Système d'exploitation

Eliana Carozza

08/09/2025

Un système d'exploitation est un programme ou un ensemble de programmes dont le but est de gérer les ressources matérielles et logicielles d'un ordinateur.

L'utilisateur interagit avec des programmes. Ces derniers ont besoin d'utiliser les ressources de la machine. Le système d'exploitation fournit un ensemble de fonctions primitives permettant d'interagir avec le matériel.

Ses principales composantes sont :
— l'*ordonnanceur*, qui décide quel programme s'exécute ;
— le *gestionnaire de mémoire*, qui répartit la mémoire ;
— les systèmes de fichiers, qui définissent la manière de stocker et organiser les fichiers ;
— la pile réseau, qui implémente différents protocoles de communication ;
— les pilotes de périphériques, qui assurent la gestion des périphériques matériels.

Le standard POSIX. Malgré la grande diversité des systèmes d'exploitation, il existe un ensemble de standards regroupés sous le nom de *POSIX*. La plupart des systèmes modernes sont compatibles avec ce standard, largement inspiré par le système UNIX.

Windows, en revanche, n'est pas un dérivé d'UNIX.

1 L'interface système ou *shell*

L'interface système est un programme permettant à l'utilisateur d'interagir avec le système d'exploitation. Il s'agit d'une invite de commandes dans laquelle on saisit des instructions spécifiques.

Historiquement, ce type d'interface était utilisé avant l'apparition des interfaces graphiques avancées. Aujourd'hui, on emploie un programme graphique appelé *émulateur de terminal*, qui continue d'afficher une invite de commandes.

1.1 Utilisateurs et groupes

Les systèmes POSIX sont multi-utilisateurs. Chaque utilisateur possède un identifiant de connexion associé à un mot de passe. L'ensemble des données de l'utilisateur, ainsi que ses fichiers personnels, constitue son *compte*. L'ensemble des interactions d'un utilisateur authentifié avec le système est appelé une *session*.

Les utilisateurs peuvent être regroupés en *groupes*. Un utilisateur appartient à un groupe principal et éventuellement à des groupes secondaires. L'identifiant numérique du groupe principal est appelé *GID*. La commande `id` permet d'afficher les identifiants numériques et les groupes de l'utilisateur courant.

Il existe enfin un utilisateur spécial nommé `root`, dont l'UID et le GID valent 0. On l'appelle le *super-utilisateur* : il correspond à l'administrateur système.

1.2 Arborescence de fichiers et chemins

L'ordinateur stocke sur son disque dur des fichiers, contenant des données. Ces fichiers peuvent être lus, modifiés, renommés, copiés ou supprimés.

Certains fichiers spéciaux, appelés *répertoires* ou *dossiers*, ont pour but de contenir d'autres fichiers ou répertoires.

Il existe un *répertoire racine*, et l'ensemble des fichiers et répertoires forme une *arborescence*.

Chemins et navigation. Le répertoire courant peut être affiché avec la commande `pwd`. On crée un répertoire avec `mkdir`, et on change de répertoire avec `cd`.

La chaîne de caractères affichée est appelée *chemin*.

— Si le chemin commence par un `/`, on parle de *chemin absolu*.
— Sinon, il s'agit d'un *chemin relatif*.
— Le nom spécial `..` désigne le répertoire parent, et le nom spécial `.` désigne le répertoire courant.

La commande `ls` permet d'afficher le contenu d'un répertoire.

Périphériques et arborescence Unix. Dans les systèmes Unix, l'arborescence inclut également les périphériques externes. Associer un périphérique à un répertoire est une opération privilégiée.

— `mount` : associer un périphérique à un répertoire ;
— `umount` : désassocier un périphérique.

1.3 Permissions et propriété des fichiers

Le système de fichiers définit des *permissions* pour trois catégories d'utilisateurs : le propriétaire, le groupe propriétaire et les autres utilisateurs.

La commande `ls -l` permet un affichage détaillé des fichiers. Pour chaque fichier, sont affichés :

— les permissions,
— le nombre de liens physiques vers le fichier,
— le nom du propriétaire et du groupe,
— la taille, la date et l'heure de dernière modification,
— le nom du fichier.

Exemple

```
alice$ chmod g+w,o-r Photos/img_001.jpg
```

Pour les répertoires : - le droit en lecture signifie que son contenu peut être listé, - le droit en écriture signifie que l'on peut modifier les entrées du répertoire (créer, supprimer ou renommer des fichiers), - le droit en exécution signifie que l'on peut en faire le répertoire courant et afficher les informations détaillées sur ses entrées.

1.4 Systèmes de fichiers, liens physiques et symboliques

Un *système de fichiers* (filesystem en anglais) est un ensemble de conventions et de structures de données permettant de stocker des données sur un support physique.

Chaque système de fichiers implémente, à sa manière, les primitives génériques offertes par le système d'exploitation :

— création d'un fichier,
— ouverture en lecture ou écriture,
— lecture ou écriture partielle,
— suppression d'un fichier,
— copie ou renommage.

Chaque système de fichiers possède des caractéristiques propres. C'est l'*administrateur système* qui choisit quel système utiliser en fonction du périphérique et des besoins.

FAT32. Ce système est simple et dépourvu de fonctionnalités avancées. Il ne permet pas de gérer finement les permissions ni les propriétaires des fichiers. En revanche, sa simplicité en fait encore un choix privilégié pour les périphériques amovibles.

Ext4. Outre la gestion des droits d'accès et de la propriété des fichiers, ce système sauvegarde également les dates de création, d'utilisation et d'accès. Il inclut aussi un mécanisme de reprise sur panne, permettant de récupérer les données en cas d'interruption imprévue.

Modes d'accès aux périphériques. Il existe deux modes d'accès aux périphériques externes :

— l'*accès par octet*, qui permet de lire et d'écrire octet par octet (exemples : clavier, souris, carte son, ports USB) ;
— l'*accès par bloc*, qui permet au système d'interagir par blocs d'octets (exemples : périphériques de stockage).

1.5 Inodes et méta-données.

La norme POSIX impose certains concepts fondamentaux : chaque fichier possède un identifiant unique appelé *inode*. Un inode est un entier qui identifie de manière unique un fichier et stocke les informations suivantes :

— taille du fichier,
— permissions,
— propriétaires et groupes,
— identifiant du périphérique physique,
— adresse physique du fichier,
— dates de création, d'accès et de modification.

L'option `ls -i` permet d'afficher l'inode de chaque fichier ou répertoire.

Le nom du fichier n'est pas stocké dans l'inode mais dans le répertoire qui le contient. Par convention, l'entrée `..` désigne le répertoire parent, et `.` le répertoire courant. La commande `ls -a` permet d'afficher également les fichiers dont le nom commence par un point.

Accès par chemin. Par exemple, lorsque l'utilisatrice `alice` exécute un programme qui accède à `Photos/img_002.jpg` :

1. décomposition du chemin `Photos/img_002.jpg`,
2. recherche dans le répertoire courant d'une entrée nommée `Photos`,
3. récupération de l'inode associé,
4. accès aux données de cet inode,
5. recherche d'une entrée nommée `img_002.jpg`,
6. récupération de l'inode associé,
7. accès aux données par le programme.

1.6 Liens physiques.

Il est possible d'associer plusieurs noms à un même inode grâce aux *liens physiques*. La commande suivante crée un lien physique entre deux fichiers :

```
alice$ ln Photos/img_002.jpg photo.jpg
```

Les deux fichiers partagent le même inode : ils pointent donc vers la même zone de données et les mêmes méta-données. Modifier les permissions de l'un modifie aussi celles de l'autre :

```
alice$ chmod g-r,o-r photo.jpg
alice$ ls -l -i Photos/photo.jpg
1573296 -rw------- 2 alice mp2i 1300458 juil. 1 15:02 photo.jpg
```

Contenu du répertoire `Photos/` :

```
total 2672
1573295 -rw-r--r-- 1 alice mp2i 1431099 juil. 1 15:02 img_001.jpg
1573296 -rw------- 2 alice mp2i 1300458 juil. 1 15:02 img_002.jpg
```

De même, modifier le contenu de `photo.jpg` revient à modifier `Photos/img_002.jpg`, puisque les deux noms pointent vers le même fichier.

En revanche, supprimer un des deux noms ne fait que supprimer l'entrée du répertoire : les données ne sont effectivement effacées que lorsqu'aucun nom ne pointe plus vers l'inode.

Les liens physiques permettent ainsi d'organiser logiquement des fichiers sans augmenter l'espace occupé. Par exemple, Alice peut posséder mille images `img_000.jpg` à `img_999.jpg` dans le répertoire `Photos`, et créer des liens physiques pour certains fichiers dans un sous-répertoire `a_imprimer`. Cela évite les copies inutiles et les déplacements fastidieux.

Limitations des liens physiques.

— ils ne peuvent être créés qu'entre deux fichiers (et non pas entre deux répertoires) ;
— ils ne peuvent exister qu'entre fichiers situés sur le même support physique.

La première limitation empêche la création de cycles dans l'arborescence. La seconde est due au fait que les inodes sont propres à un système de fichiers.

1.7 Liens symboliques.

Pour contourner ces limites, on utilise les *liens symboliques* (ou *symlinks*). Ils consistent à associer un fichier spécial contenant le chemin d'un autre fichier. [1]

Un *lien symbolique* est un fichier spécial dont le contenu est simplement un chemin vers un autre fichier. On peut créer un lien symbolique en passant l'option `-s` à la commande `ln` :

```
alice$ ln -s Photos Images
```

Cette commande crée un lien symbolique `Images` vers le répertoire `Photos`. L'affichage détaillé avec `ls -l` permet de repérer les liens :

```
alice$ ls -l Photos Images
lrwxrwxrwx 1 alice mp2i 6 juil. 1 17:59 Images -> Photos
```

Dans cet exemple, le fichier `Images` est bien un lien symbolique : - les permissions commencent par la lettre `l` (pour *link*) ; - toutes les permissions sont positionnées pour tout le monde (propriétaire, groupe et autres) [2] ; - la taille du fichier est de 6 octets, correspondant au nombre de caractères de la cible (`Photos`).

Ouvrir un lien symbolique vers un fichier, ou entrer dans un lien symbolique vers un répertoire, revient à accéder directement à la cible :

```
alice$ cd Images
alice$ ls -l
total 2672
-rw-r--r-- 1 alice mp2i 1431099 juil. 1 15:02 img_001.jpg
-rw-r--r-- 2 alice mp2i 1300458 juil. 1 15:02 img_002.jpg
```

Toutes les commandes (et plus tard toutes les fonctions systèmes) suivent automatiquement les liens symboliques, quel que soit le nombre de liens enchaînés :

```
alice$ ln -s Images Pictures
alice$ ln -s Pictures Photos2
alice$ ls -l
```

Cette suite de commandes crée un lien `Pictures` vers `Images`, puis un lien `Photos2` vers `Pictures`. La commande `cd Photos2` suit automatiquement les liens jusqu'à la cible finale.

Les liens symboliques n'enregistrant qu'un chemin, leur cible n'a pas besoin d'être s

1.8 Ligne de commande et motifs glob

Pour exécuter une commande dans un terminal, on saisit après l'invite de commande un nom de commande suivi d'une liste d'arguments, puis on valide avec la touche « Entrée » :

```
alice$ commande arg1 arg2 ... argn
```

1. Le système Windows propose un mécanisme similaire appelé « raccourcis ».
2. Certains systèmes Unix, comme BSD et MacOS X, permettent de modifier les permissions des liens symboliques.

La commande peut être donnée sous forme de *chemin relatif* ou *absolu*. Le shell s'attend alors à ce que le chemin désigne un fichier exécutable par l'utilisateur courant, et il lance son exécution.

Les arguments `arg1, ..., argn` peuvent contenir des caractères normaux ainsi que des *caractères spéciaux* utilisés dans les *motifs glob* :

— `*` : représente n'importe quelle séquence de caractères. Exemple : `Photos/img*.jpg` liste tous les fichiers commençant par `img` et finissant par `.jpg`.
— `?` : représente un seul caractère.
— `[c_1c_2...c_n]` : représente un caractère appartenant à l'ensemble listé.
— `[^c_1-c_n]` : représente un caractère qui n'appartient pas à l'intervalle indiqué. Exemple : `[a-z]` désigne une lettre minuscule, `[^0-9]` un caractère qui n'est pas un chiffre.

Ce processus de substitution des motifs est appelé *expansion de la ligne de commande*. Il est effectué par le shell avant l'exécution du programme.

```
alice$ ls /home/alice/*/img?*[23].jpg
/home/alice/Photos/img_002.jpg
```

2 Fichiers et redirections

Les systèmes POSIX proposent à chaque programme trois fichiers spéciaux :

— *l'entrée standard* (`stdin`) : un fichier virtuel dans lequel un programme peut lire. Par défaut, elle est reliée au clavier : lire dans `stdin` bloque le programme jusqu'à ce que l'utilisateur saisisse une touche ;
— *la sortie standard* (`stdout`) : reliée à l'affichage de la console, elle affiche les caractères produits par le programme ;
— *la sortie d'erreur* (`stderr`) : utilisée pour afficher les messages d'erreur.

Dans un shell POSIX, l'opérateur `>` permet de rediriger la sortie standard d'un programme vers un fichier :

```
alice$ ls -l Photos/* > liste_photos.txt
```

La commande ci-dessus n'affiche rien dans le terminal : les lignes qui auraient dû apparaître sont écrites dans le fichier `liste_photos.txt`.

L'opérateur `2>` permet de rediriger les messages d'erreur. Par défaut, `>` et `2>` écrasent le contenu du fichier cible. Leurs variantes `»` et `2»` ajoutent le nouveau contenu à la fin du fichier (s'il n'existe pas, il est créé).

Une opération puissante, introduite par UNIX et reprise par la norme POSIX, est la redirection entre commandes à l'aide de l'opérateur `|`, appelé *pipe* (ou *tube* en français).

```
alice$ ls -l */* | sort -k 5 -n -r | head -n 1
```

Cette commande fonctionne en trois étapes :

1. `ls -l */*` : affiche la liste des fichiers ;
2. `sort -k 5 -n -r` : trie la sortie de `ls` selon le 5^{e} champ (taille), traité comme un nombre, en ordre décroissant ;
3. `head -n 1` : conserve uniquement la première ligne, c'est-à-dire le fichier le plus volumineux.

2.1 Commandes utiles

Le tableau suivant présente quelques-unes des commandes les plus courantes :

`cp`	Copie un fichier.
`mv`	Déplace ou renomme un fichier.
`rm`	Supprime définitivement un fichier.
`cat`	Affiche le contenu d'un fichier dans la console.
`cut`	Extrait certaines colonnes d'un fichier.
`sort`	Trie le contenu d'un fichier.
`head`	Affiche les premières lignes d'un fichier.
`tail`	Affiche les dernières lignes d'un fichier.
`wc`	Compte le nombre de caractères, de mots et de lignes d'un fichier.
`ls`	Liste les fichiers d'un répertoire.
`mkdir`	Crée un répertoire.
`echo`	Affiche un message dans la console.
`ln`	Crée un lien symbolique ou physique.

Il existe des *pages de manuel* (`man`) pour toutes les fonctions des bibliothèques C et OCaml. On peut rechercher les pages liées à un sujet en utilisant la commande :

```
1 alice$ man -k sujet
```