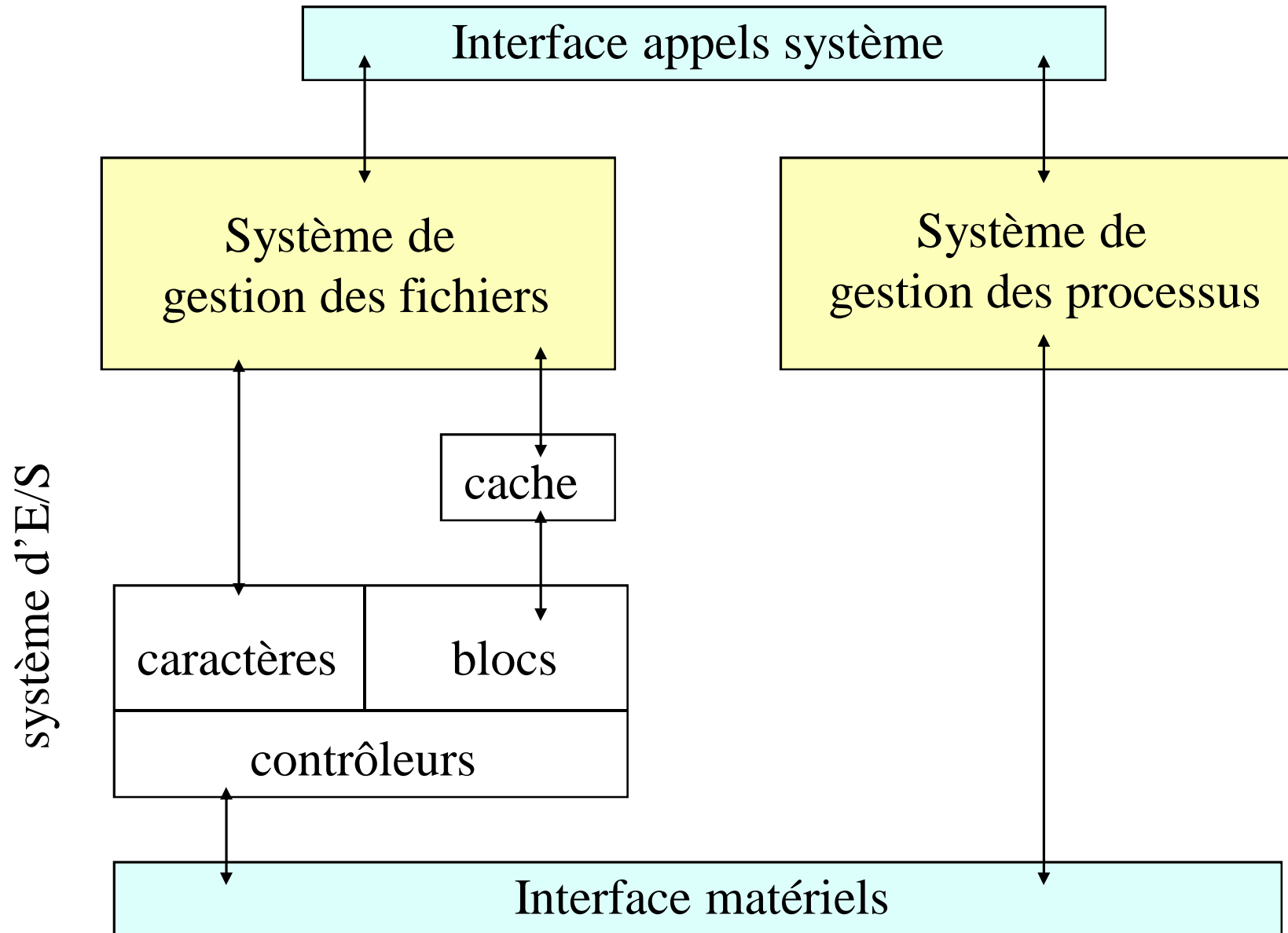


UNIX: Système de Gestion de Fichiers

INTRODUCTION

- **Système de Gestion de Fichiers**
 - Sous système Unix
 - Les fichiers :
 - fichiers du système d'exploitation
 - programmes
 - bibliothèque
 - paramètres
 - périphériques gérés à travers des fichiers
 - moyens de communications entre processus
 - fichiers de données



- **Les périphériques**
 - Système de gestion des **E/S**
 - couche entre le S.G.F et le matériel
 - Primitives d'accès aux périphériques dans les **pilotes**
 - pilotes périphériques type **caractères**
 - pilotes périphériques type **blocs**
 - Actions sur les **contrôleurs**
 - un contrôleur par type de périphérique
 - Caractéristique Unix:
 - Interface d'appel périphérique utilise le SGF
 - un périphérique = un fichier type caractère ou bloc

- **Les moyens de communications Unix**
 - tube anonyme
 - fichier sans nom
 - type FIFO
 - communication entre processus de même filiation
 - tube nommé
 - fichier avec nom
 - type FIFO
 - communication entre processus quelconques
 - socket
 - fichier de communication entre processus distants

GÉNÉRALITÉS

- **Le système de fichiers**
 - gestion de **un** ou **plusieurs systèmes** de fichiers
 - **montage** possible: système unique
 - un système de fichiers accessible sur disque par:
 - un n° périphérique logique
 - n° du système = n° de périphérique**
 - des adresses physiques calculées par le **pilote**

- **Fichier:**
 - ensemble de **données**: **fichier logiciel**
 - répertoire (d)
 - ordinaire : programmes, données.... (-)
 - tubes nommés (p)
 - sockets (s)
 - liens (l)
 - ressource **périphérique**: **fichier matériel**
 - bloc (b)
 - caractères (c)
 - identifié par un **nom**
 - généralement sur **disque**

- **Organisation/disque**

- **bloc**

- unité d'échange: 512, 1024... octets

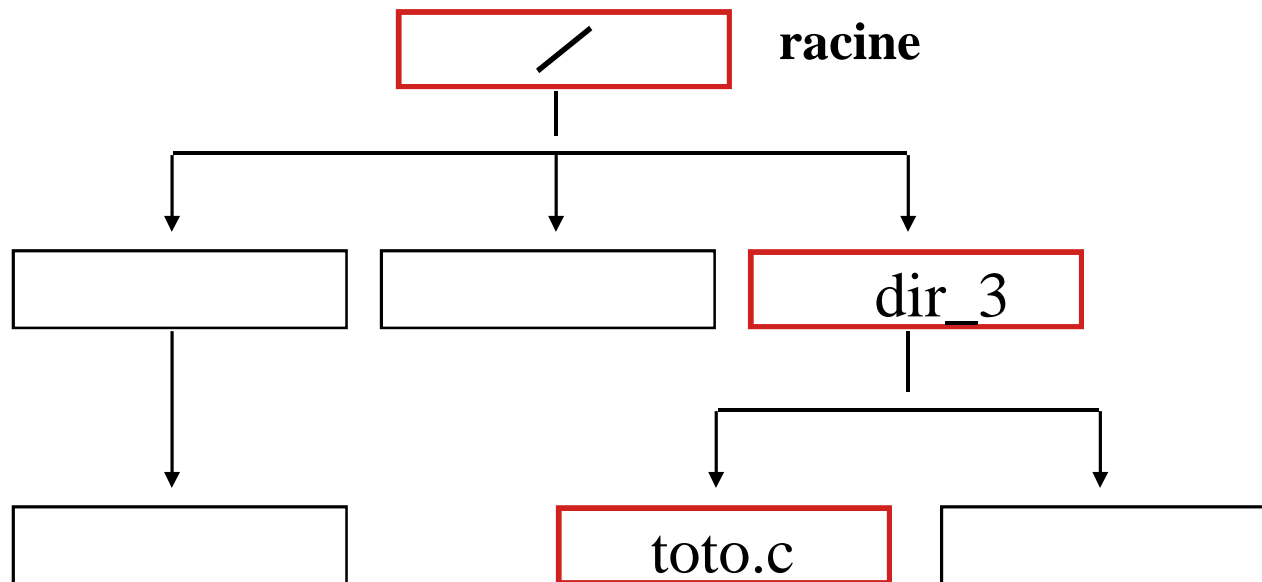
- **blocs du système**

- **boot:** 1er bloc ou autres blocs sur disque
 - **super-bloc:** état du système de fichiers
 - n° premiers blocs libres
 - liste des i-noeuds libres
 - **table des i-noeuds** configurée à l'installation
 - **blocs des données**



- **Le super bloc - Gestion de l'espace disque**
 - la taille du S.F.
 - le nombre de blocs libres dans le S.F
 - une liste des blocs libres disponibles dans S.F
 - l'indice du 1er bloc libre dans la liste des blocs libres
 - la taille de la table des i-noeuds
 - le nombre d'i-noeuds libres dans le S.F
 - une liste d'i-noeuds libres dans le S.F.
 - l'indice du 1er i-noeud libre dans la liste des i-noeuds libres
 - des champs verrous pour les listes des blocs et i-noeuds libres
 - un drapeau indiquant si le super bloc a été modifié

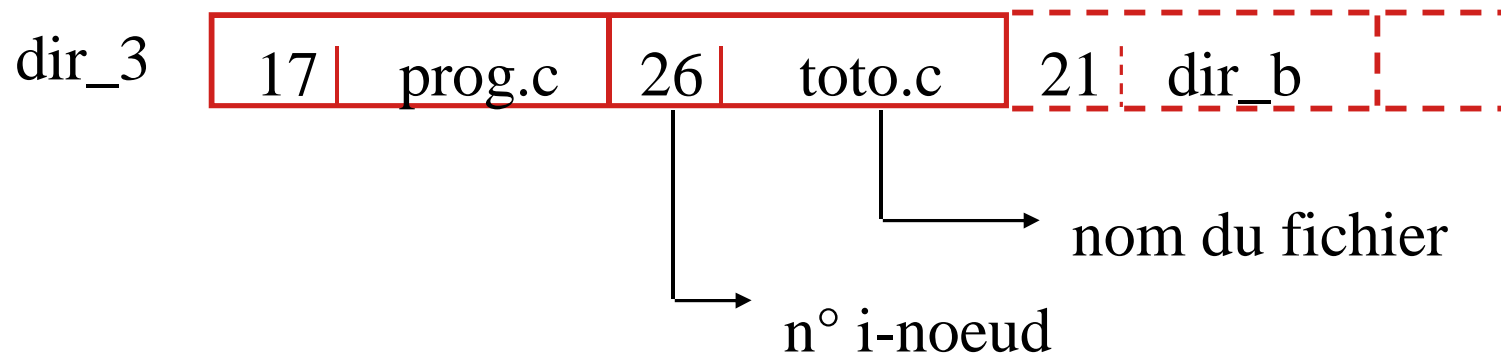
- **Arborescence**



- situer un fichier dans l'arborescence
 - son nom
 - son chemin d'accès: **/dir_3/ toto.c**
- fichiers système: idem mais avec protections renforcées

- **Notion de fichier**

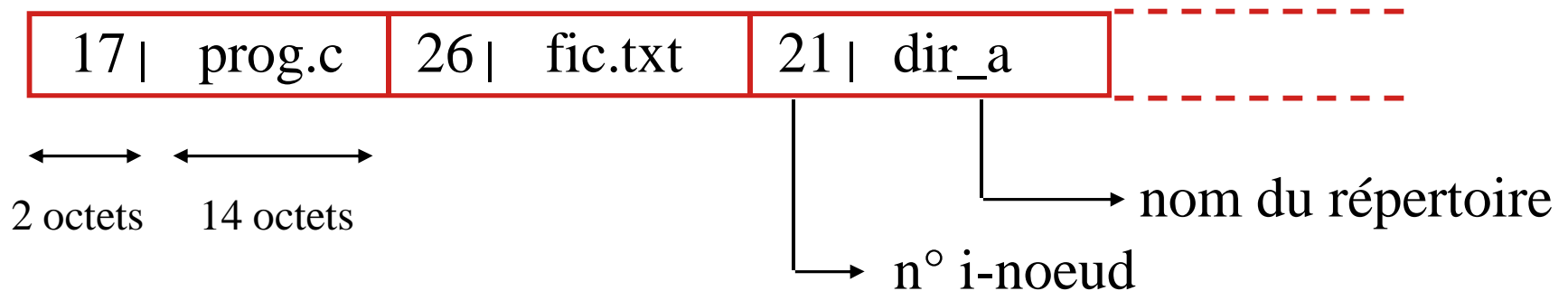
- identification: **nom du fichier**
- localisation fichier par l'algorithme '**namei**'
 - parcourt des blocs de données type '**répertoire**'
 - recherche du nom du fichier



- **i-noeud** du fichier dans la table des i-noeuds
 - pointeurs vers blocs de données

- **répertoire**
 - **fichier** référencé dans la table des i-noeuds
 - pointe vers des blocs de données type ‘répertoire’
 - bloc données répertoire = $n * 16$ octets
 - n° i-noeud
 - nom du fichier ou répertoire
 - **racine** /: i-noeud n° 2

bloc de données type ‘répertoire’



- **Algorithmes**

- L'algorithme 'namei' appelle d'autres algorithmes:
 - 'iget' attribuer i-noeud en mémoire
 - 'iput' libérer i-noeud en mémoire
 - 'bmap' maj paramètres noyau pour accès fichier
- D'autres algorithmes:
 - 'alloc' réservation espace fichier sur disque
 - 'free' libération espace fichier
 - 'ialloc' réservation i-noeud pour un fichier
 - 'ifree' libération i-noeud d'un fichier

LES I-NOEUDS

- **définition**
 - i-noeud (i-node) est un descripteur conservant les informations du fichier sur disque
 - i-noeud = noeud d'index
 - 64 caractères/ i-noeud
 - table des i-noeuds configurée à l'installation
 - n°1 blocs défectueux
 - n°2 racine pour le système de fichiers
 - gestion des i-noeuds
 - sur disque
 - en mémoire

- **i-noeud sur disque**
 - un fichier = un i-noeud + bloc de données
 - table des i-noeuds après le super-bloc
 - un n° = n° entrée dans table des i-noeuds du disque
 - i-noeud utilisé par le noyau
 - contenu des informations dans un i-noeud (64 octets):
 - nom propriétaire
 - type du fichier
 - permissions
 - date dernier accès
 - nombre de liens vers d'autres fichiers (répertoire)
 - table des adresses des données
 - taille: nombre d'octets
 - Localisation des données

- **Exemple**

un i-noeud 

Mohamed
etudiants
ordinaire
rwXr--r-x
23 oct 2014 15:30
22 oct 2014 10:02
23 déc 2014 09:03
5412 octets
table d'adresses des blocs de données

propriétaire
groupe
type du fichier
droits d'accès
date dernier accès
date fichier modifié
date i-noeud modifié
taille
adresses données

- **i-noeud en mémoire**
 - appel fichier => copie i-noeud en mémoire
 - ajout d'informations pour accès physique aux données
 - état i-noeud: bloqué ou non
 - n° logique périphérique
 - n° ligne dans table des i-noeuds en mémoire
 - pointeurs: chaînage des i-noeuds en mémoire
 - compteurs de fichiers ouverts

- Contrôle des accès
 - **verrou**: protection des utilisations multiples
 - **flag** : processus en attente d'accès
 - **compteur d'accès** : si zéro demande , i-noeud recopié sur disque

- **Liste des i-noeuds libres en mémoire**
 - si compteur i-noeud =0 alors placé dans cette liste
 - i-noeuds inactifs = un cache
 - réallocation si demande à nouveau
 - technique limitant les accès disque
 - libération des i-noeuds plus anciens si cache plein

– **i-noeud n'est pas en mémoire**

- **recherche** adresse physique de l'entrée de l'i-noeud dans table des i-noeuds disque

N° bloc:

$((n^{\circ}\text{i-noeud}-1) / \text{nb i-noeuds par bloc}) + n^{\circ} \text{ 1er bloc table des i-noeuds}$

$$((8 - 1) / 3) + 10 = 7 / 3 + 10 = 2 + 10 = \underline{12}$$

N° octet:

$((n^{\circ}\text{i-noeud}-1) \text{ modulo } (\text{nb i-noeuds par bloc})) * \text{taille d'un i-noeud}$

$$((8 - 1) \text{ modulo } 3) * 64 = 7 \text{ modulo } 3 * 64 = 1 * 64 = \underline{64}$$

- **copie** i-noeud dans les i-noeuds en mémoire en récupérant une place dans la liste des i-noeuds libres

- **i-noeud est en mémoire**
 - actif
 - verrouillé alors attente
 - non verrouillé compteur accès +1
 - inactif dans le cache
 - réallocation i-noeud mémoire
 - compteur accès +1

- **Libération d'un i-noeud mémoire**
 - libération i-noeud: accès libéré
 - procédure libération i-noeud de la mémoire
 - **iput ()**
 - compteur = compteur - 1
 - si compteur = 0 plus aucun processus
 - copie i-noeud maj sur disque
 - i-noeud mémoire placé dans la liste des i-noeuds libres en mémoire
 - remarque: si compteur de liens nul (plus aucune données dans le fichier) alors suppression des blocs de données

- **Création d'un i-noeud disque**
 - création d'un fichier
 - paramètres:
 - nom du fichier
 - chemin d'accès jusqu'au répertoire
 - procédure d'allocation d'un i-noeud sur disque
 - **ialloc()** recherche i-noeud libre
 - scrute liste des i-noeuds libres dans super-bloc
 - la liste est vide
 - la liste n'est pas vide

liste des i-noeuds libres (super-bloc)

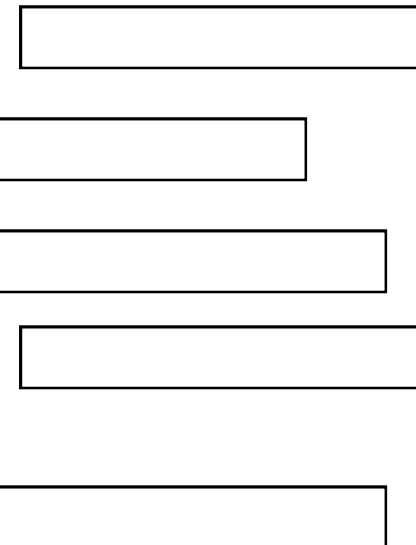
12	17	54	125	127	250	vide
----	----	----	-----	-----	-----	------

table des i-noeuds

entrée
n°250

d
0
-
1
d

blocs de données



- **Cas place dans la liste:**
 - ajout n° i-noeud libéré
 - maj ‘plus grand n° libre’ dans super bloc
 - maj compteur nombre i-noeuds libres
- **Cas liste pleine:**
 - si n° i-noeud libéré < ‘plus grand n° libre’
 - n° i-noeud libéré remplace ‘plus grand n° libre’
 - sinon pas de maj liste
- Dans les deux cas:
 - maj table des i-noeuds disque (type = 0)
 - suppression des blocs de données

L'INTERFACE D'ACCÈS

- Primitives d'accès à un fichier
 - par une **commande** du shell ou script
 - par une **requête** lors d'un appel dans un programme
- Deux types de données
 - données mémoire:
 - procédures système d'accès
 - informations temporaires
 - données disque:
 - données attachées au fichier sur disque

- **Descripteur**
 - infos fichier sont transférées en mémoire
 - un descripteur par fichier et par processus
 - interface entre le processus et le S.F. pour les échanges
- Descripteurs particuliers
 - entrée standard (clavier): n° 0
 - sortie standard (écran): n° 1
 - sortie erreur (fichier erreurs): n° 2
- Structure d'un descripteur:
 - `/usr/sys/include/sys/user.h`

- **Table des fichiers**

- une table des fichiers pour tous les processus gérée par le noyau
- créée à partir de la table des i-noeuds disque
- description des propriétés du fichier et éléments pour l'accès
- un appel fichier = une entrée dans la table
 - avec opération demandée (r,w,r/w)
- structure d'une entrée dans `/usr/sys/include/sys/file.h`
- chaque élément de la table pointe l'i-noeud du fichier dans liste des i-noeuds en mémoire

descripteurs

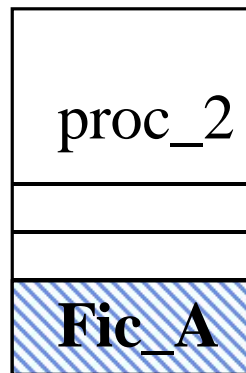
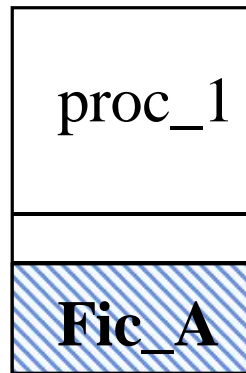
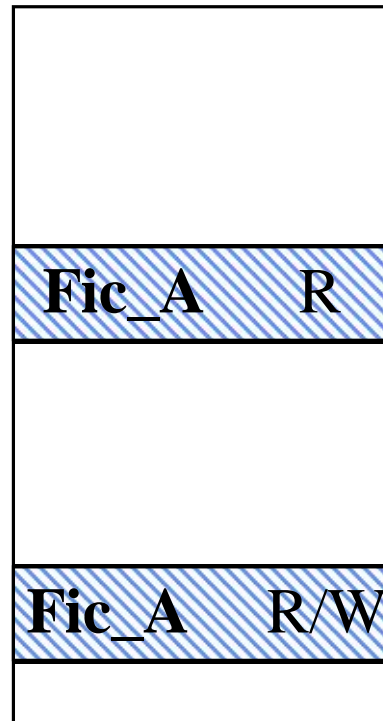
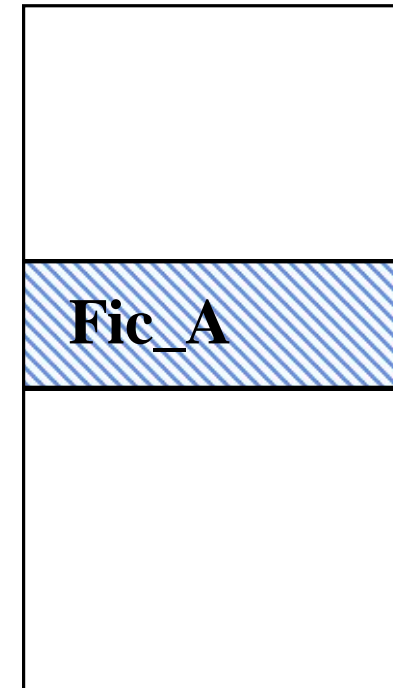


table des fichiers



Liste des i-noeuds



LES PERMISSIONS

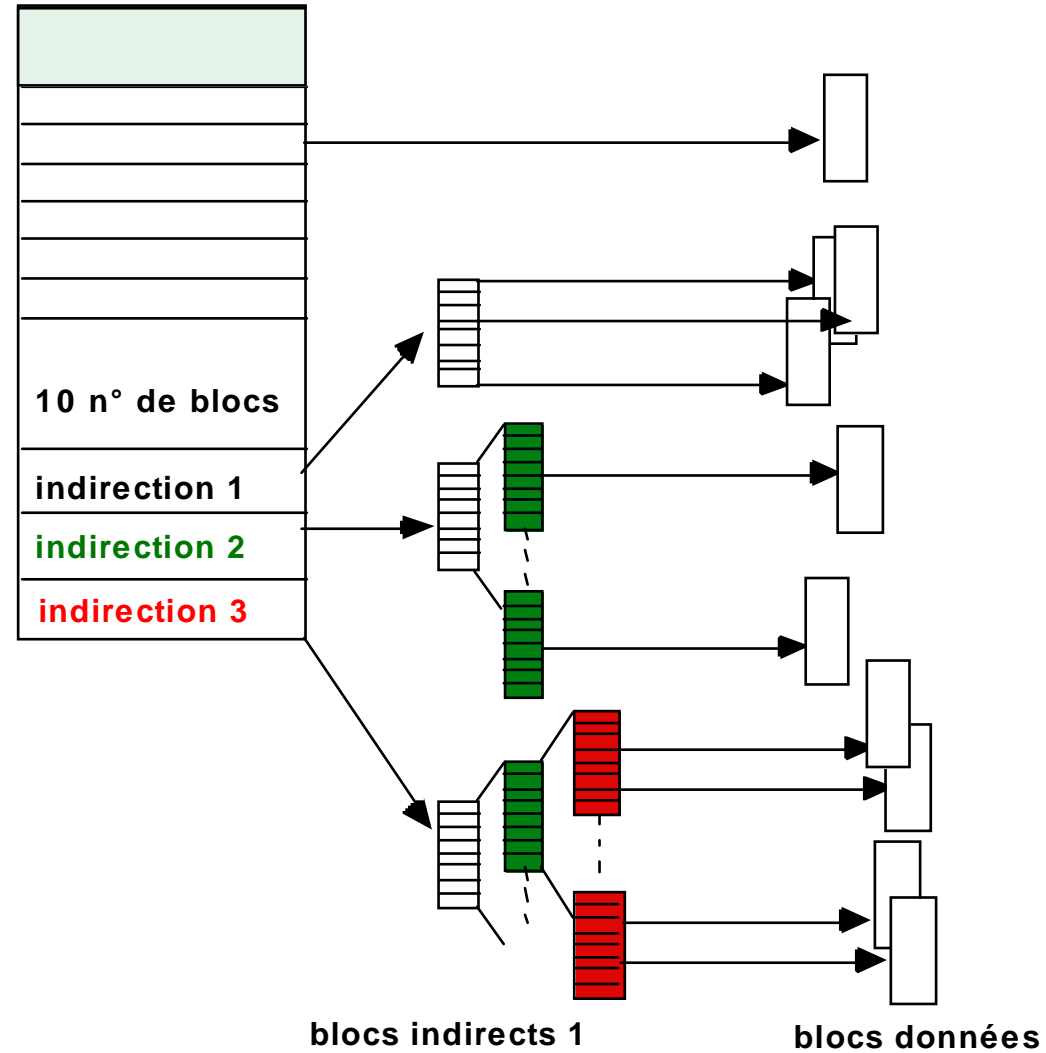
- **Contrôle des accès** aux fichiers renforcé sous Unix:
 - système multi utilisateurs
 - fichiers système et fichiers utilisateurs sous le même système de gestion des fichiers
- **niveaux de protection:**
 - droits -> r: lecture w: écriture x: exécution
 - pour le propriétaire du fichier
 - pour le groupe d'utilisateurs dans lequel se situe le propriétaire
 - pour tous les autres utilisateurs
- **décision des niveaux de protection:**
 - le propriétaire
 - le su

ORGANISATION DES FICHIERS SUR DISQUE

- Organisation
 - non contigüe, aléatoire
 - **Gestion dynamique** de l'espace du disque
- Blocs
 - de taille égale
 - 3 types
 - **blocs de données**
 - **blocs directs**: adresses des blocs de données
 - **blocs indirects**: adresses des blocs directs ou indirects
 - table des i-noeuds sur disque utilise ces différents types de blocs

- Pour chaque i-noeud de la table des i-noeuds:
 - 13 blocs: une **table des adresses des blocs fichiers**
 - 10 blocs directs: 10 n° de blocs de données
 - 3 blocs indirects:
 - n° bloc indirect simple: n° du bloc ayant 256 n° de blocs directs
 - n° bloc indirect double: n° du bloc ayant 256 n° de blocs indirects
 - n° bloc indirect triple: n° du bloc ayant 256 n° de blocs indirects double
- si:
 - adresse sur 32 bits et bloc de 1024 octets
 - => taille maxi / fichier **16 GO**
- un bloc libre est à zéro

Table des adresses
dans un i-noeud



- **Attribution des blocs**
- **super bloc**
 - nb de blocs libres
 - une liste
 - de 50 n° de blocs libres (cache)
 - un pointeur vers d'autres listes de blocs libres
- utilitaire **mkfs**: création S.F.
 - chaînage des blocs libres
- algorithme **alloc()**
 - allocation d'un bloc à un fichier à partir de la liste
 - si liste pleine: n° bloc pris dans la liste
 - si liste épuisée: liste suivante remplace liste super bloc

Exemple:

liste des blocs libres dans super bloc

super bloc

14	21	105	52	25	10		
-----------	----	-----	----	----	----	--	--

pointeur
↓

82	72	27	103	18	46	68	261
----	----	----	-----	----	----	----	-----

bloc 14

Récupération d'un bloc libre:

liste des blocs libres dans super bloc **pleine**

super bloc

14	21	105	52	25	10	107	
-----------	----	-----	----	----	----	------------	--

bloc 107 attribué alors:

super bloc

14	21	105	52	25	10		
-----------	----	-----	----	----	----	--	--

Récupération d'un bloc libre

liste des blocs libres dans super bloc **vide**

super bloc

11							
-----------	--	--	--	--	--	--	--



14	21	105	52	25	10	107	77
----	----	-----	----	----	----	-----	----

bloc n°11

bloc 11 attribué alors:

liste des blocs libres dans super bloc

super bloc

14	21	105	52	25	10	107	77
----	----	-----	----	----	----	-----	----

- **Libération des blocs**
- **super bloc**
 - recherche d'un bloc libre
- algorithme **free()**
 - libération des blocs d'un fichier
 - attachement à la liste des blocs libres du super bloc
 - si liste non pleine: maj dans cette liste
 - si liste pleine: liste pleine écrite dans ce bloc libéré et un pointeur dans la liste sur ce nouveau bloc libre

liste des blocs libres dans super bloc **pleine**

super bloc

14	21	105	52	25	10	107	77
-----------	----	-----	----	----	----	-----	----

bloc 11 libre alors:

super bloc

11							
-----------	--	--	--	--	--	--	--



14	21	105	52	25	10	107	77
----	----	-----	----	----	----	-----	----

bloc n°11

liste des blocs libres dans super bloc **non pleine**

super bloc

14	21	105	52	25	10	107	
-----------	----	-----	----	----	----	-----	--

bloc 11 libre alors:

super bloc

14	21	105	52	25	10	107	11
-----------	----	-----	----	----	----	-----	-----------

- **Répertoires**

- Fichier de type ‘**d**’
- Accéder à un i-noeud d’un répertoire permet d’accéder aux blocs de données du répertoire
- Blocs de données du répertoire comme les fichiers:
 - blocs données
 - blocs directs
 - blocs indirects
- Données au format:
 - n° i-noeud du fichier
 - nom de fichier

– Chemin d'accès

- Définir le chemin d'accès au fichier
- syntaxe:
 - chaîne de caractères
 - plusieurs éléments séparés par /
 - chaque élément est un nom de répertoire
 - dernier élément est le nom du fichier
 - le premier / est le répertoire racine

/users/étudiants/informatique/gaston/fichier.exe

- répertoire particuliers
 - répertoire courant
 - .. répertoire parent
- un n° i-noeud associé à chacun des noms de fichier dont:
 - le répertoire courant
 - le répertoire parent

Bloc répertoire	
1	•
1	• •
4	bin
7	-
5	lib
3	etc
9	usr

↓ nom fichier
 n° i-noeud

i-noeud 9

infos
125

/usr

bloc 125 répertoire

9	•
1	• •
17	sami
57	cyrine
44	ptube.exe

i-noeud 44

infos
688

/usr/prog.exe

- Conversion nom_fichier en i-noeud
 - n° i-noeud et nom fichier dans bloc données du répertoire
 - l'algorithme 'namei'
 - pour chaque niveau convertit nom et n°
 - vérifie les droits d'accès
 - parcourt les niveaux jusqu'à trouver le nom du fichier

LES PRIMITIVES

- Structure commune des fichiers

- dans /usr/sys/include/sys/**stat.h**

struct stat

{ dev_t st_dev;	identif disque logique
ino_t st_ino;	n° fichier sur disque
mode_t st_mode;	type du fichier et droits accès
nlink_t st_nlink;	nb liens physiques
uid_t st_uid;	propriétaire
gid_t st_gid;	groupe
dev_t st_rdev;	identif disque logique (bloc/car)
off_t st_size;	taille en octets
time_t st_atime;	date dernier accès

int st_spare1;	
time_t st_mtime;	date dernière modif
int st_spare2;	
time_t st_ctime;	date dern modif i-noeud
int st_spare3;	
uint_t st_blksize;	taille d'un bloc dans un fichier
int st_blocks;	blocs alloués pour le fichier
uint_t st_flags;	flags utilisateur
uint_t st_gen;	n° génération du fichier
};	

- **Informations fichier**

- récupération dans une structure stat des informations du fichier

```
#include <sys/types.h>
```

```
#include <sys/stat.h>
```

```
int stat (const char *ef, struct stat *pt_stat)
```

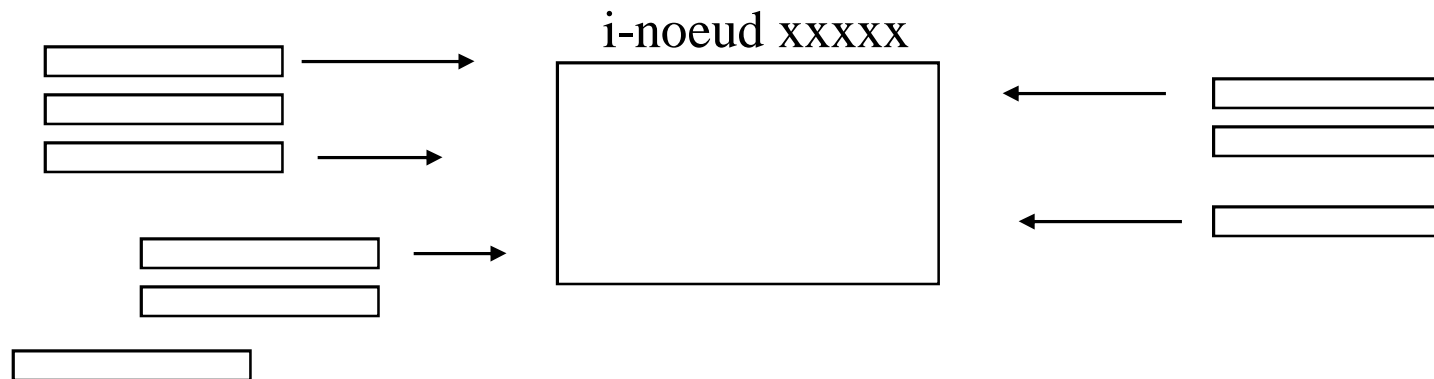
```
int fstat (const int desc, struct stat *pt_stat)
```

```
ex:      struct stat bufstat
```

```
        if (stat("fic1.c", &bufstat) == -1)
```

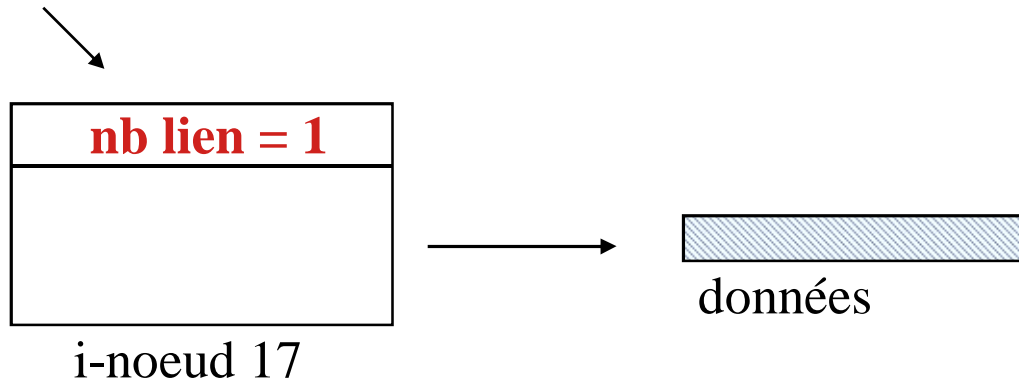
```
        perror ("erreur stat");
```

- Les liens
 - Association: nom fichier <-> i-noeud
 - Nombre de liens: nombre de fois qu'un i-noeud est référencé dans les répertoires



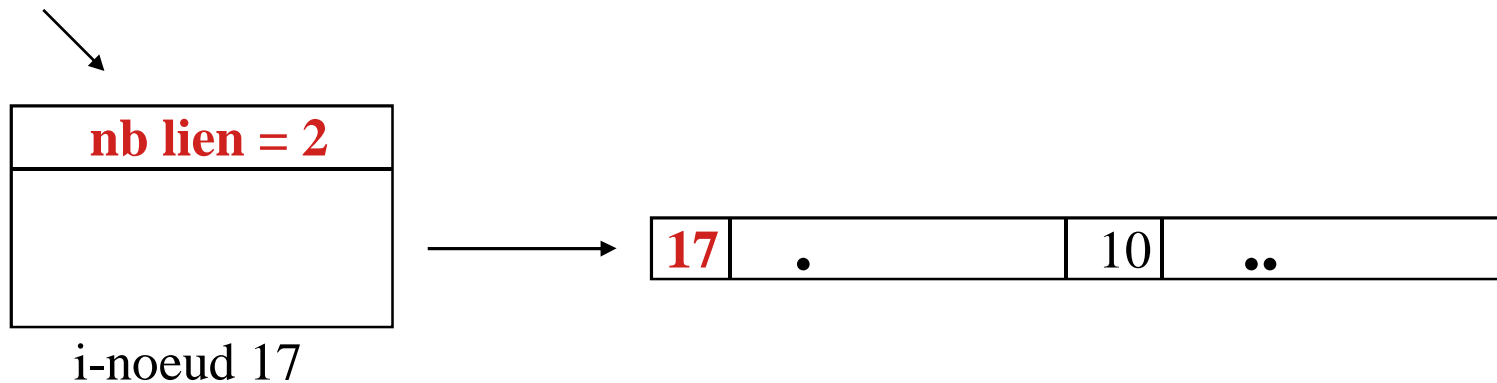
17	Fichier.doc	23	prog.exe
-----------	-------------	----	----------

vers un fichier de données

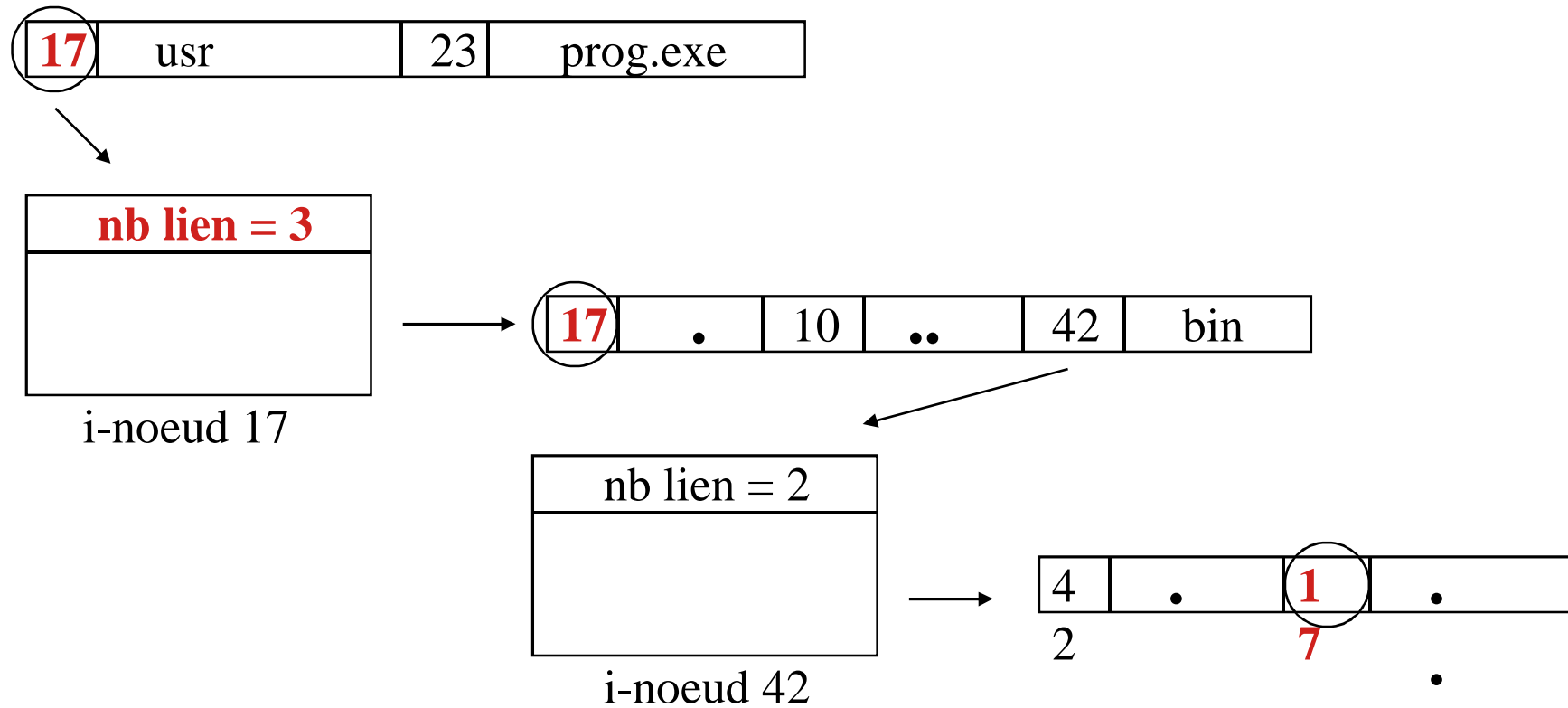


17	usr	23	prog.exe
-----------	-----	----	----------

vers un fichier répertoire



cas d'un fichier répertoire ayant un sous-répertoire



- **Créer un lien synonyme**

- deux entrées de répertoire pointent sur un même fichier de données
- un seul i-noeud

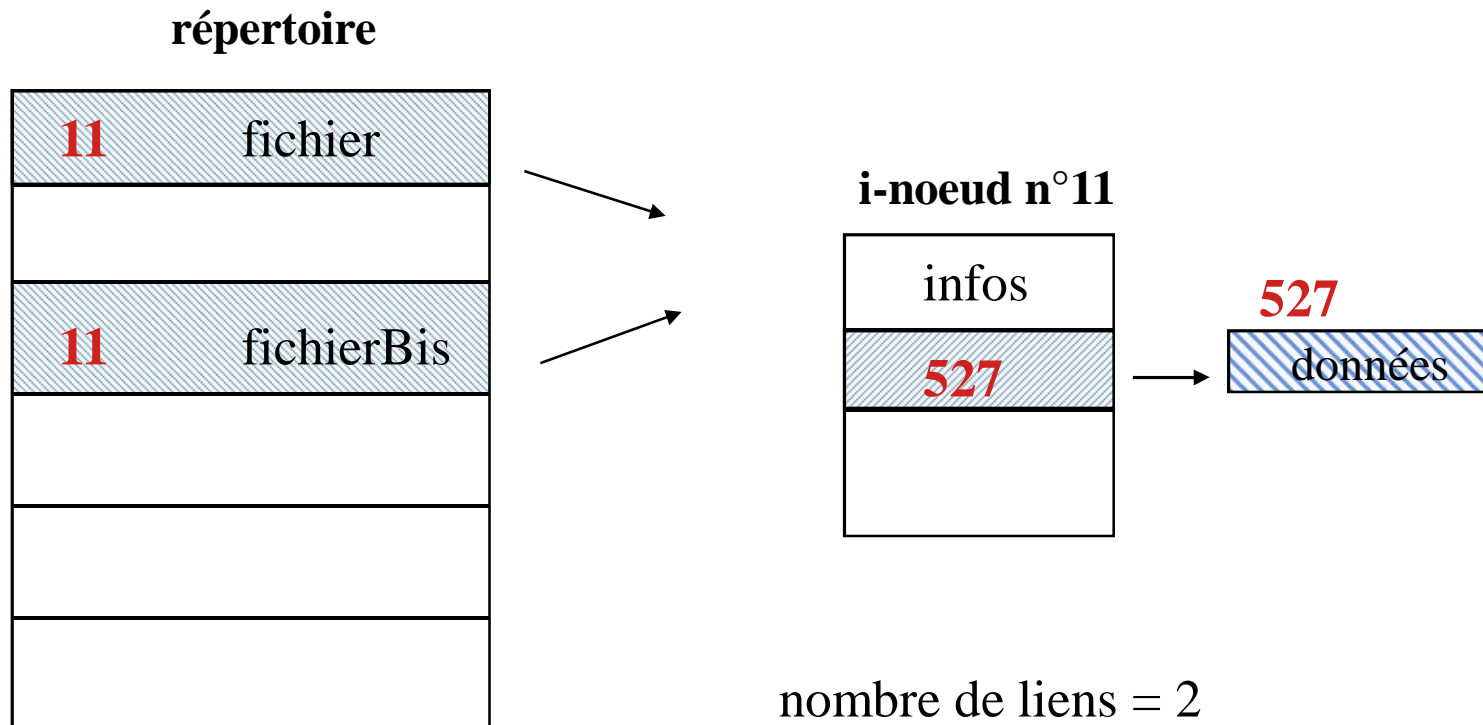
% ln fichier fichierBis

ou

#include <unistd.h>

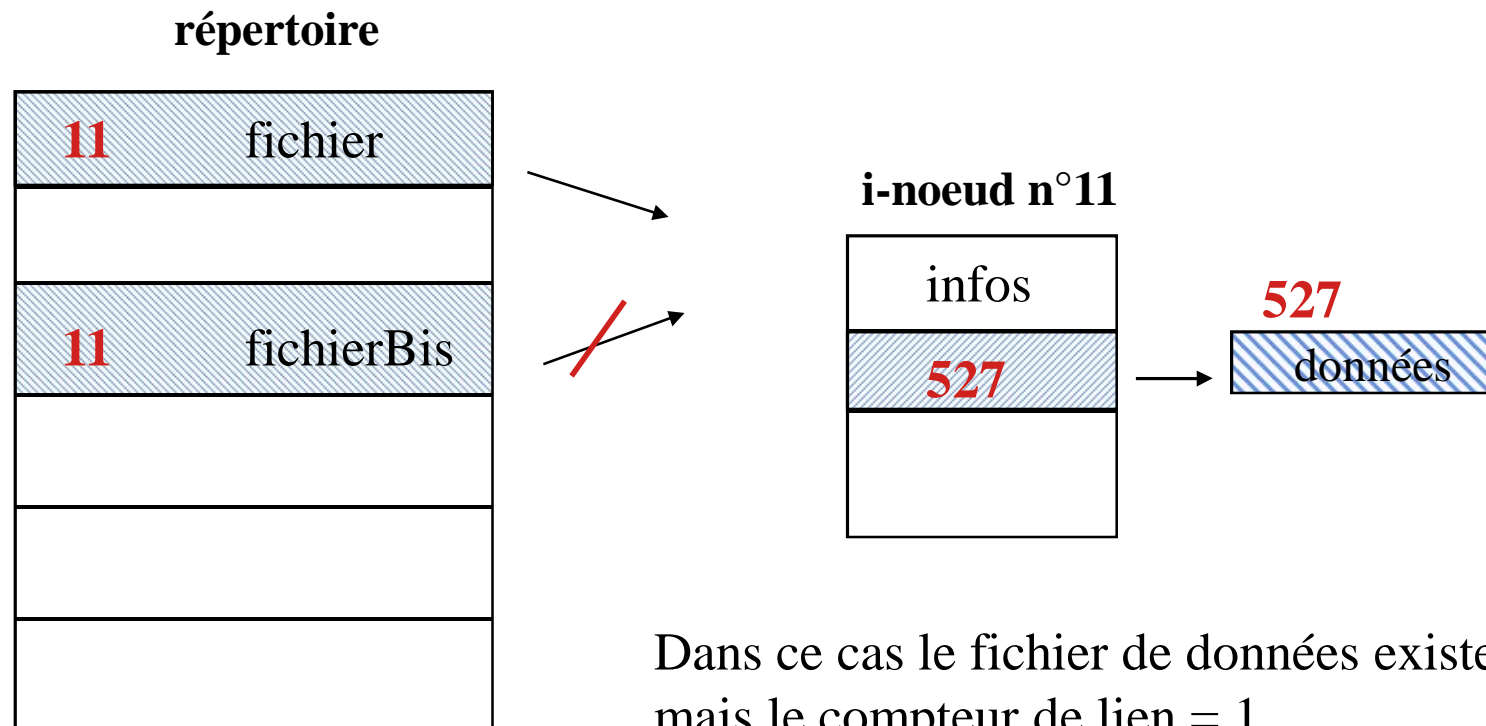
link (const char *fichier, const char * fichierBis)

- i-noeud indique 2 liens: fichier et fichierBis
- un seul fichier de données

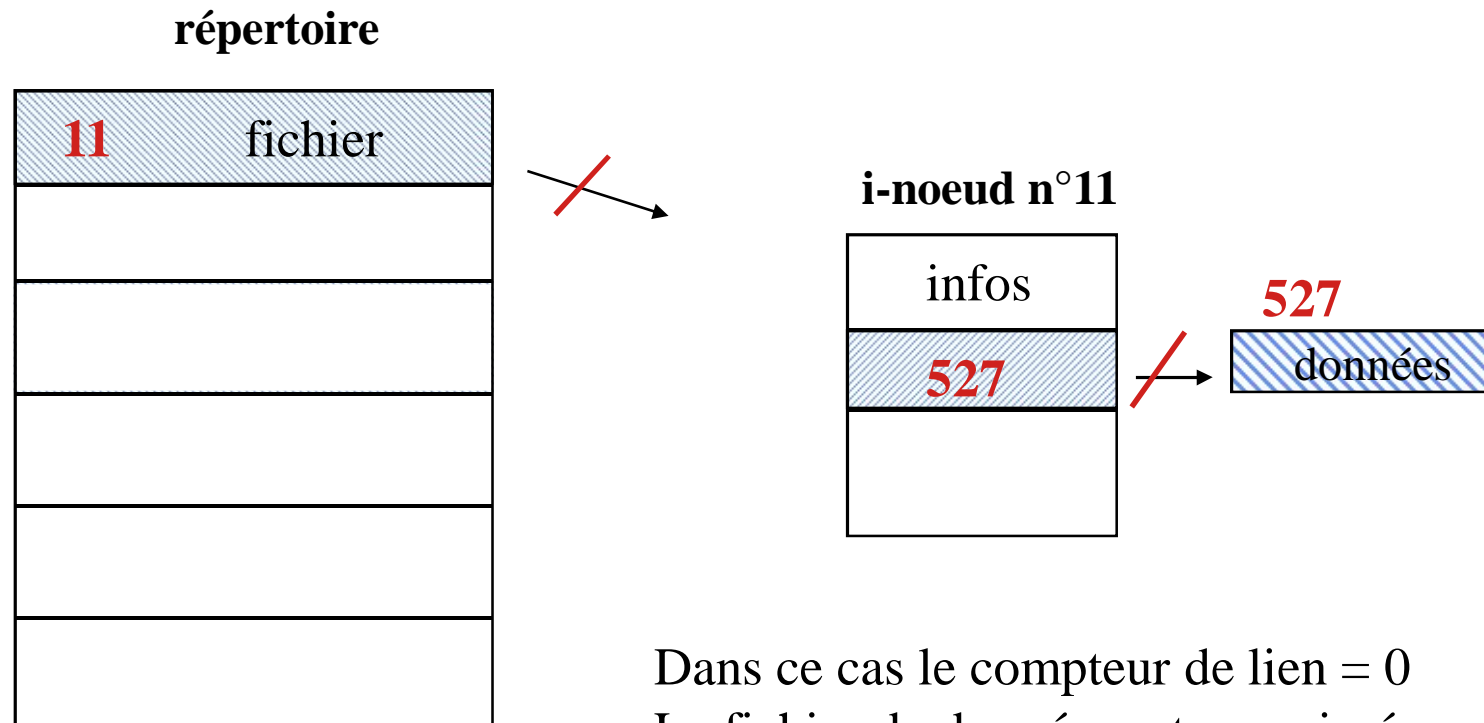


– Supprimer un lien synonyme

unlink (const char *fichierBis)



Dans ce cas le fichier de données existe toujours
mais le compteur de lien = 1



Dans ce cas le compteur de lien = 0
Le fichier de données est supprimé
Si fichier ouvert: attente fermeture
L'i-noeud devient i-noeud libre

– Cas des fichiers répertoires

- Même mécanisme
- Commandes: mkdir et rmdir
- Un fichier répertoire a au moins deux liens

2122 -rw-rw-rw- 1 durand groupa 2134 jun 21 09:21 fichier

2123 **d**rw-rw-rw- 3 durand groupa 512 jun 21 09:21 **usr**

Deux liens ou plus si un ou plusieurs autres fichiers sous-répertoire

– Lien symbolique

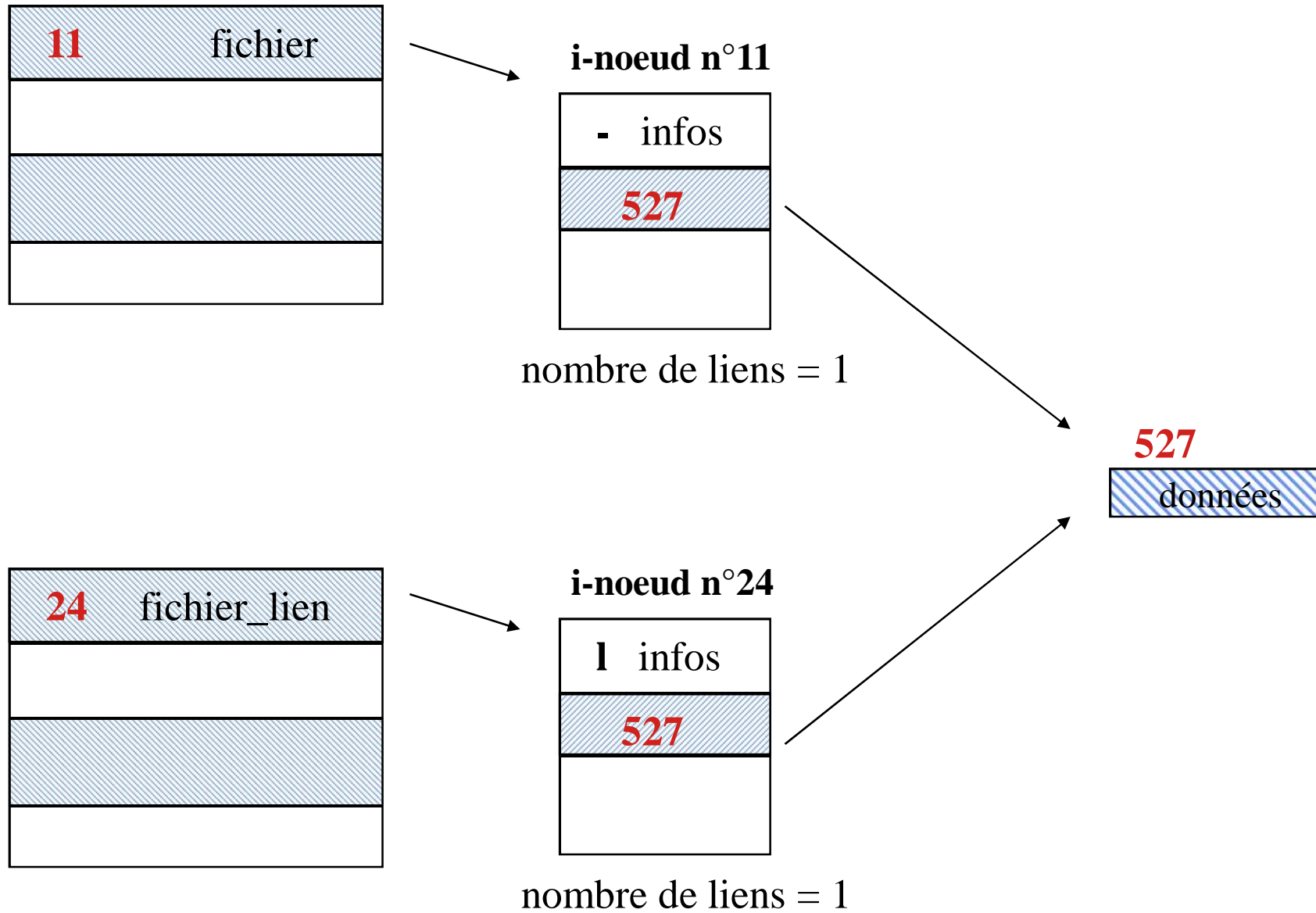
- nouveau fichier de type lien: l
- i-noeud crée
- le lien permet à des utilisateurs différents de partager un même fichier de données

% ln -s fichier lien_fichier

- le fichier lien_fichier apparait lors de la commande ls
- le nombre de lien est inchangé

```
2122 -rw-rw-rw- 1 durand  groupa 2134 jun 21 09:21 fichier
```

```
2123 lrw-rw-rw- 1 durand  groupa  512 jun 21 09:21 lien_fichier -> fichier
```

- **Copie**

- copier un fichier
 - copie physique
 - nouvel i-noeud
 - nouveau nom

% cp fichier1 fichier2

```
2122 -rw-rw-rw- 1 durand groupa 2134 jun 21 09:21 fichier1
2123 -rw-rw-rw- 1 durand groupa  512 jun 21 09:21 fichier2
```

fichier1 et fichier2 ont chacun un seul lien

- **Renommer un fichier**

- Changer de nom de lien

#include <unistd.h>

int rename(const char *ref_src, const char *ref_dest)

Commande: **mv fic_src fic_dest**

- **Les attributs d'un fichier**

```
#include <unistd.h>
```

```
int access (char *ref, int acces)
```

Commande: **ls** -la

- **Créer un i-noeud**

```
#include <sys/stat.h>
```

```
int mknod(const char *ref, mode_t mode, dev_t ressource)
```

Dans le cas d'un fichier ordinaire:

- **open**() ou creat()

Dans le cas d'un tube:

- **pipe**()

Dans le cas d'un répertoire:

- commande **mkdir**

- **Changer les droits d'accès**

#include <sys/stat.h>

int **chmod**(const char *ref, mode_t mode)

Commande: **chmod** mode fichier

- **Changer de groupe**

Commande: **chgrp** nv_grp fichier

- **Changer de propriétaire**

#include <unistd.h>

int **chown**(const char *ref, uid_t id_util, gid_t id_grp)

Commande: **chown** nv_prop fichier

- **Ouverture de fichier**

```
#include <sys/types.h>
```

```
#include <sys/stat.h>
```

```
#include <fcntl.h>
```

```
int open(const char *ref, int mode_ouv, [mode_t mode] )
```

ou

```
int creat (const char *path, mode_t mode );
```

- **Fermeture de fichier**

```
#include <unistd.h>
```

```
int close( int desc )
```

- **Lecture d'un fichier**

```
#include <unistd.h>
```

```
ssize_t read( int desc, void *ptr_buf, size_t nb_octets )
```

- **Ecriture de fichier**

```
#include <unistd.h>
```

```
ssize_t write( int desc, void *ptr_buf, size_t nb_octets )
```


BIBLIOTHEQUE E/S

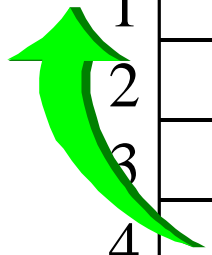
- Appels système: read(), write(), open(), close()
 - accès par les descripteurs
 - définition des zones de réception/émission
 - nb d 'octets transmis
 - pas de formatage des E/S, pas de conversion
- Appels fonctions en bibliothèque stdio.h
 - gestion tampon utilisateur
 - réduction du temps UC /appel système
 - pas de changement de contexte
 - pas de changement de processus

- Principe du tampon
 - en lecture:
 - tampon se remplit par un `read()`
 - les caractères sont récupérés dans ce tampon jusqu'à épuisement
 - en écriture
 - les caractères remplissent le tampon
 - le tampon plein est vidé par un `write()`
 - Mémoire cache avant écriture disque
- Structure `FILE` dans `stdio.h` pour la gestion du tampon
 - pointeurs du tampon du fichier
 - n° du descripteur

- Bibliothèque stdio.h
 - fopen()
 - fread()
 - fwrite()
 - fclose()
 -

Utilisation de dup() pour rediriger la sortie standard

descripteurs du processus A



0	STDIN
1	STDOUT
2	STDERR
3	tube input
4	tube output
5	
6	

DUP()

1) on crée un tube:

- deux descripteurs: 3 et 4 qui pointent sur la table des fichiers: ici tube

2) on ferme le descripteur 1

- l'entrée 1 est libre

3) on duplique le descripteur 4 avec retour = dup (4)

- le descripteur 4 est recopié dans le descripteur 1 (dup prend la première entrée libre)

- valeur de retour: le nouveau descripteur ici le 1

4) on ferme les descripteurs 3 et 4 qui ne servent plus

5) tout envoi vers le descripteur 1 concernera le tube