

# Gestion de fichiers

INF3173 – Principes des systèmes d'exploitation  
Automne 2024

Francis Giraldeau  
giraldeau.francis@uqam.ca

Université du Québec à Montréal

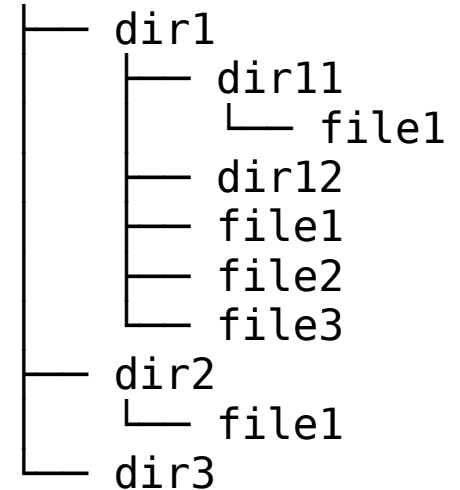


# Agenda

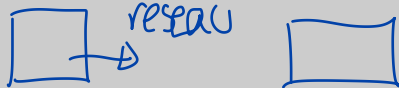
- Types de fichiers, organisation
- Opérations de gestion de fichiers
- Métadonnées
- Accès direct et avec tampon

# Chemin d'accès d'un fichier

- Organisés en arbre
- Un répertoire contient des répertoires et des fichiers
- Un fichier est une feuille
- Le nom d'un fichier est unique dans un répertoire donné
- Répertoire courant: "."
- Répertoire précédent: ".."
- Chemin absolu: /dir1/dir2/file1
- Chemin relatif: dépend du répertoire courant du programme



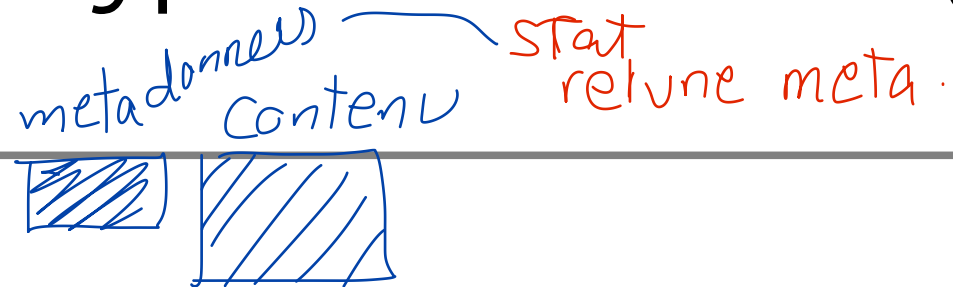
# Types de fichiers

<b>Régulier</b>	Séquence d'octets d'une taille définie.
<b>Répertoire</b>	Contient une liste de fichiers enfants.
<b>Lien dur</b>	Chemin d'accès pointant sur un fichier existant. Incrémente le nombre de référence du fichier. L'effacement décrémente le nombre de référence au fichier. Le fichier est effacé lorsque le nombre de référence atteint 0.
<b>Lien symbolique</b>	Chemin d'accès vers un fichier existant. Si le fichier référencé est effacé, alors le lien devient invalide.
<b>Tube</b> <i>→ local</i> <i>pipe</i>	Chemin d'accès pour un tampon mémoire en espace noyau. Le contenu n'est pas écrit sur le périphérique.
<b>Socket</b> 	Chemin d'accès pour une communication bidirectionnelle entre deux processus locaux. Le contenu n'est pas écrit sur le périphérique.
<b>Fichier de périphérique</b>	La lecture ou l'écriture avec ces fichiers produisent des entrées sorties avec les périphériques qui y correspondent.

# Exemples de types de fichiers (1)

## Fichier régulier

```
$ echo "Hello" > fichier.txt
$ stat fichier.txt
  File: «fichier.txt»
  Size: 6                      Blocks: 8          IO Block: 4096   fichier
Device: 801h/2049d           Inode: 3801109      Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/ francis)   Gid: ( 1000/ francis)
Access: 2022-02-12 14:17:47.177845165 -0500
Modify: 2022-02-12 14:17:47.177845165 -0500
Change: 2022-02-12 14:17:47.177845165 -0500
```



## Répertoire

```
$ mkdir gazou
$ stat gazou
  File: «gazou»
  Size: 4096                    Blocks: 8          IO Block: 4096   répertoire
Device: 801h/2049d           Inode: 3850287      Links: 2
Access: (0775/drwxrwxr-x)  Uid: ( 1000/ francis)   Gid: ( 1000/ francis)
Access: 2022-02-12 14:19:51.881848566 -0500
Modify: 2022-02-12 14:19:51.881848566 -0500
Change: 2022-02-12 14:19:51.881848566 -0500
```

# Exemples de types de fichiers (2)

## Lien dur et symbolique

```
$ ln -s fichier.txt lien_symbolique.txt
$ ln fichier.txt lien_dur.txt
```

accès	ref	taille	chemin
-rw-rw-r--	2	6	fichier.txt
-rw-rw-r--	2	6	lien_dur.txt
lrwxrwxrwx	1	11	lien_sym.txt -> fichier.txt

```
$ rm fichier.txt
```

accès	ref	taille	chemin
-rw-rw-r--	1	6	lien_dur.txt
lrwxrwxrwx	1	11	lien_sym.txt -> fichier.txt

compteur de référence

chaque fois qu'on crée  
un ln, ça va ajouter

les liens durs doivent  
être en même disque  
pour ln -s oui

Lien dur similaire à un  
compteur de référence.  
Lien vers des données  
du même disque.

Lien symbolique brisé, mais  
le lien dur existe encore  
et son contenu n'a pas été effacé

# Exemples de types de fichiers (3)

Tube nommé (socket similaire)

```
$ mkfifo myfifo
$ stat myfifo
  File: «myfifo»
  Size: 0                Blocks: 0                IO Block: 4096   FIFO
Device: 801h/2049d      Inode: 3801135         Links: 1
Access: (0664/prw-rw-r-- )  Uid: ( 1000/ francis)   Gid: ( 1000/ francis)
Access: 2022-02-12 14:40:24.137882164 -0500
Modify: 2022-02-12 14:40:24.137882164 -0500
Change: 2022-02-12 14:40:24.137882164 -0500
```

# Exemple de type de fichier (4)

→ envoi info aux ferru et à mesur

Fichier de périphérique caractère

```
# cat /dev/input/by-id/usb-Logitech_Keyboard-event-kbd | hexdump
00000000 185d 4f38 0000 0000 2a89 0007 0000 0000
00000010 0004 0004 0028 0007 185d 4f38 0000 0000
00000020 2a90 0007 0000 0000 0001 001c 0000 0000
00000030 185d 4f38 0000 0000 2a91 0007 0000 0000
00000040 0000 0000 0000 0000 185f 4f38 0000 0000
[...]
```

Les octets émis par le clavier à chaque pression d'une touche sont affichés à mesure

Fichier de périphérique bloc

```
# hexdump /dev/sda -n 12 -s 500
00001f4 0000 0000 0000 0000 0000 0000 aa55
```

Déplacement quelconque dans l'espace d'adresse du périphérique

Signature Master Boot Record!



# Opérations sur les fichiers (1)

<b>Création</b>	Crée un nouveau fichier à un chemin d'accès.
<b>Ouverture</b>	Ouvre un fichier existant à partir de son chemin d'accès.
<b>Fermeture</b>	Indiquer que le traitement du fichier est terminé.
<b>Lecture</b>	Retourne une portion du contenu du fichier. Opération relative à l'emplacement courant.
<b>Écriture</b>	Écrit le contenu d'un fichier à partir d'un tampon. Opération relative à l'emplacement courant.
<b>Agrandissement</b>	Écrit le contenu à la fin du fichier. Augmente la taille du fichier.
<b>Troncature</b>	Réduit la taille du fichier en définissant la nouvelle taille.

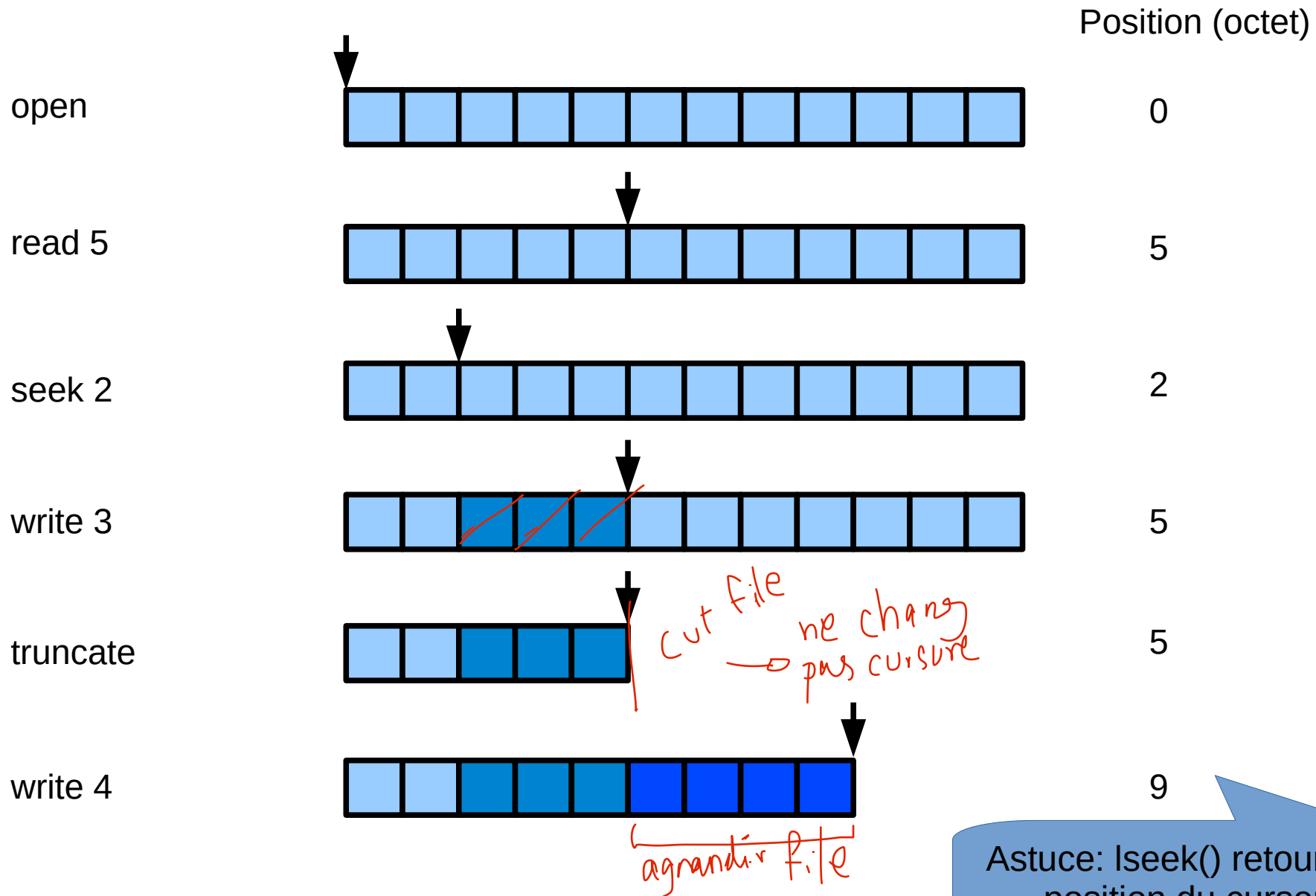
# Opérations sur les fichiers (2)

<b>Renommer</b>	Change le chemin d'accès du fichier.
<b>Verrouillage</b>	Verrouille un fichier ou une partie du contenu pour assurer l'exclusion mutuelle.
<b>Suppression</b>	Supprime le fichier et son contenu. Les blocs peuvent être réutilisés.
<b>Synchronisation</b>	Attendre que le contenu soit écrit sur le périphérique.
<b>Miroir en mémoire</b>	Le contenu du fichier devient accessible comme s'il était chargé entièrement en mémoire. Les modifications en mémoire sont reportées dans le fichier.
<b>Status</b>	Retourne les informations sur le fichier, comme la taille, la date de modification, etc.
<b>Lister</b>	Liste le contenu d'un répertoire.
<b>Permissions</b>	Change les droits d'accès du fichier.

# Gestion de fichiers réguliers

- Création: `creat()`
- Ouvrir: `open()`
- Fermer: `close()`
- Lire: `read()`
- Écrire: `write()`
- Rétrécir ou agrandir: `truncate()`
- Renommer: `rename()`
- Informations: `stat()`
- Permissions: `chmod()`
- Déplacement du curseur: `lseek()`
- Supprimer: `unlink()`
- Lister un répertoire: `readdir()`

# Opérations relative au curseur



Astuce: lseek() retourne la position du curseur

# Exercice : que va afficher ce code?

```
void mystere() {  
    char val[4] = {0xCA, 0xFE, 0xBA, 0xBE};  
    char buf[2];  
    int fd = open("data", //  
                  O_CREAT | O_TRUNC | O_RDWR, //  
                  0644); // Operation 1  
    ftruncate(fd, 8); // Operation 2  
    lseek(fd, 2, SEEK_SET); // Operation 3  
    write(fd, val, sizeof(val)); // Operation 4  
    lseek(fd, 4, SEEK_SET); // Operation 5  
    read(fd, buf, sizeof(buf)); // Operation 6  
    printf("%x %x\n", 0xFF & buf[0], 0xFF & buf[1]);  
}
```

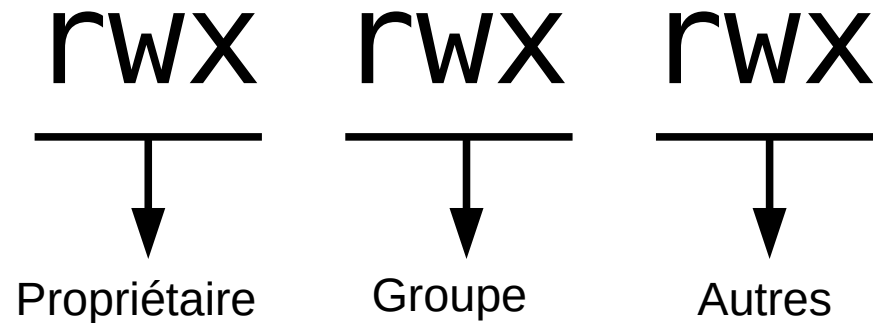
Pos Cur

0  
0  
2  
6  
4  
6

Handwritten annotations: Red arrows and boxes highlight the file operations and the data being read. A red box around the 'BA, BE' in the write operation points to the '4' in the lseek operation. A red box around the '0, 0' in the read operation points to the '0' in the lseek operation. A red box around the '0, 0' in the printf operation points to the '0' in the lseek operation.

Réponse: ba be

# Permissions



3-bit : octal

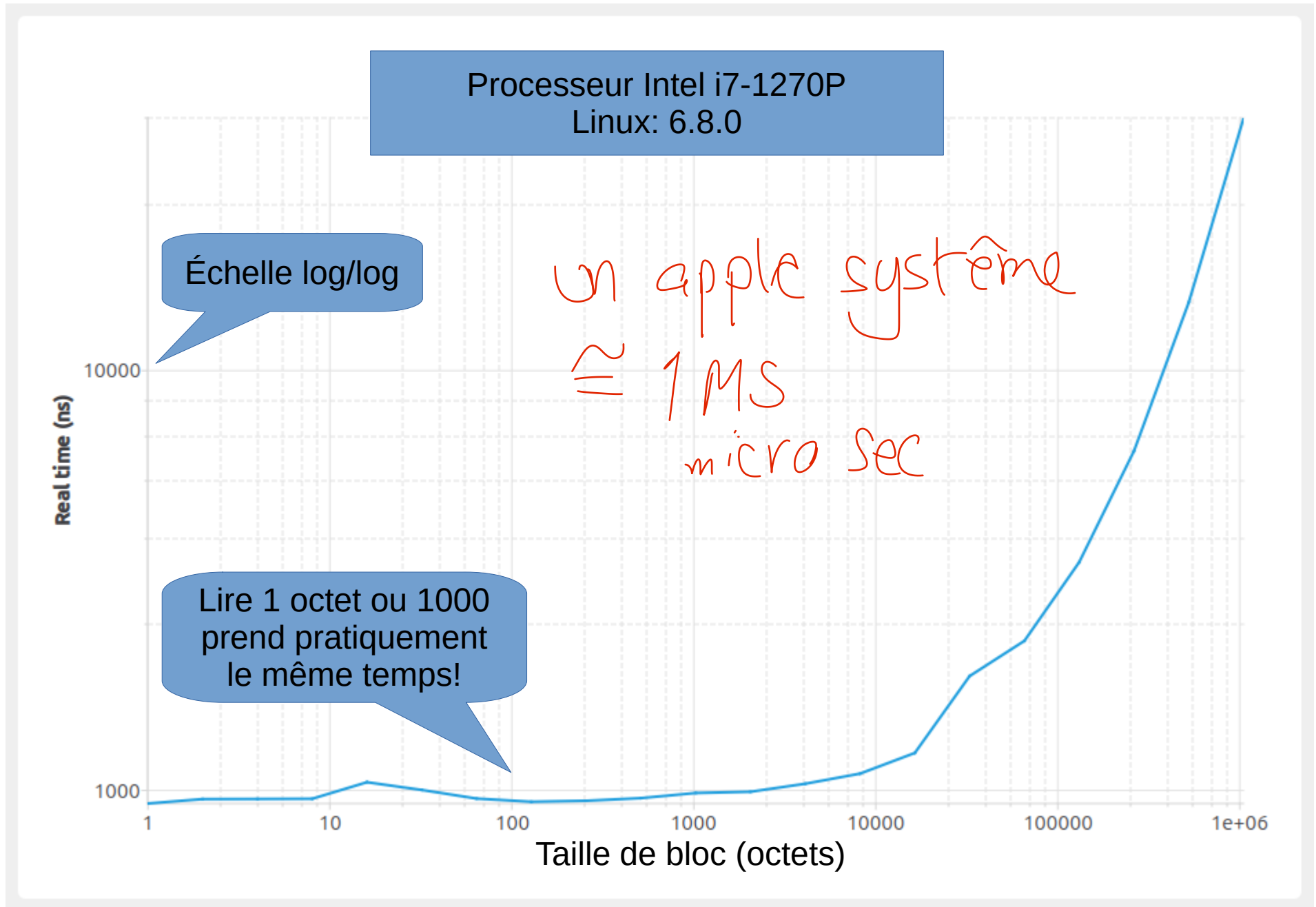
Binaire	:	000	001	010	011	100	101	110	111
Octal	:	0	1	2	3	4	5	6	7
Permissions:		- - -	- - x	- w -	- w x	r - -	r - x	rw -	rw x

Binaire	:	111	101	101	110	100	100
Octal	:	7	5	5	6	4	4
Permissions:		rw x	r - x	r - x	rw -	r - -	r - -

# Différence entre write() et fwrite()

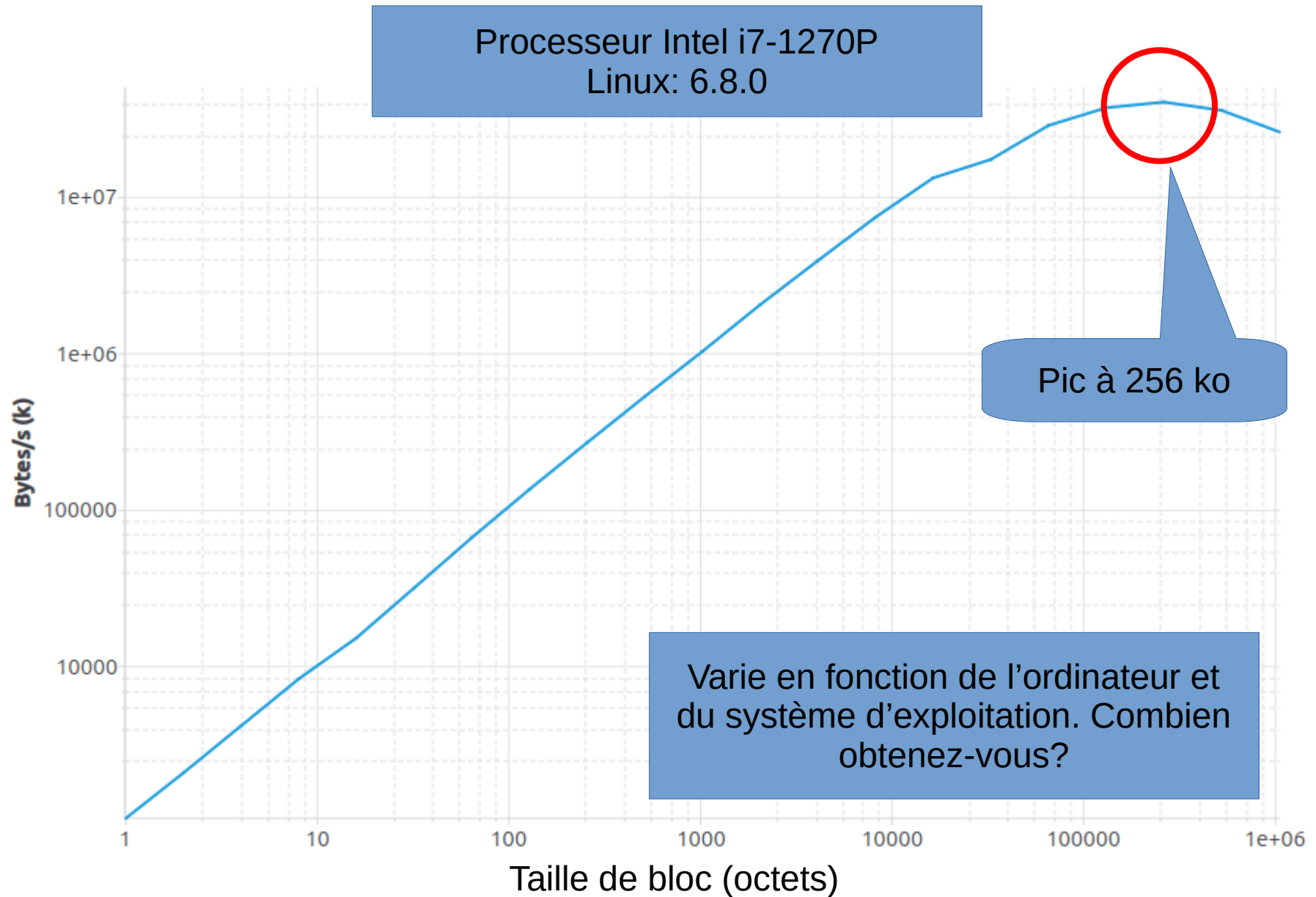
- Chaque appel à write() produit un appel système.
- Un appel à fwrite() copie les données dans un tampon temporaire.
- Lorsque le tampon est plein, un appel système write() est fait.
- Permet de réduire le surcoût des appels systèmes pour de petits transferts.
- Exemple: printf()

# Délais de lecture read() selon la taille de bloc





# Débit de lecture read() selon la taille de bloc



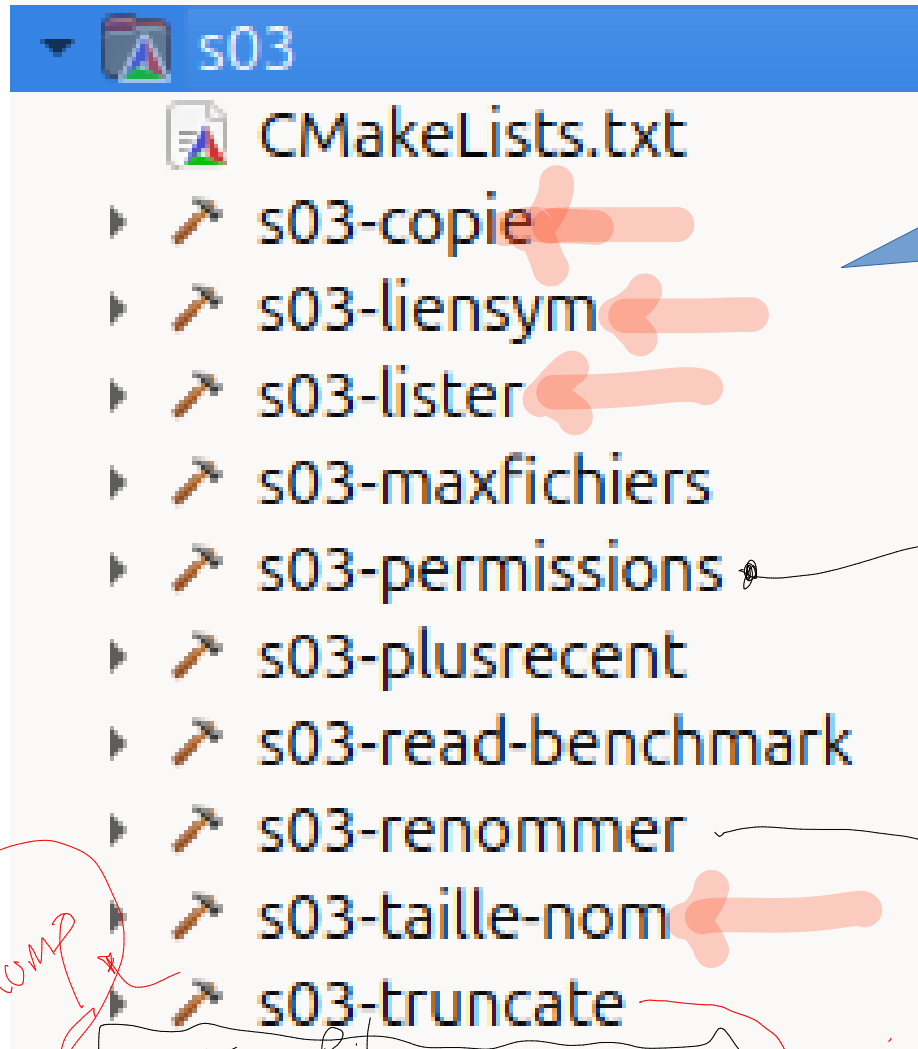
# Entêtes utiles

```
#include <fcntl.h>           // File Control: open, permissions
#include <stdio.h>           // printf, fread, fwrite
#include <stdlib.h>          // random, sort, etc.
#include <sys/resource.h>    // getrlimit
#include <sys/stat.h>        // stat meta-info
#include <time.h>            // struct timespec - date creation
#include <unistd.h> posix   // fork, close, unlink
```

*get\_directory\_entry  
opendir (".-");*

*seek(0) → met 0 par tout  
( printf( "%s " ) → il arrete quand  
il arrive a 0 → null )  
( à la place on doit mettre  
un for loop )*

# Exemples



Expériences pour maîtriser les différentes fonctions

hexdump  
-C  
cut a file in 2 Byte  
→ il met 0 à la place de bit

start → st-mode  
↳ pour permission  
Contenu ne change pas