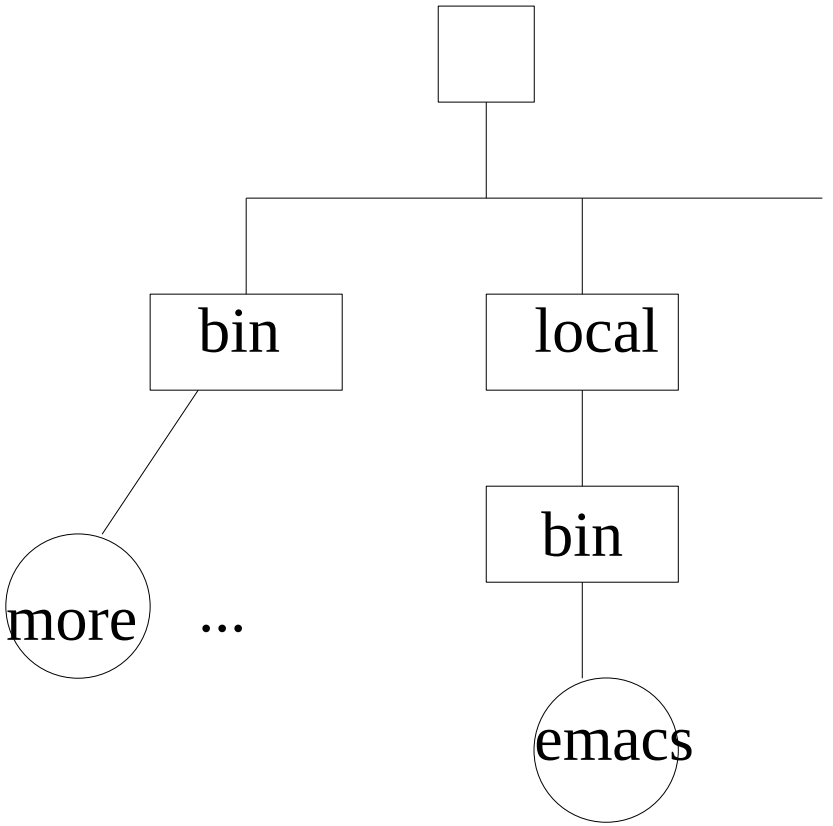
- Système de fichiers = ensemble de répertoires et de fichiers stockés sur un volume amovible
- Par extension, arborescence de répertoires et de fichiers accessible globalement de façon indépendante
- Montage d'un système de fichiers
  - Intégration dynamique d'un système de fichiers dans l'environnement courant du système
  - Liaison entre répertoire racine du système de fichiers et un répertoire de l'arborescence courante

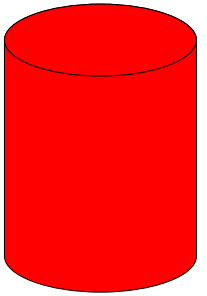
Montage système de fichiers (1)



/usr = répertoire vide

Montage système de fichiers (2)



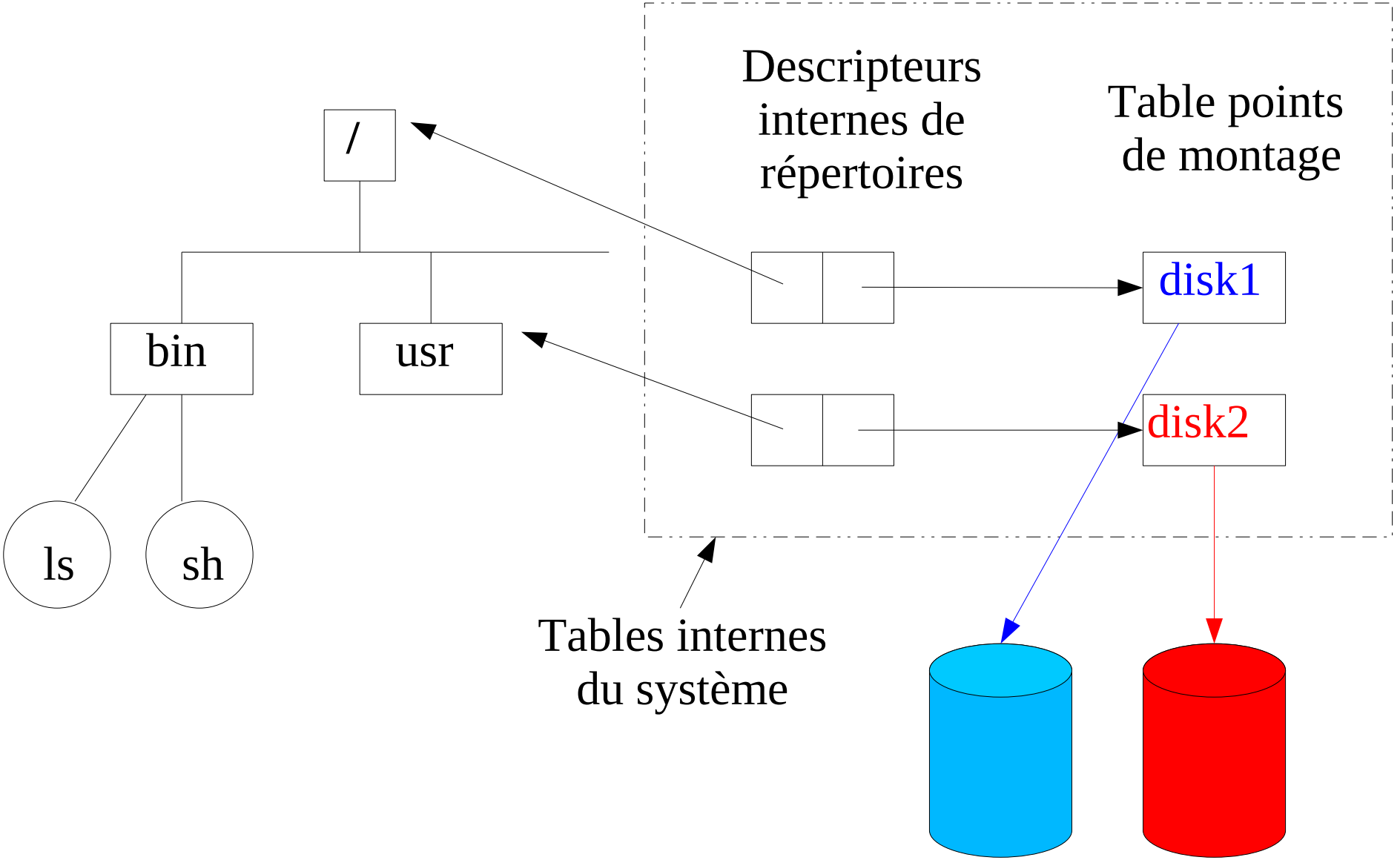


disk2

Montage système de fichiers (3)

- `mount disk2 /usr`
  - Intègre système de fichiers de `disk2` dans arborescence du système sous le répertoire `/usr`
- répertoire racine du système de fichiers de `disk2` désigné par `/usr`
  - `/usr/local/bin/emacs` désigne le fichier `local/bin/emacs` sur le disque disk2

Montage système de fichiers (4)



Démontage système de fichiers

- `umount /usr`
  - Défait association répertoire `/usr` et le sous-système de fichiers auquel il donne accès
- Assure cohérence du contenu du sous-système de fichiers avant démontage volume amovible de son support
- Opération refusée par système tant que fichiers du sous-système accédés par processus

Plan

- Désignation des fichiers
- Organisation arborescente
- Montages de systèmes de fichiers
- Virtual File System
- Exemple SGF : Unix File System

Systèmes de Gestion de Fichiers

- Existe différents types de systèmes de fichiers
  - Structure interne
  - Méta-données
    - Types de fichier
    - Dates de création, de modification, de lecture
    - Droits d'accès
  - Localisation des informations sur support
- Exemples
  - MSDOS (**M**icro**S**oft **D**isk **O**perating **S**ystem)
  - UFS (**U**nix **F**ile **S**ystem)

Exemples de SGF

- UFS (**U**nix **F**ile **S**ystem)
  - Standard sur Unix BSD
- NFS (**N**etwork **F**ile **S**ystem)
  - Accès transparent à systèmes de fichiers localisés sur machines distantes
  - ≠ système de fichiers réparti
- CRAMFS (**C**ompressed **R**AM **F**ile **S**ystem)
- MSDOS (**M**icro**S**oft **D**isk **O**perating **S**ystem)

Virtual File System

- Systèmes (ex : Unix) capables de gérer plusieurs types de systèmes de fichiers simultanément
- Chaque type de SGF exporte ses opérations spécifiques à travers interfaces internes génériques
  - Opérations sur systèmes de fichiers (montage, ...)
  - Opérations de manipulation des répertoires et des fichiers (créer répertoire, lire contenu fichier, ...)
- Objets (fichiers, répertoires) représentés en mémoire par descripteurs génériques (*vnode*)

Rôle du VFS

- Mise en oeuvre de l'interprétation des noms composés
  - gestion des liens symboliques, des symboles spéciaux (« / », « .. », « . »)
  - Gestion des points de montages
- Gestion du cache
  - Contenu des fichiers géré dans un cache
  - optimiser performances

Représentation Mémoire

- Chaque fichier & chaque répertoire accédé est représenté en mémoire
- Représentation temporaire comprenant
  - une partie générique appellée *vnode*
  - une partie spécifique du SGF (*inode* pour UFS)
- vnode contient notamment :
  - Compteur de références
  - Table des opérations exportées par SGF

Plan

- Désignation des fichiers
- Organisation arborescente
- Montages de systèmes de fichiers
- Virtual File System
- Exemple SGF : Unix File System

Fichiers Unix

- Non structurés
  - Suite linéaire d'octets
  - Contenu des fichiers non interprété par SGF
- Lectures/écritures séquentielles par défaut
- Taille nulle à la création, augmentée au fur et à mesure des écritures séquentielles
- Possibilité de créer des "trous" dans un fichier
  - En déplaçant pointeur d'écriture après taille du fichier
  - N'occupent pas d'espace

Représentation UFS sur disque

- Disque = suite linéaire de blocs de 512 octets
- Décomposé en plusieurs partitions
- Super Block = méta-données
  - Magic number UFS, nombre inodes, nombre blocs de données, liste de blocs libres
- Table des *inodes* : 1 inode par fichiers et par répertoire
- Blocs de données
- Répertoires contiennent uniquement noms simples et numéros inodes correspondants

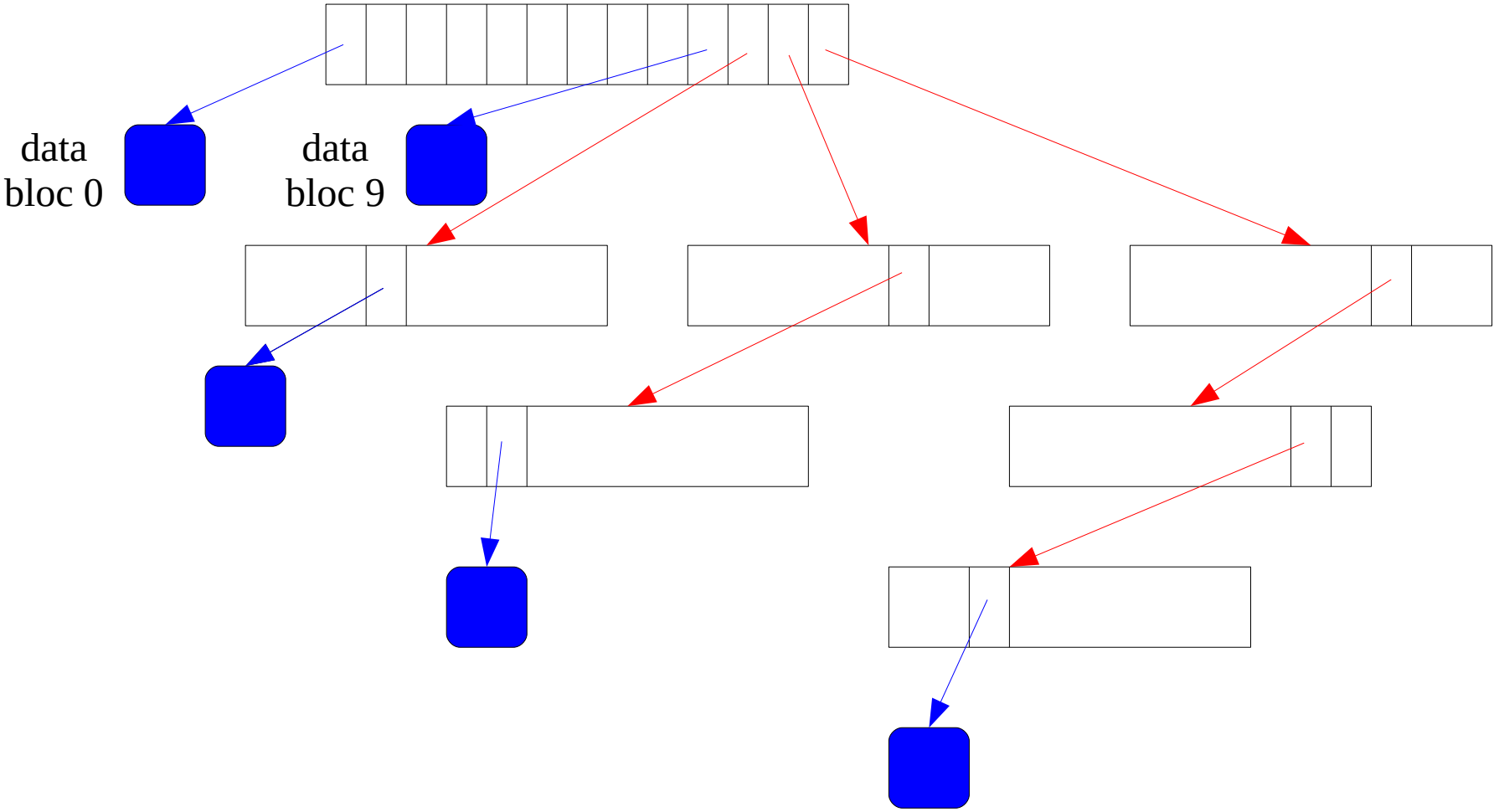
Inode UFS = attributs du fichier

- Type (fichier ordinaire, répertoire, etc)
- Identificateurs propriétaire et de son groupe
- Droits d'accès : propriétaire, membres groupe, autres
- Taille du fichier (en octets)
- Dates de création, de dernière modification, de dernière lecture
- Localisation blocs de données sur disque
  - index bloc = index octet / 512

Implantation Physique

- Inode contient table de 13 numéros de blocs
- Entrées 0 à 9 pour data bloc d'index 0 à 9
- Entrée 10 pour bloc d'indirection niveau 1
  => 128 data blocs d'index 10 à 137
- Entrée 11 pour bloc d'indirection niveau 2
  => 128^2 data blocs d'index 138 à 16522
- Entrée 12 pour bloc d'indirection niveau 3
  => 128^3 data blocs d'index 16523 à 2 113 674

Table de blocs à 3 niveaux



Caractéristiques UFS

- Temps d'accès aux données non uniforme
  - Privilégie petits fichiers
- Caractéristiques système de fichiers UFS fixées à sa création
  - Taille d'un bloc (512 octets)
  - Nombre de blocs
  - Nombre inodes

Améliorations

- Passer taille bloc de 512 à 1024 octets pour améliorer performances
  - Chaque E/S disque manipule 2 fois plus données
  - Augmente taille max. fichiers sans bloc d'indirection
- Taille bloc 2^n avec n <= 10 (4096 octets)
  - Permet créer fichiers de taille de 2^{32} octets
  - Support simultané de systèmes de fichiers de tailles de blocs différentes
- Inconvénient : augmente fragmentation interne