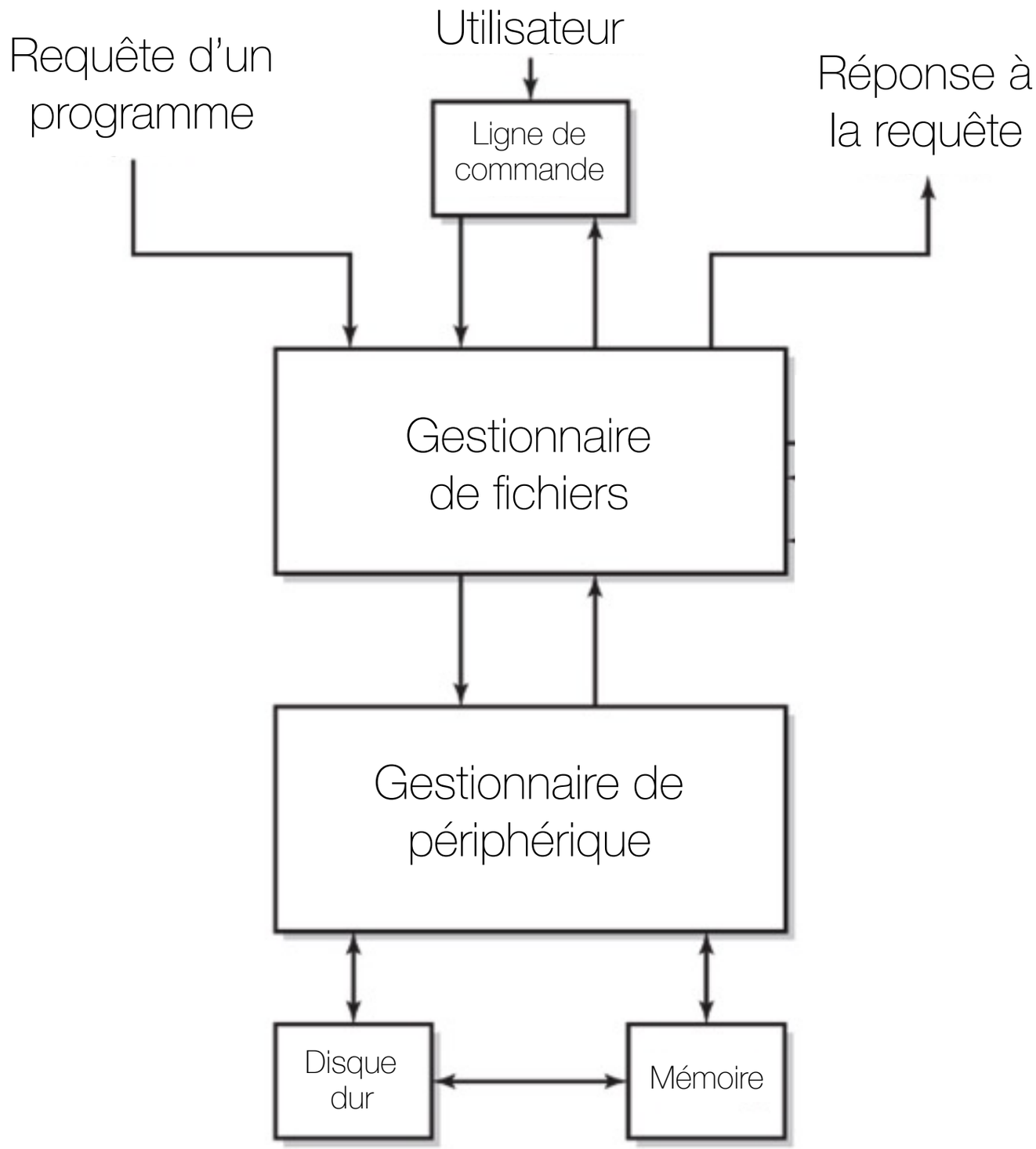
Gestion des fichiers

GIF-1001 Ordinateurs: Structure et Applications, Hiver 2016

Jean-François Lalonde

Source: http://www.business-opportunities.biz/2014/11/16/40-home-offices-that-are-messier-than-yours/

Aujourd'hui: gestionnaire de fichiers



Source: Englander, ch. 17

Système de fichiers

- Tous les systèmes d'exploitation offrent des systèmes de fichiers.
- Un système de fichier est un ensemble de fonctions du système d'exploitation permettant:
  - d'entreposer des blocs de données (des fichiers) sur un ou plusieurs périphériques.
  - à l'utilisateur d'organiser ses informations et d'utiliser ses périphériques d'entreposage simplement.
  - au programmeur de sauvegarder et de charger des données rapidement sans se préoccuper:
    - de la nature du périphérique
    - du format et de l'organisation logique du périphérique.

Bloc de données et “cluster”

- Dans un disque dur, les données sont disposées par bloc
- Idéalement, un système de fichier saurait ce qui emmagasiné dans chaque bloc de mémoire.
  - Les blocs sont petits par rapport à l’espace de mémoire requis pour donner l’information sur le contenu du bloc.
- Les données sont donc regroupées par clusters sur un disque dur, c’est-à-dire par groupes de blocs (cluster = unité d’allocation en français). Tous les clusters ont la même taille.
- Quelle est la taille minimum d’un cluster?
  - La taille minimum d’un cluster est de 1 bloc de données.
  - Sa taille sera toujours un multiple de la taille d’un bloc de donnée.

Compromis—taille des clusters

- Un cluster ne sert qu'à un seul fichier.
- Plus les clusters sont gros, plus il y aura d'espace disque gaspillé pour entreposer des fichiers.
  - Si les clusters ont 512o, combien d'octets seront gaspillés pour un fichier de 1000o?
    - 24 octets seront gaspillés
  - Si les clusters ont 2048o, combien d'octets seront gaspillés pour ce même fichier de 1000o?
    - 1048 octets seront gaspillés.
- Plus les clusters sont petits, plus le système d'exploitation devra en gérer.
  - Il faut plus d'espace disque pour maintenir de l'information sur 1024 clusters de 512o que sur 256 clusters de 2 Ko.
  - Aussi, un fichier réparti sur 6 petits clusters non contigus sera probablement plus long à lire qu'un fichier dans 1 seul gros cluster.

Allocation du disque

- Problème similaire à l'allocation mémoire
- Différents types d'allocation:
  - contigüe
  - chaînée
  - indexée

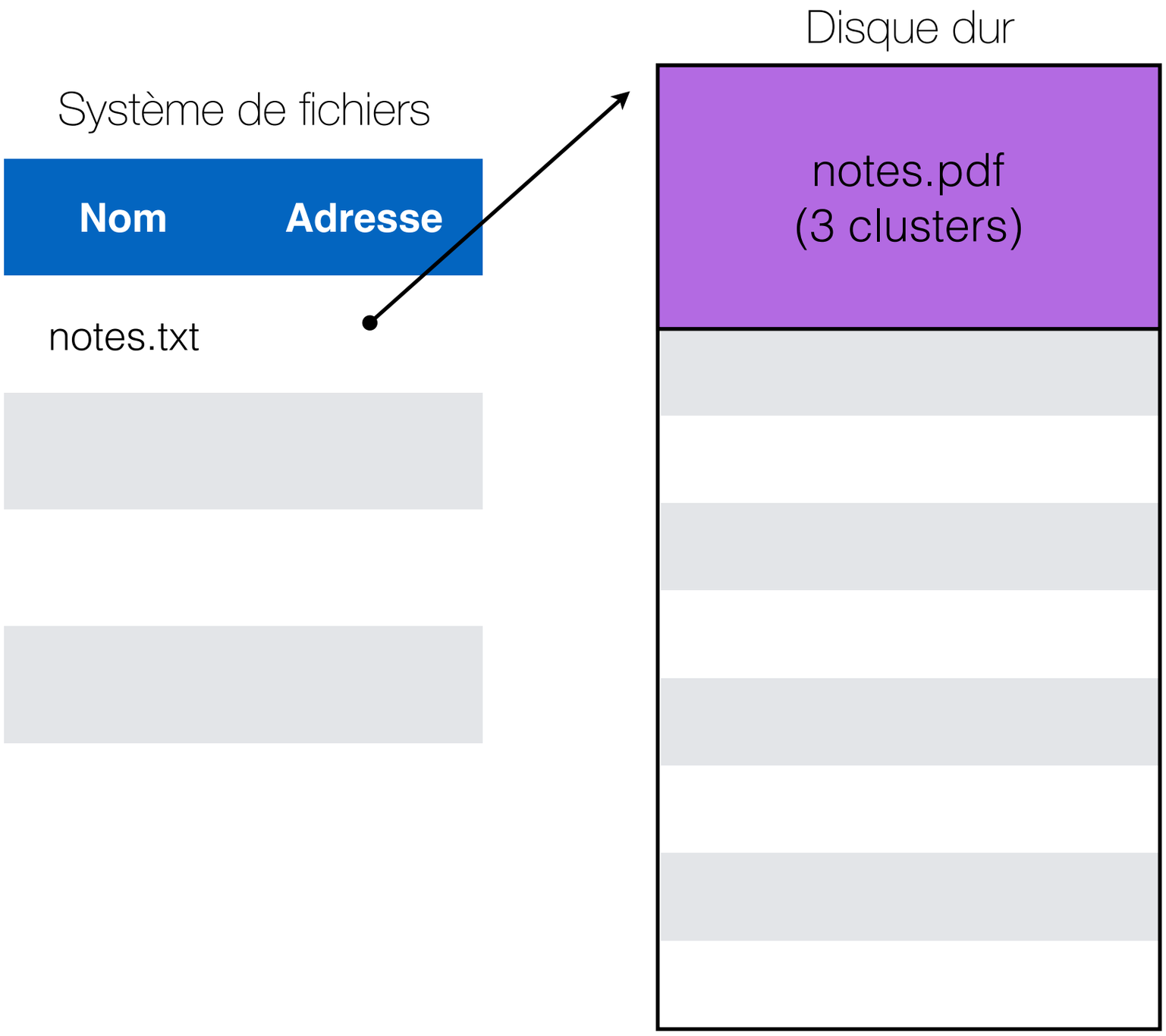
Allocation contigüe

Système de fichiers

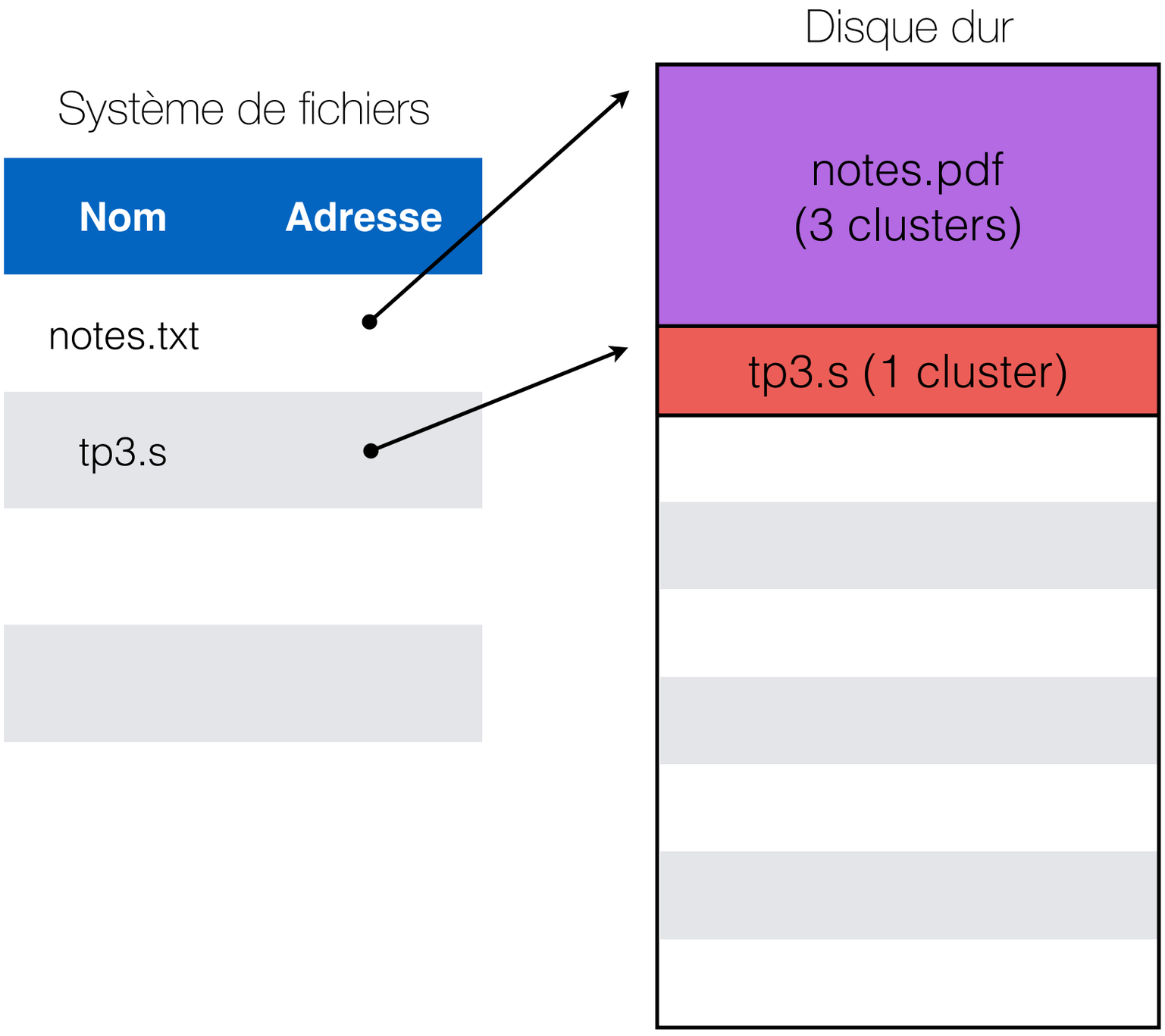
Nom	Adresse

Disque dur

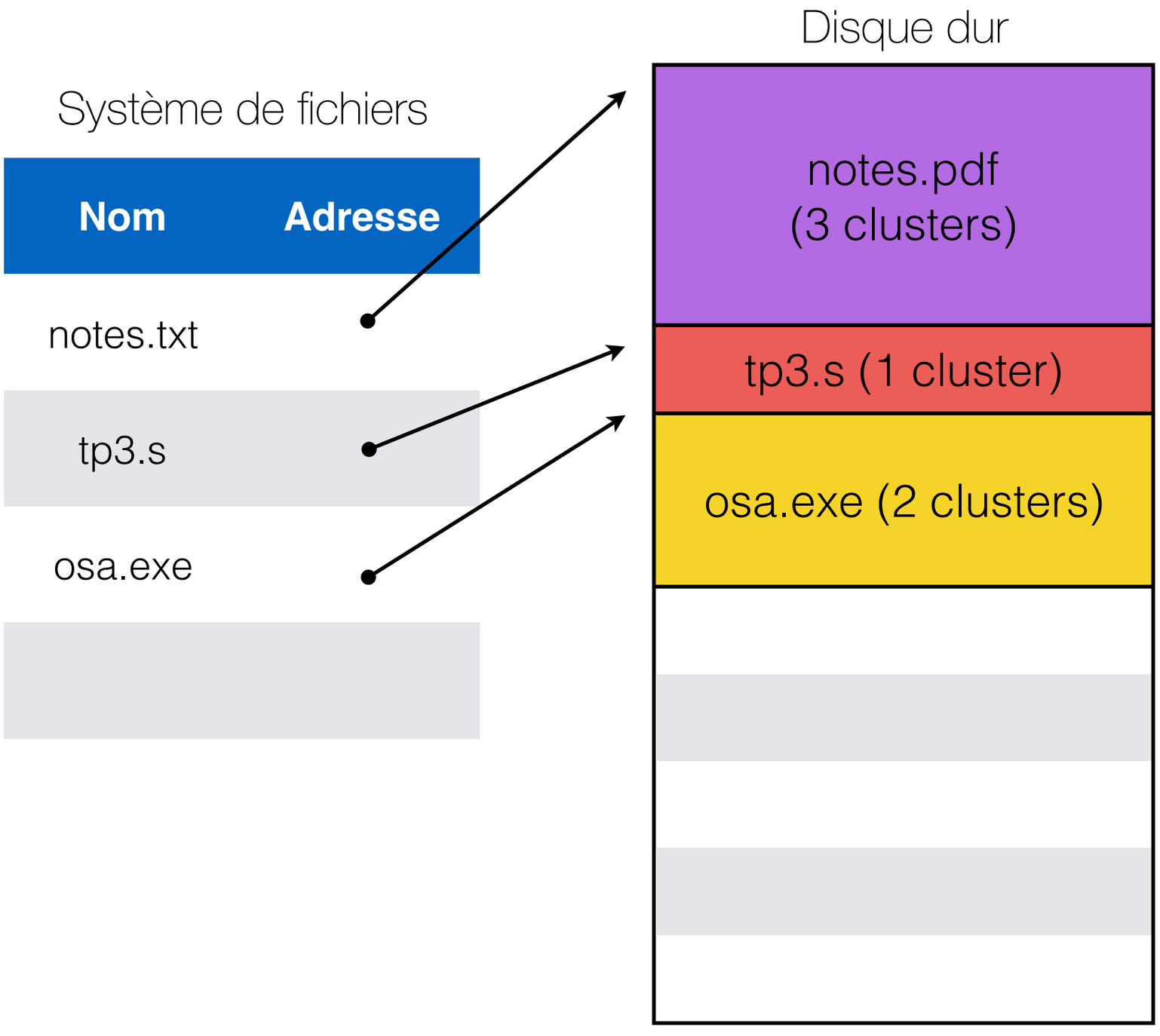
Allocation contigüe



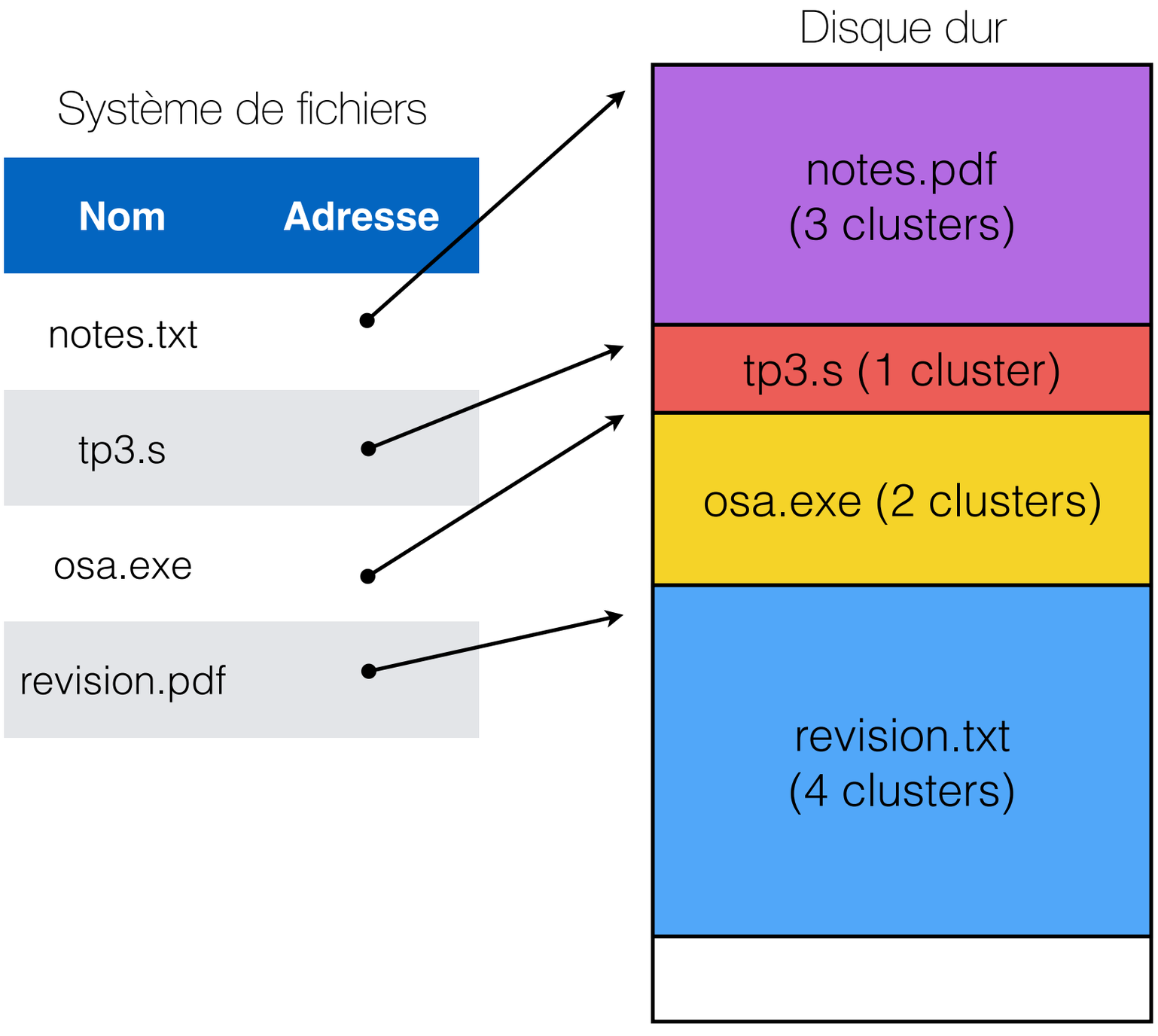
Allocation contigüe



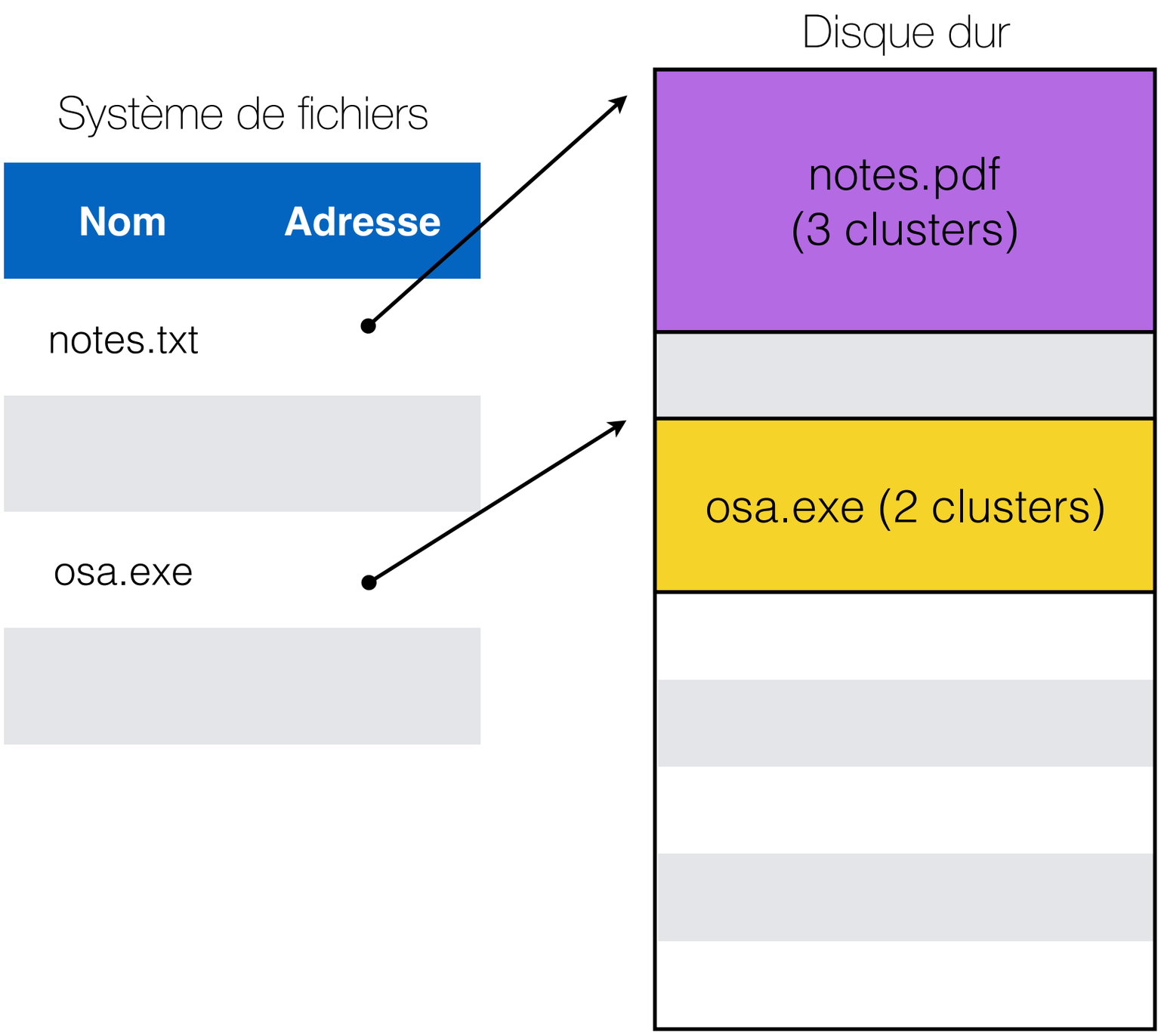
Allocation contigüe



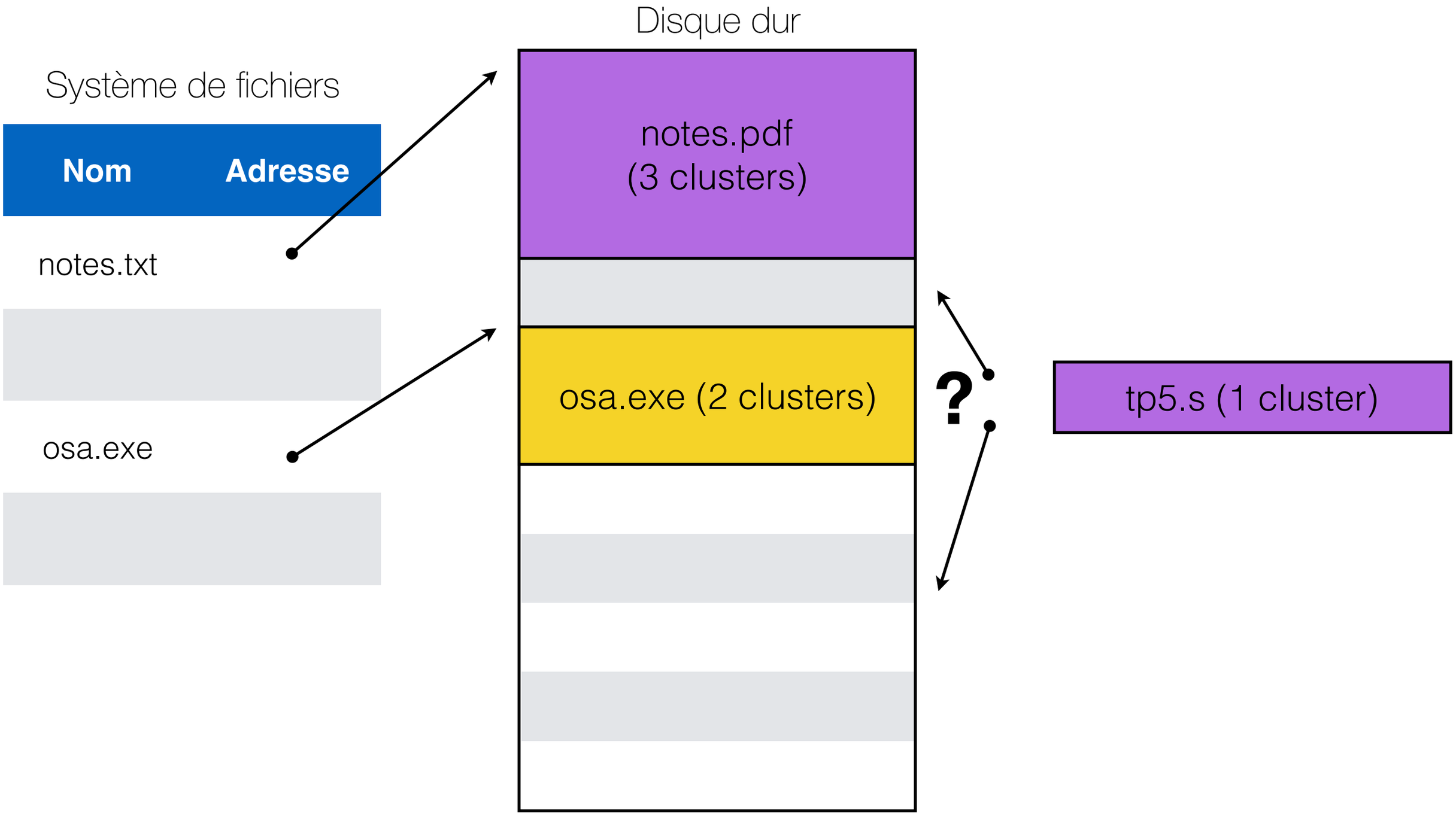
Allocation contigüe



Allocation contigüe



Allocation contigüe



Allocation contigüe

- La façon la plus simple de mettre un fichier sur un disque est de mettre toutes les données de ce fichier dans un partie unique du disque.

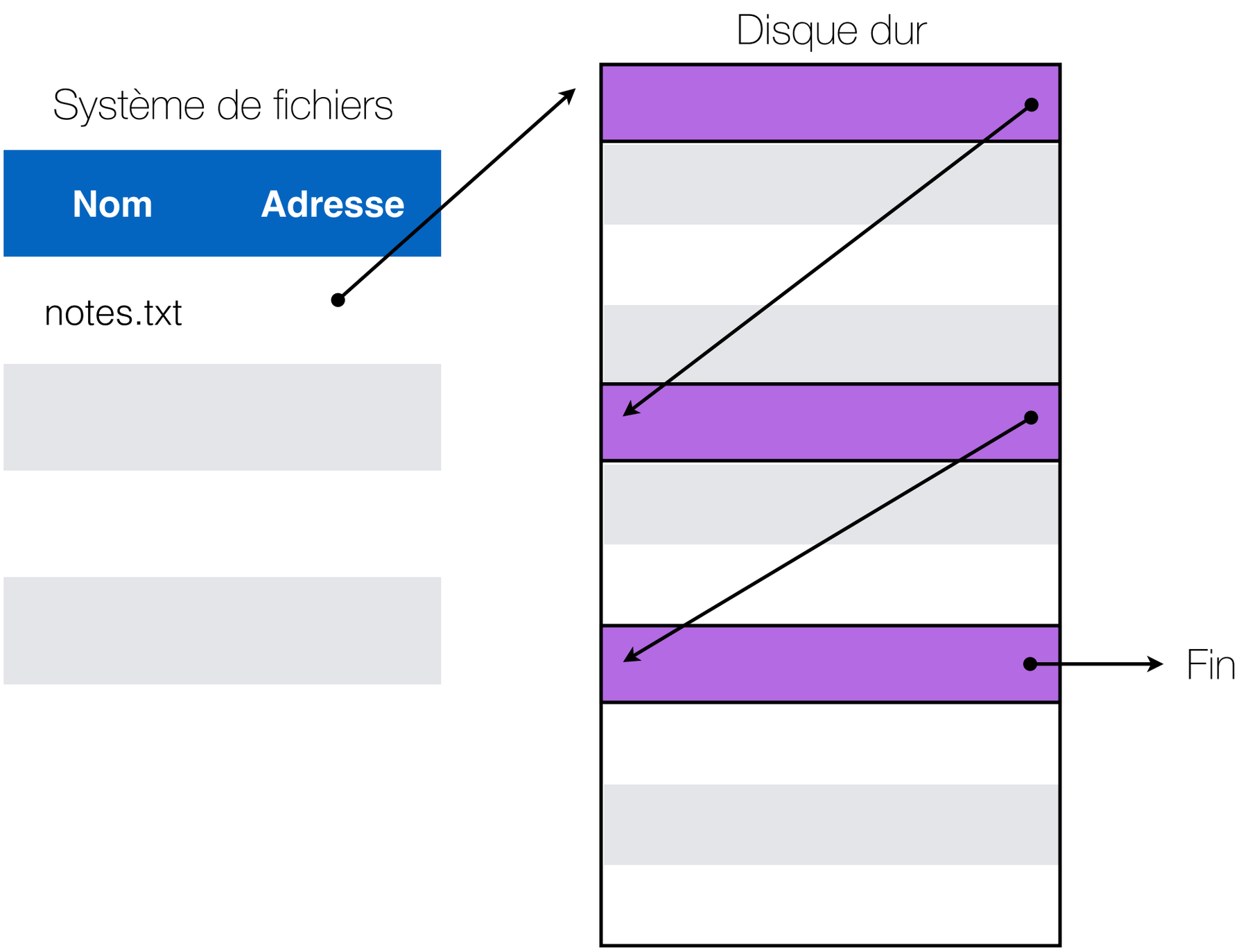
- Il existe plusieurs algorithmes utilisés pour mettre un fichier sur le disque. Dans tous les cas, il faut trouver une espace suffisamment grand pour le fichier et pour sa croissance anticipée. Comme pour l'allocation contigüe de mémoire pour les processus, on retrouve les algorithmes de "first-fit", "best-fit" et "worse-fit".

- L'allocation contigüe implique l'existence d'une table qui donne l'emplacement de départ de chaque fichier sur le disque.

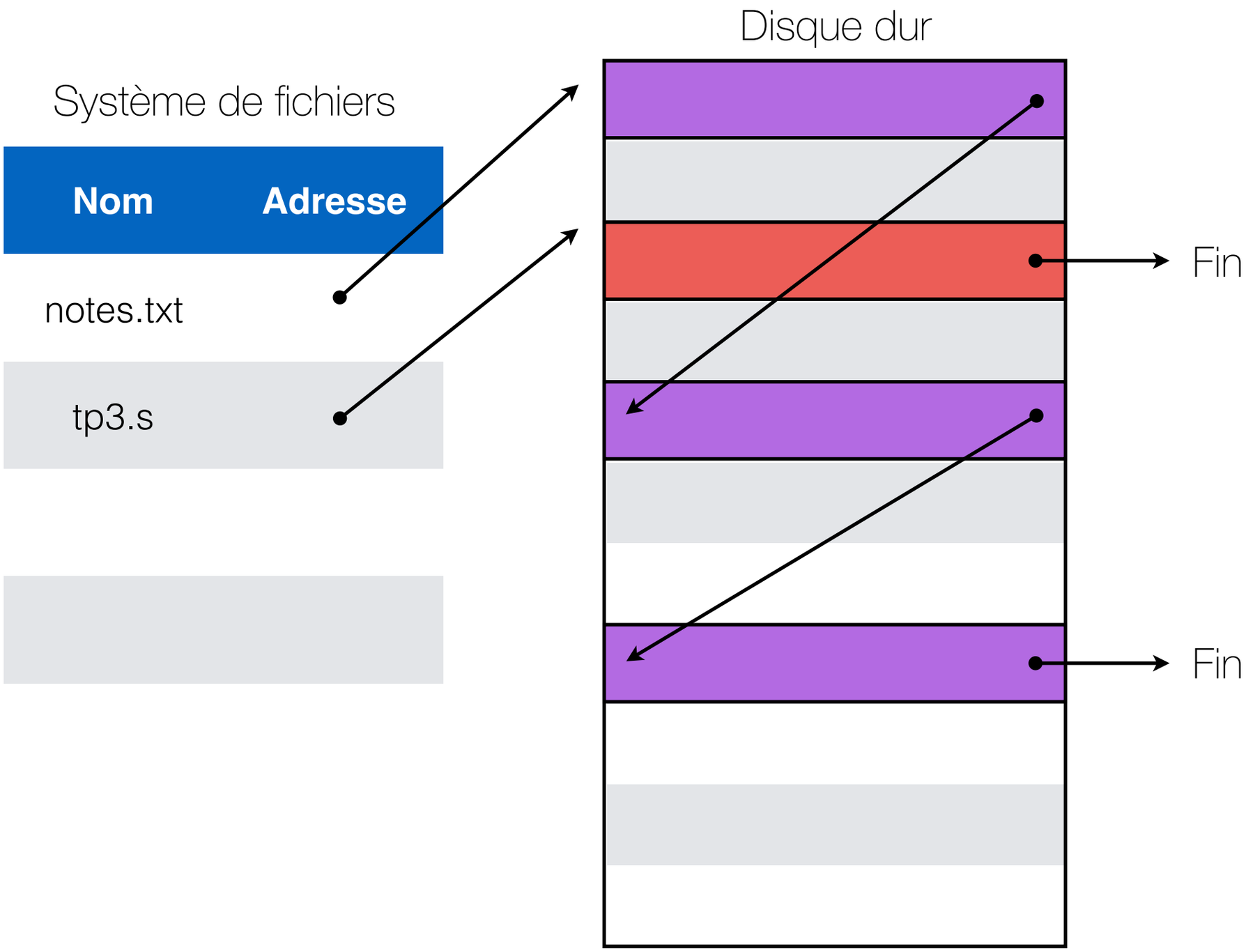
Allocation contigüe

- Avantages?
  - Accéder au fichier est très simple
- Inconvénients?
  - Si un fichier grandit trop, il est possible qu'il faille le re-localiser dans un espace plus grand.
  - À la longue, le disque devient fragmenté

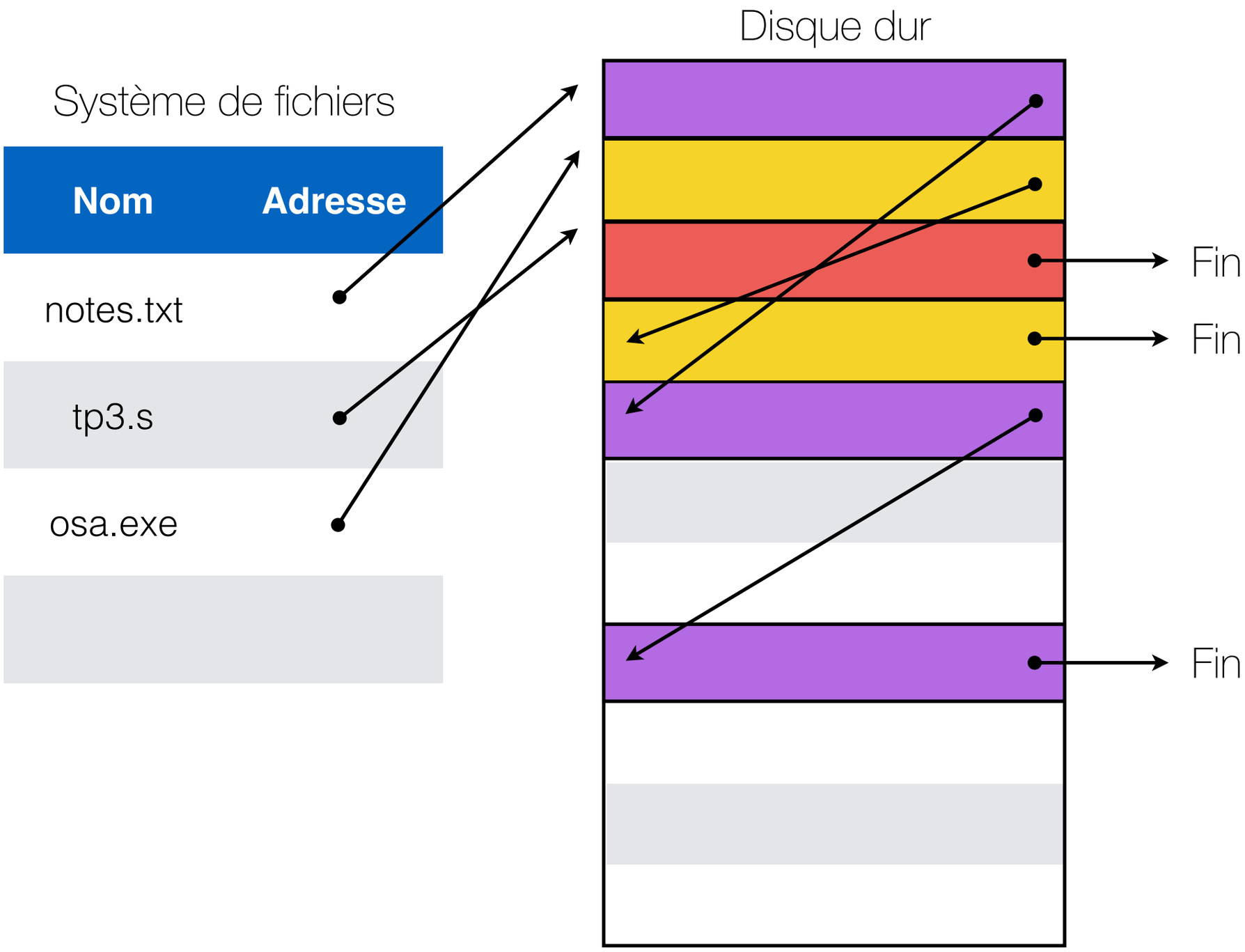
Allocation chaînée



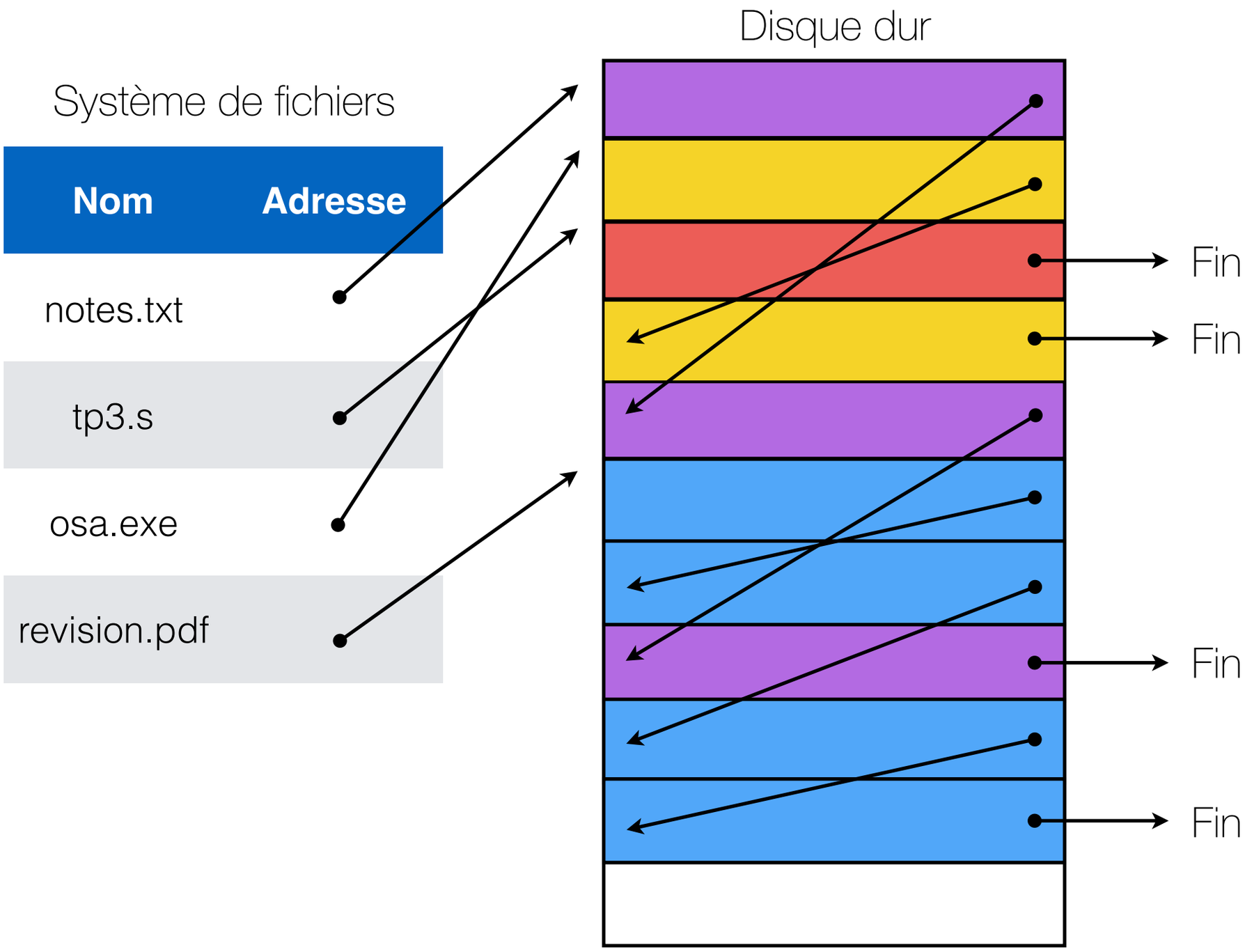
Allocation chaînée



Allocation chaînée



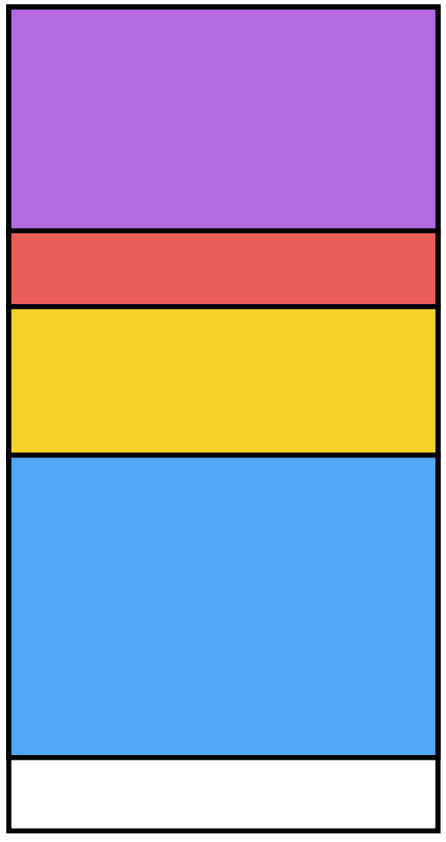
Allocation chaînée



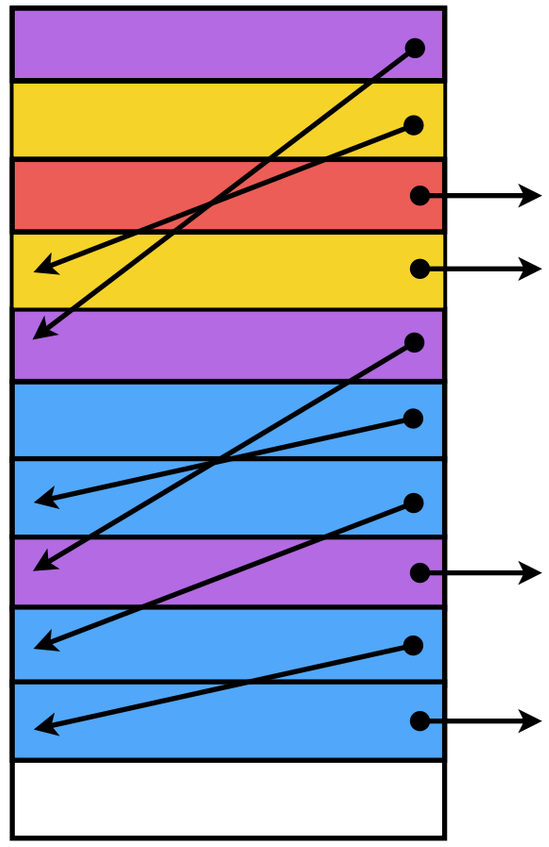
Allocation chaînée

- Les données ne sont pas contigües en mémoire
- Chaque cluster contient un lien vers le cluster suivant dans le fichier

Allocation contigüe



Allocation chaînée



Allocation chaînée

- Avantages?
  - Pas de fragmentation
  - Modifier la taille d'un fichier est facile
- Inconvénients?
  - Il faut "chercher" le morceau du fichier qui nous intéresse
  - Il faut stocker des pointeurs additionnels

Allocation chaînée avec table d'allocation (FAT)

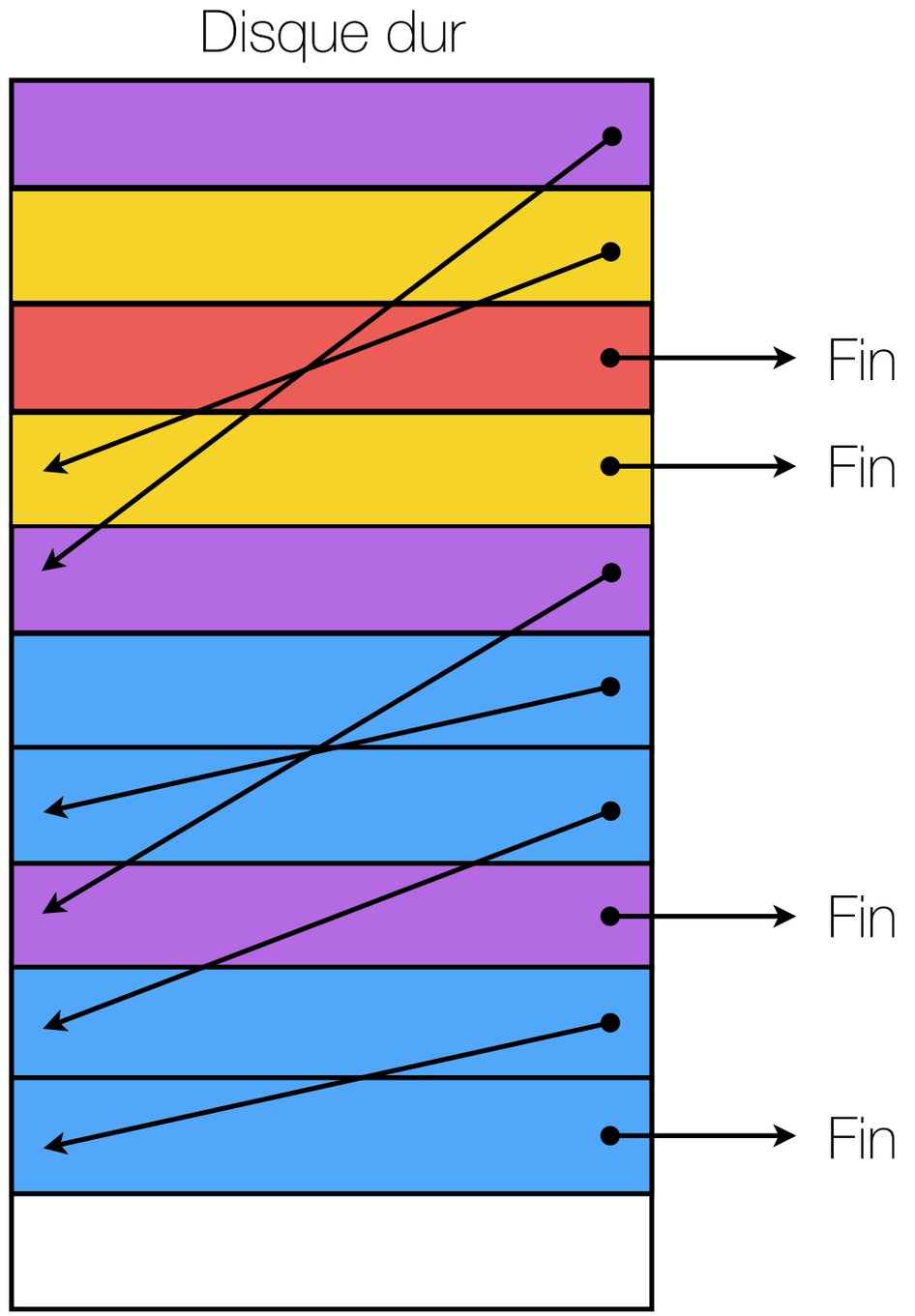
- Comme l'allocation chaînée, mais les liens sont stockés dans une table séparée
- Le système de fichier "FAT" ("File Allocation Table") contient deux tables:
  - La **table de répertoire** contient le nom et le chemin de tous les fichiers. Pour chaque fichier de la table, le premier cluster du fichier est identifié.
  - La **table d'allocation** (FAT), décrit l'utilisation de chaque cluster.
- Dans la FAT:
  - un fichier est identifié sous forme d'une chaîne de cluster: chaque cluster pointe sur le cluster suivant du fichier.
  - un caractère spécial indique qu'un cluster n'a jamais été utilisé (ex: '0').
  - un autre caractère spécial indique la fin d'un fichier (ex: '-1').

Allocation chaînée avec table d'allocation (FAT)

Système de fichiers

Nom	1er cluster
notes.txt	
tp3.s	
osa.exe	
revision.pdf	

Cluster	Cluster suivant
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	



Allocation chaînée avec table d'allocation (FAT)

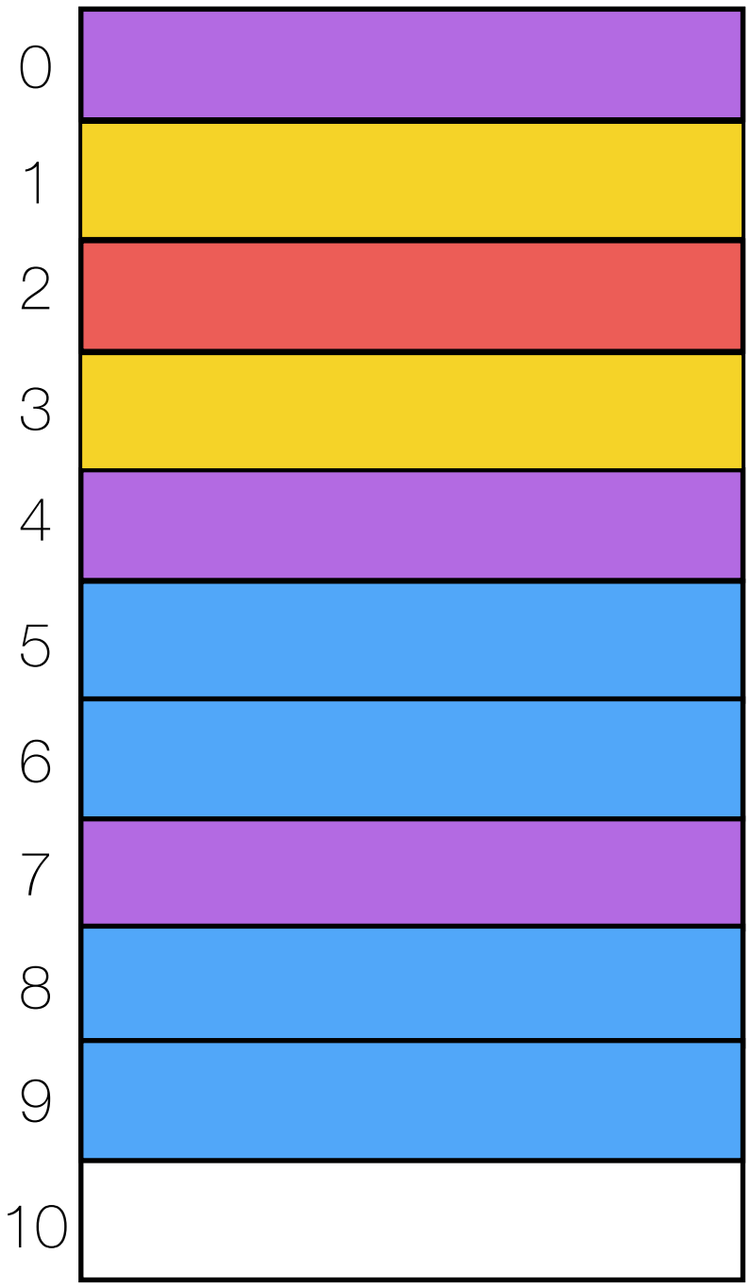
Table de répertoire

Nom	1er cluster
notes.txt	0
tp3.s	2
osa.exe	1
revision.pdf	5

Table d'allocation (FAT)

Cluster	Cluster suivant
0	4
1	3
2	-1
3	-1
4	7
5	6
6	8
7	-1
8	9
9	-1
10	-1

Disque dur



Exercice #1

Table de répertoire

Nom	1er cluster
osa.txt	10

Table d'allocation (FAT)

cluster	Cluster suivant
0	-1
1	4
2	1
3	0
4	-1
5	7
6	0
7	2
8	5
9	0
10	8

Disque dur

cluster	Données
0	
1	'OS
2	S D
3	
4	A!\0
5	E C
6	
7	OUR
8	E L
9	
10	VIV

Exercice #2

Table de répertoire

Nom	1er cluster
venus.txt	1
mars.txt	5
terre.txt	8

Table d'allocation (FAT)

cluster	Cluster suivant
0	-2
1	2
2	12
3	-1
4	-2
5	6
6	7
7	-1
8	9
9	10
10	3
11	-2
12	-1
13	-2

Disque dur

cluster	Données
0	
1	fe
2	mm
3	s
4	
5	ho
6	mm
7	es
8	en
9	fa
10	nt
11	
12	es
13	

Problèmes du système FAT (DOS)

- Tout d'abord, la FAT occupe assez d'espace.
  - Si un disque de 256Mo (2^{28}o) a des clusters de 4Ko (2^{12}o). Si la FAT contient le minimum d'information, quelle est la taille de la FAT?
    - Le disque a $2^{28} / 2^{12} = 2^{(28-12)} = 2^{16}$ clusters
    - Le minimum d'information est de 16 bits (2 octets) par cluster (il faut 16 bits pour identifier le cluster suivant du fichier)
    - La FAT occupe donc $2^{16} * 2o = 128Ko$ d'espace mémoire.
- Le FAT ne gère pas la sécurité des fichiers.
- Les noms des fichiers sont en ASCII. Les japonais doivent connaître l'anglais pour utiliser le DOS.

FAT et fautes de disque

- Le FAT ne se protège pas des fautes de disque.
- Par exemple, si le système plante pendant une écriture de la table d'allocation (perte d'alimentation ou autre), la FAT peut devenir corrompue. Un fichier peut être partiellement ou totalement perdu.
- Deux scénarios principaux peuvent survenir lors de la corruption de FAT:
  - **Fichier croisé**: un fichier dont la FAT a été accidentellement modifiée de telle sorte que la chaîne de cluster d'un fichier soit altérée. Le cluster d'un fichier (ex: "abc.txt") pointe sur le cluster d'un autre fichier (ex: "def.txt") de telle sorte que la fin de "abc.txt" est perdue. Lorsque "abc.txt" est lu, la fin du fichier est celle de "def.txt".
  - **Bloc perdu**: un cluster qui n'est pas alloué à un fichier, mais qui n'est pas libre non plus. Par exemple, la fin de "abc.txt" dans l'exemple précédent serait perdue, car plus aucun cluster ne pointe dessus. De plus, ces clusters ne sont pas libre, car ils ont été attribués à "abc.txt".

FAT16 versus FAT32

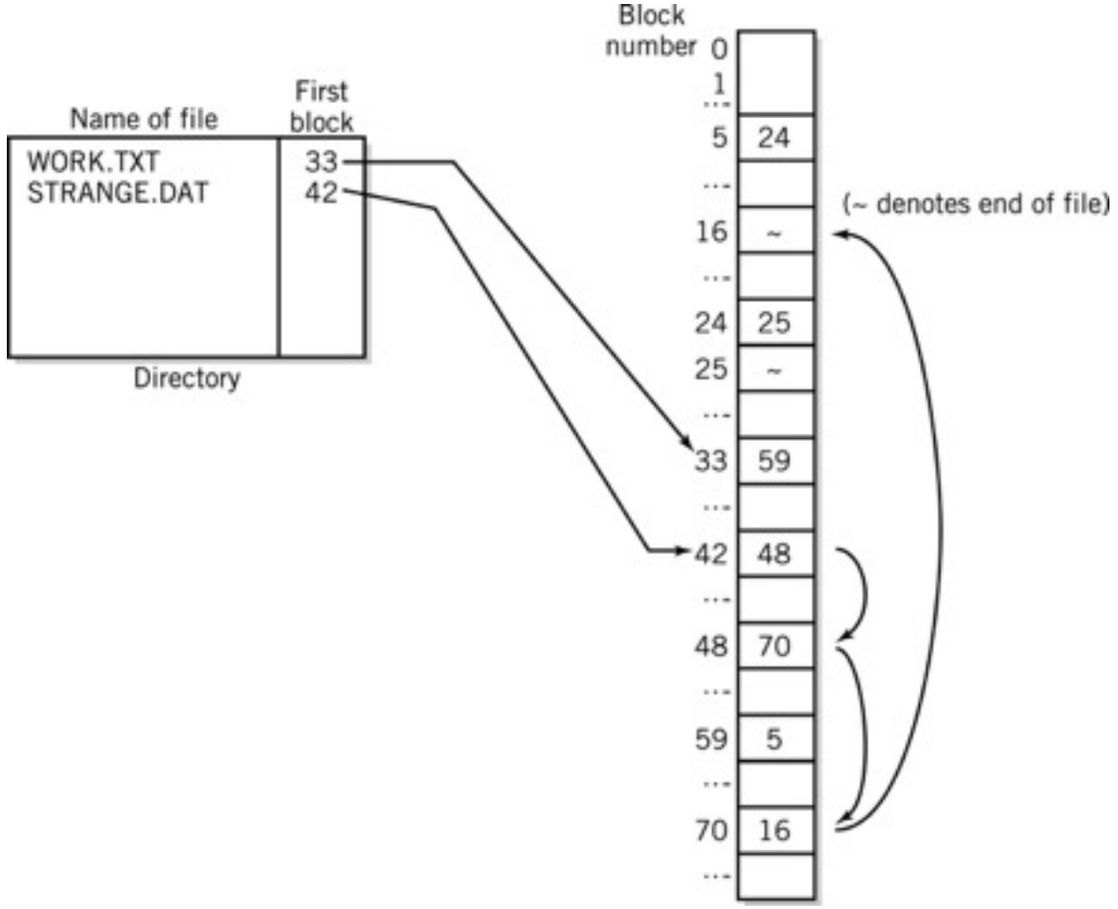
- Le FAT16 peut adresser jusqu'à 2^{16} clusters, soit 65536. Cette quantité de clusters est très petite par rapport à un disque dur moderne.
  - Par exemple, un disque dur de 16Go (2^{34} octets) devra avoir des clusters de
    - $2^{34} / 2^{16} = 2^{(34-16)} = 2^{18}$ = 256Ko.
  - Avec des clusters de cette taille, énormément de mémoire disque est gaspillée. Pourquoi?
    - Il faut que tous les fichiers aient au moins cette taille (ou un multiple de cette taille)

- Le FAT32 a été créé dans le but d'étendre les capacités du FAT16. Le FAT32 peut adresser 2^{32} clusters, soit ~4.3 milliards de clusters.
  - Les clusters du FAT32 peuvent être plus petits que ceux du FAT16 pour une taille de disque dur donnée.
  - En contrepartie, la FAT d'un système FAT32 sera plus grosse. Combien faut-il d'octets pour indiquer le cluster dans FAT16 et FAT32?
    - en FAT16, il nous faut 2 octets (16 bits)
    - en FAT32, il nous faut 4 octets (32 bits)

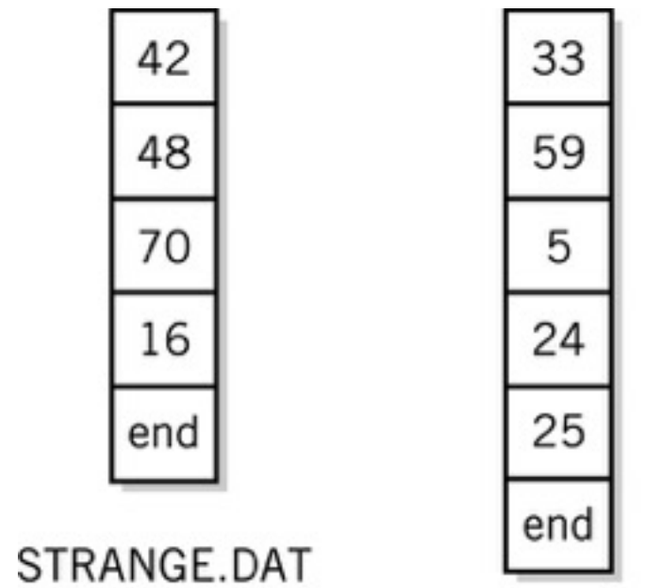
Allocation indexée

- Allocation non-contigüe
- Au lieu d'utiliser une seule table (la FAT) pour tous les fichiers, on utilise une table *par* fichier (appelée la table d'index)

Allocation chaînée



Allocation indexée



source: Englander, ch. 17

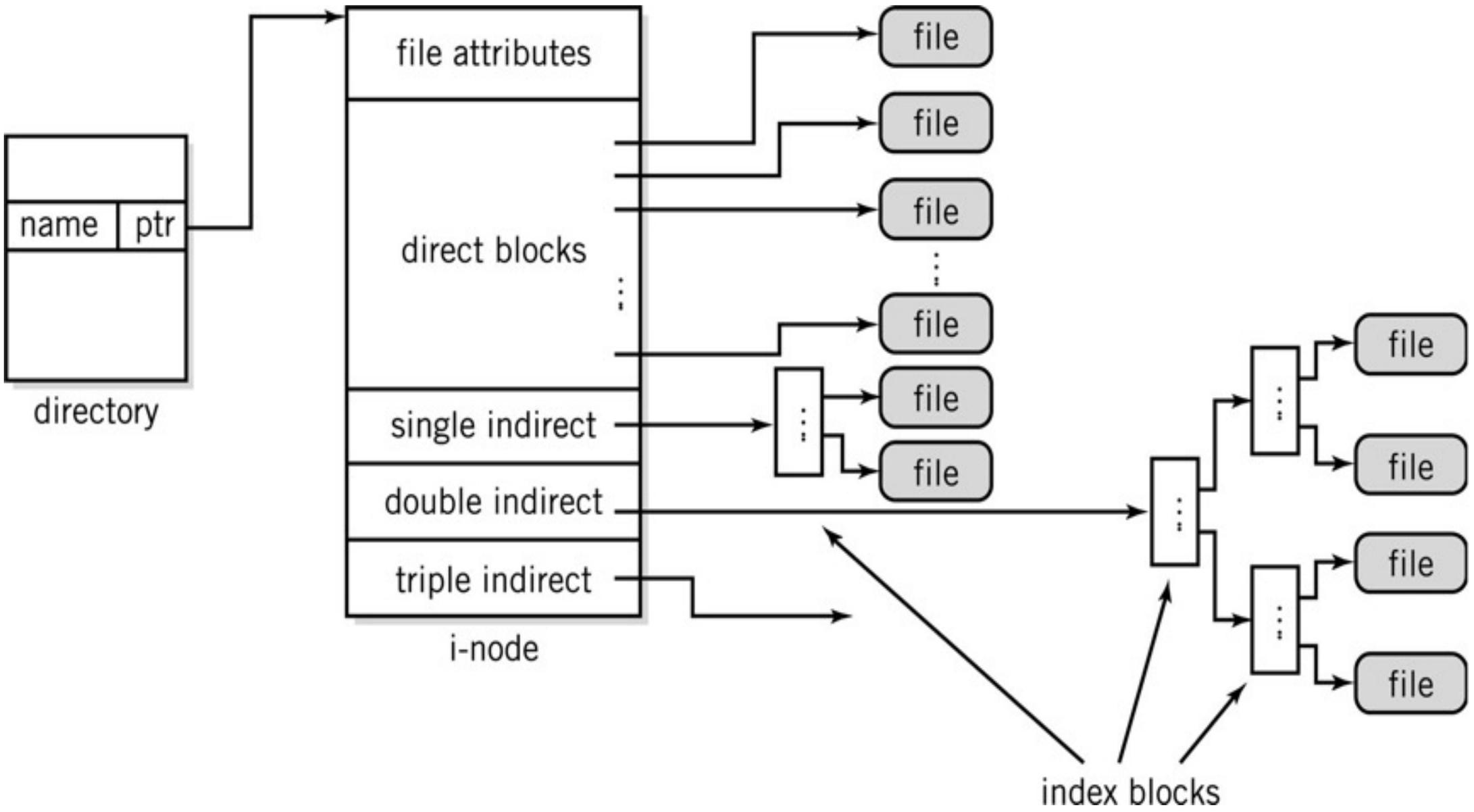
Allocation indexée

- Avantages?
  - Pas de fragmentation
  - Peut accéder à n'importe quelle section d'un fichier facilement
  - Meilleure utilisation de la mémoire: seules les tables d'index des fichiers ouverts sont chargés en mémoire.
    - (en FAT, la table doit entière doit être chargée en mémoire)
- Inconvénients?
  - Il faut charger la table d'index

“Inodes”

- Un “inode” (“index node”) est une structure donnant l’emplacement d’un fichier contenant 10 pointeurs (10*4bytes) sur des clusters du disque.
- En plus des 10 pointeurs, chaque inode contient un pointeur sur un autre inode permettant au fichier d’utiliser 10 clusters additionnels au besoin.
- Pour des fichiers plus gros encore, chaque inode contient un pointeur sur un autre inode contenant 10 pointeurs sur d’autres inode permettant au fichier d’utiliser 10 clusters additionnels au besoin.
- Finalement, chaque inode contient un pointeur sur un autre inode contenant lui-même des pointeurs sur des inode contenant des pointeurs…

Inodes



source: Englander, ch. 17

Inodes

- Avantage: flexible!
  - Très rapide pour de petits fichiers (pointeurs direct)
  - Peut supporter de très gros fichiers (100’s de Go!)

Structure de disque sous UNIX

- Unix gère ses clusters sur 4 octets, comme le FAT32.
- Unix possède une structure appelée “table des inodes”.
- Un inode (“index node”) de la table est utilisé pour chaque fichier. En d’autres mots, chaque entrée dans le répertoire de fichier contient un numéro d’Inode associé au fichier de l’entrée.
- Il existe plusieurs versions de système de fichiers sous UNIX. Vous retrouverez, ext, ext2, ext3, ReiserFS, GFS, etc. Les Inodes sont présents dans la plupart de ces « Files Systems » (ext2 par exemple), mais pas dans tous (ReiserFS par exemple).

Gestion des blocs libres

- Sur un disque, les clusters peuvent être libres ou utilisés pour stocker de l'information sur un fichier. Un bloc libre est donc une partie du disque qui ne contient rien.
- Les systèmes de fichier doivent savoir quels sont les blocs du disque qui sont libres. Il existe deux manières principales pour garder cette information:
  - Un **bitmap** (Unix et NTFS): simplement un ensemble de bits consécutifs qui représentent l'état d'un cluster. Lorsque le Nième bit vaut 0 ou 1, le Nième cluster est libre ou occupé.
    - Exemple: un disque a 16 clusters. Le bitmap 1000100010001111b (ou 888Fh) indique que les clusters 0,4,8,12,13,14 et 15 sont occupés et que les autres clusters sont libres.
  - Une **liste doublement chaînée** (FAT): on peut aussi stocker la liste de tous les clusters libres avec une liste doublement chaînée. Lorsqu'un bloc libre est requis, un bloc est retiré de la liste. Lorsque d'un fichier est effacé, des blocs sont simplement remis dans la liste.

Autres systèmes de fichiers et comparaison

- Il existe plus de systèmes de fichiers que ceux présentés en classe, même si ceux-ci sont les principaux. Si vous voulez avoir un aperçu des autres systèmes de fichiers, allez voir http://en.wikipedia.org/wiki/Comparison_of_file_systems.
- Plusieurs critères existent afin de comparer les systèmes de fichiers entre eux. En voici une liste incomplète:
  - Taille maximum supportée pour un disque.
  - Taille maximum supportée pour un fichier.
  - Caractères et longueur pour les noms de fichiers.
  - Vitesse d'accès à un fichier (par exemple, la fragmentation d'un fichier ou de nombreux accès disque afin de trouver ou est situé le fichier peut ralentir l'accès considérablement).
  - Vitesse d'accès à la structure de répertoires.
  - Résistance aux fautes de disques (par exemple, un journal des transactions effectué sur le disque est-il maintenu? Aussi, quelle est la conséquence d'une faute de disque).
  - Support pour la sécurité des fichiers.
  - Types et quantité d'information maintenue en mémoire pour chaque fichier (exemple: heure et date de création).

Exercice: mémoire paginée et FAT

- Un système possède les caractéristiques suivantes:
  - la mémoire est paginée (pages de 4Ko);
  - on utilise l'allocation FAT16 pour le disque dur (clusters de 4Ko);
  - les pages qui ne sont pas en mémoire principale sont stockées sur le disque dur dans un fichier nommé « SWAP ».

- Décrivez les étapes nécessaires pour qu'un programme puisse lire une donnée à une adresse virtuelle si:
  - (mémoire paginée) il y a faute de page, mais il y a une trame libre en RAM;
  - (allocation FAT) la page nécessaire est située dans le deuxième cluster du fichier « SWAP ».

Exercice: mémoire paginée et FAT

1. Tout d'abord, le CPU traduit l'*adresse virtuelle* en *adresse physique.* Il:
    1. Sépare l'adresse virtuelle en deux: # de page et position dans la page
    2. Convertit le # de page en # de trame. Pour ce faire, il utilise la *table des pages*. Cependant, la table des pages nous indique qu'il y a faute de page!
2. Le CPU traite la faute de page. Il:
    1. Trouve une trame libre en RAM. La *table des pages inverse* peut être utilisée pour cela.
    2. Trouve la page à charger sur le disque dur. Le CPU:
        1. Regarde dans la *table de répertoire* pour déterminer l'emplacement du premier cluster correspondant au fichier SWAP.
        2. Regarde dans la *table d'allocation (FAT)* pour déterminer le cluster suivant. Ce cluster correspond à la page qu'il faut copier en RAM!
    3. Copie la page du disque dur jusque dans la trame en RAM. Pour ce faire, il peut même utiliser le *contrôleur de DMA*!
    4. Met à jour la table des pages.
3. L'adresse physique peut maintenant être lue!

Références et exercices

- Références
  - Irv Englander: chapitre 17
  - http://en.wikipedia.org/wiki/Comparison_of_file_systems