Système d'Exploitation

Introduction

Université Fde Tours
Faculté des Sciences et Techniques
Antenne Universitaire de Blois

Licence Sciences et Technologies

Mention : Informatique

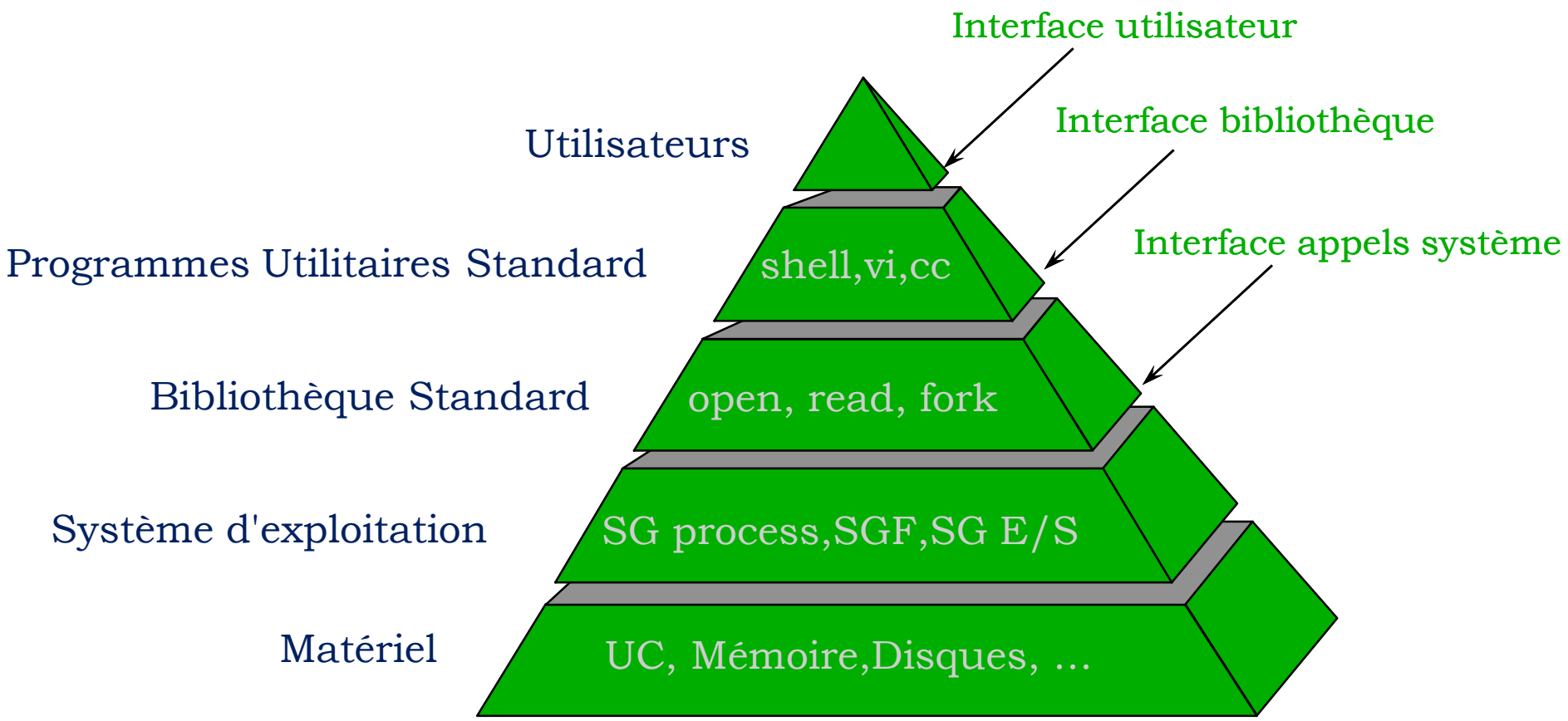
2ème Année

Mohamed Taghelit
taghelit@univ-tours.fr

Qu'est-ce qu'un Système d'Exploitation

- Un programme qui agit comme un intermédiaire entre un utilisateur d'un ordinateur et le matériel informatique
- Un SE est un allocateur de ressources
  - Gère toutes les ressources,
  - Décide, dans le cas de requêtes contradictoires, de l'utilisation efficace et équitable des ressources .
- Un SE est un programme de contrôle
  - Contrôle l'exécution des programmes pour éviter les erreurs et la mauvaise utilisation de l'ordinateur
- Objectifs du système d'exploitation :
  - Exécute les programmes utilisateurs,
  - Facilite l'utilisation du système informatique,
  - Utilise de manière efficace le matériel informatique.

Structure d'un Système Informatique



Identification d'un Utilisateur dans le Système

- Identification d'un Utilisateur
  - *login*: *UNAME* (*User NAME*), nom par lequel le système reconnaît l'utilisateur et ouvre une session de travail
  - *mot de passe* : permet l'authentification de l'utilisateur du *login* introduit

- Principe

  L'identification de l'utilisateur et la définition des permissions qui lui sont appliquées reposent sur deux notions :
  - Identifiant Utilisateur : *UID* (*User IDentification*), numéro unique associé à chaque utilisateur (*login*). Information système qui référence un utilisateur une fois qu'il est connecté.
  - Groupe d'Utilisateurs : *GID* (*Group IDentification*), numéro unique associé à chaque groupe d'utilisateurs. A un groupe est également associé un nom : *GNAME* (*Groupe NAME*). Remarque : chaque utilisateur appartient à un groupe principal et peut être membre d'autres groupes.

Les Fichiers `/etc/passwd` et `/etc/group`

- Le fichier `/etc/passwd` est un fichier texte contenant autant de lignes que d'utilisateurs enregistrés.

```
-rw-r—r--  1  root  root   1258  jun  15  10:56  /etc/passwd
```

```
pierre : gT6bx07ePkB2d : 500 : 550 : Pierre MARTIN,Enseignant,20110905 : /home/pierre : /bin/bash
```

UNAME — *mot de passe* — *UID* — *GID* — *gecos* — *homedir* — *login shell*

- Le fichier `/etc/shadow` est un fichier texte contenant les mots de passe cryptés des utilisateurs enregistrés.

```
-rw-r—r--  1  root  root   1258  jun  15  10:56  /etc/passwd
---------  1  root  root    943  jun  15  10:56  /etc/shadow
```

```
pierre : gT6bx07ePkB2d : 12566 : 0 : 99999 : 7 :  :  :
```

UNAME

*mot de passe crypté (x ou * dans passwd)*

nbr jours écoulés → dernier chgt passwd*

nbr min de jours entre 2 chgts

nbr max de jours de validité du passwd

nbr de jours avant expiration pour avertissement

nbr de jours après expiration pour désactivation

*date expiration passwd en nbr de jours**

réservé

** depuis 01/01/1970*

- Le fichier `/etc/group` est un fichier texte contenant autant de lignes que de groupes enregistrés.

```
-rw-r--r--  1  root  root   1041  jun  29  11:33  /etc/group
```

```
enseignants : ! : 550 : pierre, alain, paul,…
```

GNAME — *mot de passe* — *GID* — *liste des membres*

Commandes d'Identification

- La commande `id`
  - elle affiche les informations sur l'utilisateurs qui tape la commande

```
$ id
uid=500(pierre) gid=550(enseignants) groups=551(calcul), 600(licence)
```

UID, UNAME — *groupe principal (actif)* — *groupes supplémentaires*

  - elle peut être utilisée pour afficher les informations d'un autre utilisateur

```
$ id admin
uid=0(laurand) gid=0(administrateur) groups=110(scolarite)
```

- La commande `groups`
  - elle affiche les groupes auquel l'utilisateur courant (ou fourni en paramètre) est membre

```
$ groups
pierre : enseignants calcul licence
```

- La commande `newgrp`
  - elle permet de changer (le temps de la session) de groupe actif.

```
$ newgrp calcul
$ id
uid=500(pierre) gid=551(calcul) groups=550(enseignants),551(calcul),
600(licence)
```

  - NB : le changement de groupe actif agit sur les propriétés des fichiers et répertoire créés par la suite.

Commandes d'Identification 2

- La commande `who`
  - elle affiche la liste des utilisateurs connectés

```
$ who
admin    tty0         jun 25 08:18
pierre   pts/2        jun 25 09:05          (papin.univ-tours.fr)
```

UNAME — *numéro de la ligne de communication* — *date et heure de connexion* — *adresse du poste de travail*

  - elle peut être utilisée pour afficher uniquement la ligne concernant l'utilisateur

```
$ who am i /* whoami */
pierre   pts/2        jun 25 09:05          (papin.univ-tours.fr)
```

- La commande `w`
  - identique à `who` mais affiche la commande que l'utilisateur exécute plutôt que son poste

```
$ w
10:31AM       up 7 days,     2 users, load average: 7.42, 4.69, 6.53
User    tty   login@         idle   JCPU    PCPU    what
admin   tty0  08:18AM        3      27      11      csh
pierre  pts/2 09:05AM        0      43:27   43:27   /bin/firefox
```

Charge processeur pour les dernières 1, 5 et 15 minutes.

  - chaque utilisateur peut voir ce que font les autres. Commande souvent utilisée par l'administrateur, qui peut éventuellement envoyer un message à un utilisateur (`write`) ou à tous (`wall`).

Commandes d'Identification 3

- La commande `finger`
  - elle affiche les information *GECOS* des utilisateurs connectés

```
$ finger
Login  Name                 TTY     Idle    When           Site Info
admin  Pierre Laurent       tty0    5       Mon 08:18      ADMIN
pierre Pierre MARTIN        pts/2   2:15    Mon 09:05      Licence 2011/12
```

  - elle peut être utilisée avec un argument (*login*) pour afficher uniquement l'identité d'un utilisateur (même s'il n'est pas connecté)

```
$ finger pierre
Login name: pierre              In real life : Pierre MARTIN
Site Info: Enseignant, 20100905,
Directory: /home/pierre                 Shell: /bin/bash
On since Sep 05 08:15:00 on pts/2
       from papin.univ-tours.fr
       (messages off)
No Plan.
```

les commandes `who`, `w` et `finger` fournissent des informations sur les utilisateurs et peuvent faciliter le travail de surveillance de l'administrateur.

Modification des Informations d'un Utilisateur

- La commande `passwd`
  - elle permet de modifier le mot de passe de l'utilisateur

```
$ passwd
Changing password for "pierre"
pierre's Old password: xxxxx                    /* évite à une tierce personne de le faire */
pierre's New password: xxxxxx
Enter the new password again: xxxxxx
```

- La commande `chsh` (*`change shell`*)
  - elle permet de modifier le *`login shell`*

```
$ chsh                          /* mode interactif  */
$ chsh pierre /bin/csh          /* met en place le nouveau login shell, /bin/csh */
```

- La commande `chfn` (*`change finger information`*)
  - elle permet de modifier la valeur du champs *`GECOS`*

```
$ chfn
pierre's current gecos:
                "Pierre MARTIN, Enseignant, 20100905,"
Change (yes) or (no)? > y
To?>Pierre-Paul MARTIN,Professeur,20100901,
```

Les commandes précédentes modifient le contenu du fichier `/etc/passwd (/etc/shadow)`

Filtrage de l'Affichage

- Les commandes précédentes d'identification peuvent être enchainées avec des commandes de comptabilisation et de filtrage pour fournir des résultats plus précis. L'enchainement de commandes utilise le caractère "`|`" qui traduit une redirection des flux entrants/sortants (*pipe*).

- La commande `wc [-cwl]` (*Word Count*)
  - elle compte le nombre de caractères `[-c]`, de mots `[-w]` ou de lignes `[-l]` d'un texte

```
$ who | wc -l /* affiche le nombre d'utilisateurs connectés */
7
```

- La commande `grep`
  - elle filtre le flux de texte qu'elle reçoit et ne laisse passer que les lignes contenant le motif donné en argument

```
$ w | grep firefox    /* affiche tous les utilisateurs qui exécutent firefox */
pierre  pts/2 09:05AM        0       43:27   43:27   /bin/firefox
alain   pts/1 09:21PM        2       16:12   16:12   /bin/firefox

$ w | grep firefox | wc -l
2
```

Ajout d'Utilisateur et de Groupe d'Utilisateurs

- Ajouter un nouveau groupe d'utilisateurs revient à procéder manuellement à l'étape suivante :
  1. Modification du fichier `/etc/group`,
- Ajouter un nouvel utilisateur revient à procéder manuellement aux étapes suivantes :
  2. Modification du fichier `/etc/passwd`,
  3. Positionnement du mot de passe initial,
  4. Création du répertoire personnel de l'utilisateur et changement de propriétaire,
  5. Copie des fichiers d'environnement.
- Ajouter un nouvel utilisateur par les commandes `useradd` et `groupadd`

```
$ groupadd -g 601 licence2
$ useradd -u 550 -g licence2 -c 'Bob SLEIGH,Licence2
     Info' -d /home/bob -m -s /bin/bash bob
$ ls -l /home | grep bob
drwxr-xr-x 2 bob licence2 4096 sep 25 8:15 /home/bob
```

1	`$ vi /etc/group`
	…
	`licence:x:600:pierre`
	`licence2:x:601:`
	`$ echo "licence2:!::" >> /etc/gshadow`

2	`$ vi /etc/passwd`
	…
	`bob::550:601:Bob SLEIGH,`
	`Licence2:/home/bob:/bin/bash`
	`$ vi /etc/shadow`
	…
	`bob::11654:-1:99999:-1:::`

3	`$ passwd bob`
	`Changing password for user bob`
	`New UNIX password:` **`xxxxxx`**
	…

4	`$ mkdir /home/bob` `$ chown bob.licence2 /home/bob`
5	`$ cp /etc/skel/.bas* /home/bob` `$ chown -R bob.licence2 /home/bob`

Suppression d'Utilisateur et de Groupe d'Utilisateurs

- Supprimer un groupe revient à supprimer la ligne dans le fichier `/etc/group` qui le décrit.
  - si des utilisateurs appartiennent au groupe supprimé, ils deviennent "orphelins" de groupe (le `GID` remplace le nom de groupe).

```
$ ls -l /home | grep bob        /* après suppression du groupe licence2 de /etc/group */
drwxr-xr-x 2 bob 601 4096 sep 25 8:15 /home/bob
```

  - La commande `groupdel` permet de supprimer un groupe qui n'est groupe principal d'aucun utilisateur (suppression à la fois dans le fichier `/etc/group` et `/etc/gshadow`)

```
$ groupdel licence2
groupdel : impossible d'enlever l'utilisateur de son groupe primaire
```

- Supprimer un utilisateur revient à supprimer la ligne dans le fichier `/etc/passwd` qui le décrit.
  - si l'utilisateur supprimé possède toujours des fichiers et répertoires, ils deviennent "orphelins" de propriétaire (l'`UID` remplace le nom de propriétaire).

```
$ ls -l /home | grep bob        /* après suppression de l'utilisateur bob de /etc/passwd */
drwxr-xr-x 2 550 licence2 4096 sep 25 8:15 /home/bob
```

Les fichiers et répertoires de l'utilisateur supprimé doivent êtres aussi supprimés. S'ils ne le sont pas et que l'`UID` de l'utilisateur supprimé est affecté à un nouvel utilisateur, ce dernier en devient propriétaires.

  - La commande `userdel` permet de supprimer un utilisateur

```
$ userdel -r bob        /* l'option -r permet de supprimer également le répertoire et le fichier contenant le
                           courriel. Si absente, alors il faut faire la suppression "à la main" */
```

L'Environnement de Travail ou shell

- Principe : au début d'une session, chaque utilisateur exécute un logiciel initial, ou interpréteur de commandes, son *login shell*. Il constitue un environnement de travail composé de commandes, de variables, de fichiers de démarrage, etc.

- Les différents shells, répartis en deux familles :
  - Bourne Shell : le Bourne Shell (*bsh*), le Korn Shell (*ksh*) et le Bourne Again Shell (*bash*)
  - C Shell : le C Shell (*csh*) et le *tcsh*
  - Le shell POSIX (*sh*) → *ksh* ou *bash* (shells respectant cette norme)

```
$ ls -il /bin/*sh*
96589  -rwxr-xr-x 1 root root 423096 may 25 2008 /bin/bash
96589  lrwxrwxrwx 1 root root      4 jun  3 2008 /bin/sh -> bash
```

- Changement de shell
  - En cours de session : exécution par saisie du nom →
  - A la connexion : cf. *chsh*

- Liste des shells disponibles → `/etc/shells` (fichier)

```
$ bsh      /* empilement du bsh */
$ csh      /* empilement du csh */
% ksh      /* empilement du ksh */
$ exit     /* retour au csh */
% exit     /* retour au bsh */
$ exit     /* retour au login shell */
$
```

Caractéristiques des shells – Les *Builtins*

Tous les shells rendent les mêmes services aux utilisateurs. Cependant, ils se différencient par les commandes internes (*builtins*), la syntaxe et les variables prédéfinies.

- Les commandes internes
  - font partie du code du shell (ne peuvent être supprimées)

Commande	Interprétation (*bash*)
`. f.script`	Exécute "`f.script`" dans l'environnement du shell.
`alias`	Crée un alias.
`bg`	Place le processus en arrière-plan.
`builtin`	Exécute la commande interne et non une commande externe du même nom.
`cd`	Change de répertoire.
`echo`	Affiche le texte en argument à l'écran.
`enable`	Active ou désactive les commandes internes du shell.
`exec`	Lance l'exécution de l'argument en remplacement du shell.
`exit`	Termine le shell.
`export`	Exporte une liste de variables.
`fg`	Place le processus en avant-plan.
`help`	Affiche une aide sur les commandes internes.
`history`	Affiche l'historique des commandes.
`kill`	Gère les signaux.

Commande	Interprétation (*bash*)
`let`	Évalue les expressions.
`logout`	Quitte le login shell.
`pwd`	Affiche le répertoire courant.
`read`	Lit l'entrée standard.
`return`	Retourne une valeur au shell.
`set`	Définit les paramètres de l'environnement.
`shift`	Décale l'argument dans la liste des arguments.
`source`	Exécute le script donné en argument dans le contexte du shell.
`times`	Affiche le temps d'exécution cumulé (utilisateur et système).
`trap`	Bloque ou exécute une procédure sur un signal.
`type`	Affiche le type de la commande (alias, fichier, etc).
`typeset`	Affiche ou modifie les attributs des variables.
`umask`	Définit le masque des droits par défaut.
`unalias`	Enlève un alias.
`unset`	Supprime une variable.

Caractéristiques des shells – Les Variables

- Les variables communes du shell

Variable	Interprétation (*bash*)
BASH	Chemin complet du shell.
BASH_VERSION	Numéro de version du shell.
ENV	Chemin du script d'initialisation du shell.
EUID	UID effectif de l'utilisateur.
ERRNO	Code d'erreur de la dernière commande.
HISTFILE	Nom du fichier historique des commandes.
HISTFILESIZE	Taille du fichier historique.
HISTSIZE	Taille du fichier historique en nombre de lignes.
HOME	Chemin du répertoire personnel.
MAIL	Nom du fichier courriel.

Variable	Interprétation (*bash*)
MAILCHECK	Fréquence en secondes de vérification du courrier.
PATH	Liste des répertoires de recherche.
PPID	Numéro du processus père.
PS1	Invite de saisie numéro 1.
PS2	Invite de saisie numéro 2.
PWD	Répertoire courant.
RANDOM	Nombre aléatoire.
SECONDS	Temps écoulé depuis le début du shell.
SHLVL	Nombre d'instances du shell.
TMOUT	Délai d'inactivité en secondes.
UID	Numéro UID de l'utilisateur.

- Contenu des variables
  - le contenu d'une variable peut être affiché en utilisant la commande `echo` suivie du nom de la variable préfixée par `$`.

```
$ echo $PATH
```

Les Fichiers de Démarrage – Cas du `bash`

Contiennent des commandes exécutées systématiquement et automatiquement lors d'une nouvelle session, de la fin d'une session ou d'un nouveau shell.

- A l'ouverture d'une session
  - `/etc/profile` : fichier système. Tous les utilisateurs qui se connectent l'exécutent. Permet de mettre en place un environnement minimal.
  - `[~/.bash_profile, ~/.bash_login]~/.profile` : fichier propre à un utilisateur. Permet de modifier et/ou de compléter l'environnement minimal.

- A l'exécution d'un shell
  - `~/.bashrc` : fichier propre à un utilisateur. Leur rôle est de contenir des syntaxes spécifiques qui ne doivent pas être écrites dans les fichiers précédents.

- A la fermeture d'une session
  - `~/.bash_logout` : fichier propre à un utilisateur. Contient les commandes qui seront exécutées lors de la fermeture de session par l'utilisateur.

Le Changement de Session – *su* et *login*

- La commande *su* permet à un utilisateur de se connecter par-dessus la session actuelle sans déconnecter cette dernière.

```
[alain]$ pwd
/home/alain
[alain]$ su paul
Password: xxxxxx
[paul]$ pwd
/home/alain
[paul]$ exit
[alain]$
```

Conservation de l'environnement de la session précédente.

/* retour à la session d'alain */

```
[alain]$ pwd
/home/alain
[alain]$ su - paul
Password: xxxxxx
[paul]$ pwd
/home/paul
[paul]$ exit
[alain]$
```

Création d'un nouvel environnement. Exécution des fichiers de démarrage.

/* retour à la session d'alain */

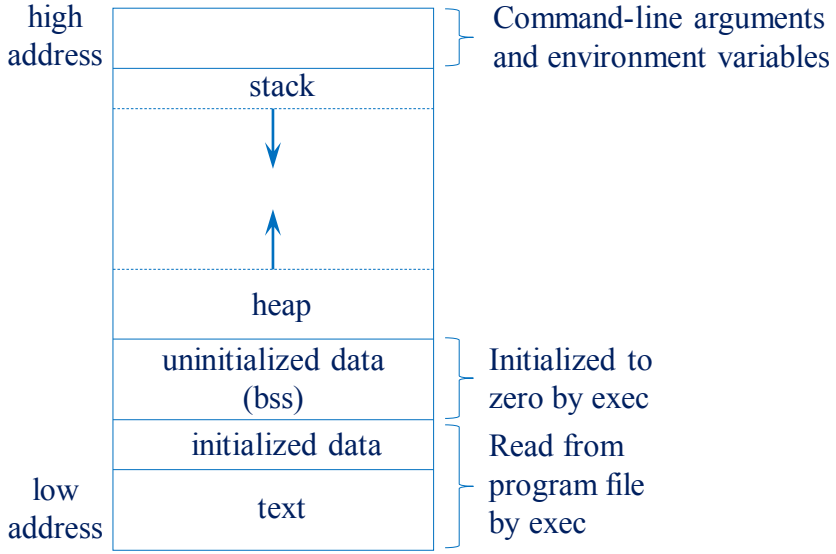
- La commande *login* permet à un utilisateur de se connecter en remplacement de la session actuelle. Cette dernière est définitivement fermée.

```
[alain]$ exec login
login: paul
paul's Password: xxxxxx
Last login:
[paul]$ pwd
/home/paul
```

Notions de Programme et de Processus

- Un programme est une suite d'opérations que l'ordinateur doit exécuter.

- Un processus est une instance d'exécution d'un programme
  - Un processus nécessite des ressources pour s'exécuter (CPU, mémoire, fichiers, E/S, données d'initialisation)
  - La terminaison d'un processus nécessite la récupération des ressources réutilisables



Typical memory arrangement

Types et Modes d'Exécution des Processus

- Types de processus
  - processus système

    Attachés à aucun terminal, ils sont créés par :
    - le noyau : scheduler, pagedaemon, …
    - `init` (`/etc/init`) : démons `lpd`, `ftpd`, ...
  - processus utilisateurs

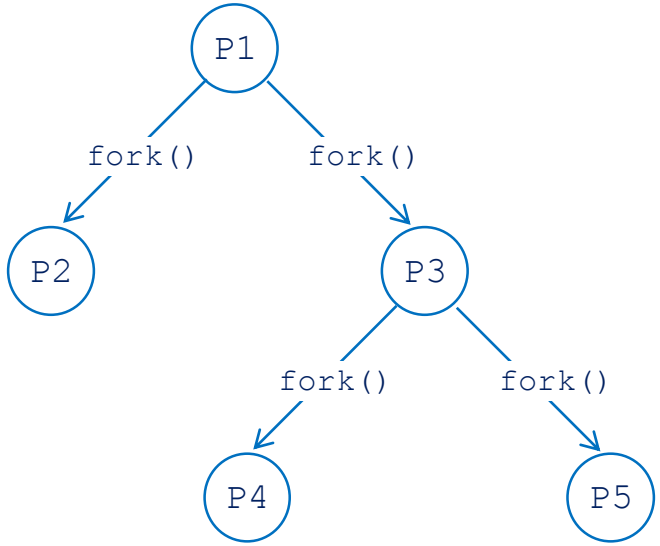
    Lancés par un utilisateur depuis un terminal.

- Modes d'exécution
  - mode utilisateur

    Le processus exécute ses instructions et utilise ses propres données.
  - mode noyau

    Le processus exécute les instructions du noyau.

- Filiation des processus



  - processus `P1` : père des processus `P2` et `P3`
  - processus `P3` : père des processus `P4` et `P5`

Attributs des Processus

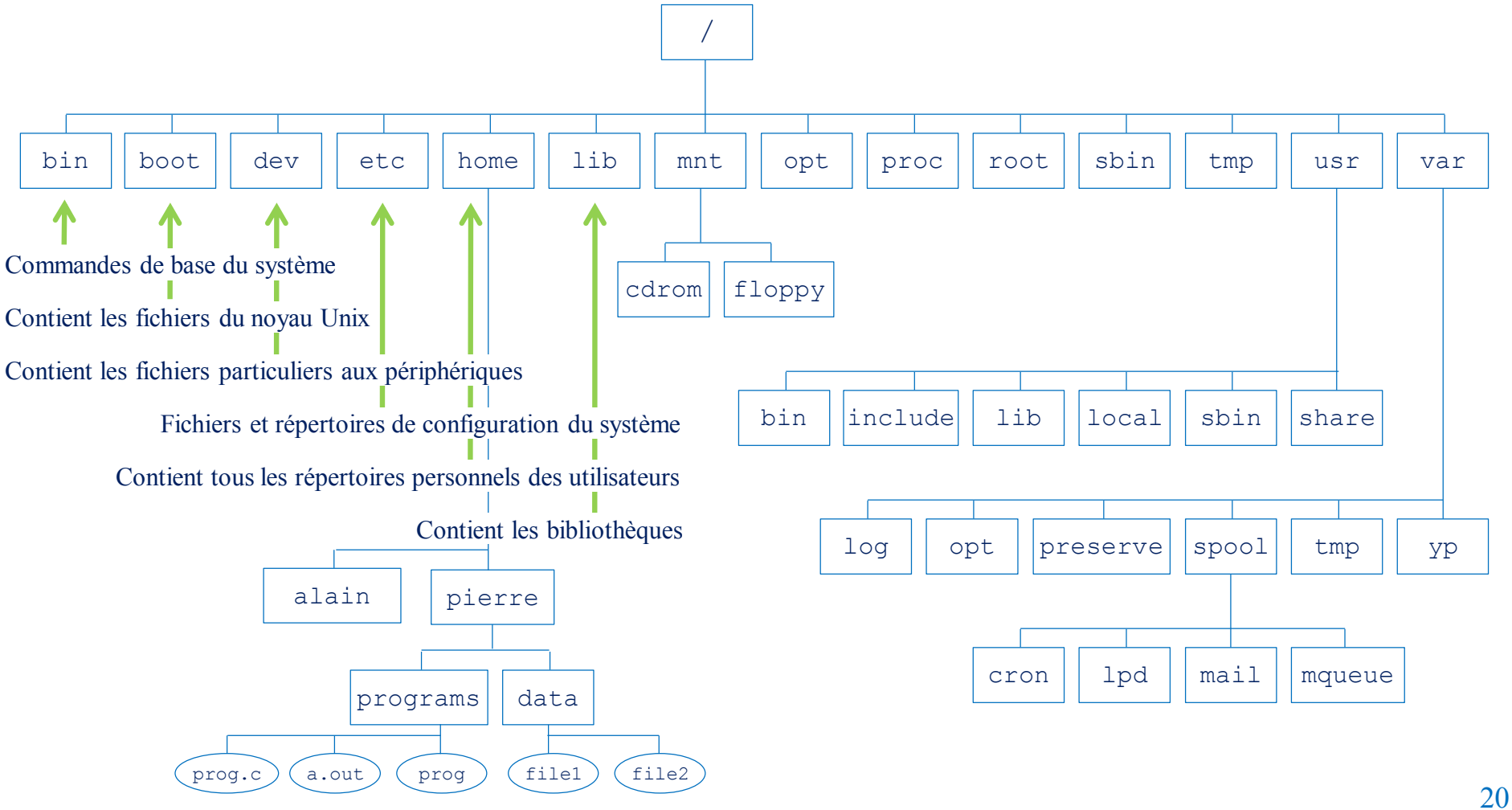
Un certain nombre d'attributs sont associés à un processus. Ils permettent, entre autres, de l'identifier et d'établir ses droits lors de l'exécution. Parmi eux :

- `PID` (*`Process IDentification`*) : numéro unique associé à chaque processus,
- `PPID` (*`Parent Process IDentification`*) : `PID` du processus père,
- `UID` (*`User IDentification`*) : appelé propriétaire réel du processus. Il s'agit de l'`UID` de l'utilisateur qui a lancé (créer) le processus.
- `EUID` (*`Effective User IDentification`*) : appelé propriétaire effectif du processus. Il est en général égal à l'`UID` de l'utilisateur qui a lancé (créer) le processus sauf, éventuellement, si le programme exécutable a son `set-UID` bit positionné.
- `GID` (*`GroupIDentification`*) : appelé groupe propriétaire réel du processus. Il s'agit du `GID` de l'utilisateur qui a lancé (créer) le processus.
- `EGID` (*`Effective GroupIDentification`*) : appelé groupe propriétaire effectif du processus. Il est en général égal au `GID` de l'utilisateur qui a lancé (créer) le processus sauf, éventuellement, si le programme exécutable a son `set-GID` bit positionné.

Les droits d'un processus sont ceux attribués à son `EUID` (éventuellement `EGID`).

Les Fichiers et les Répertoires

- Arborescence du Système de Fichiers (cas d'Unix)



Les Différents Types de Fichiers

- Types de fichiers
  - Fichiers réguliers (`-`)

    fichiers de différents formats pouvant contenir un exécutable en binaire, du texte, de la voix, de l'image, …
  - Répertoires (`d`)

    fichiers système qui impliquent et conservent la structure du système de fichiers
  - Fichiers spéciaux caractères (`c`)

    fichiers associés aux périphériques en mode caractères (écran, imprimante, …)
  - Fichiers spéciaux blocs (`b`)

    fichiers associés aux périphériques en mode blocs (disque, lecteur optique, …)
  - FIFO (`p`)

    fichiers associés aux tubes (pipes) nommés
  - Socket (`s`)

    fichiers associés aux points de communication
  - Liens symboliques (`l`)

    fichiers liés à d'autres fichiers

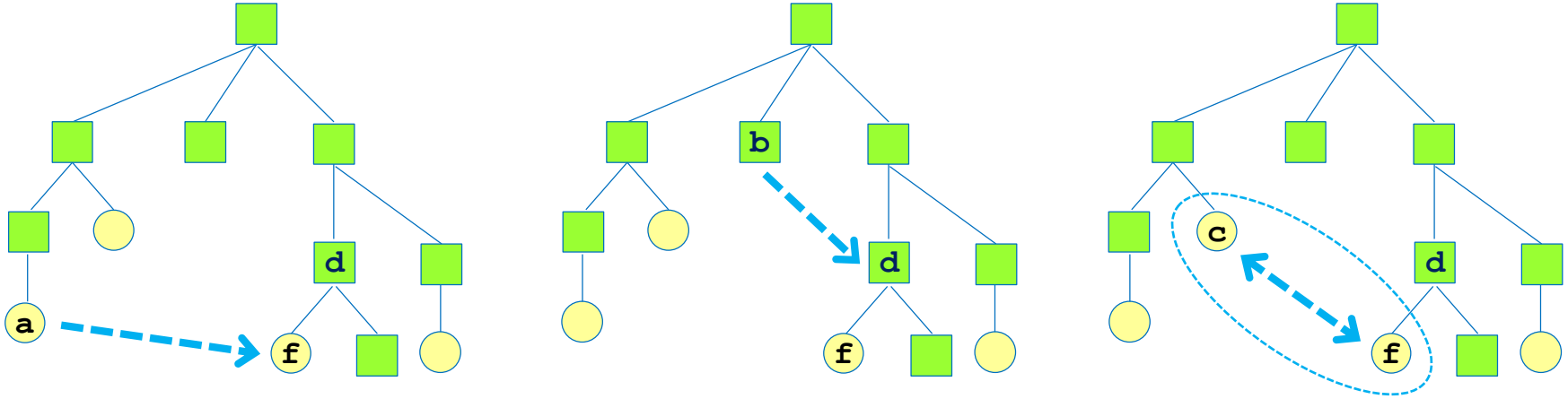
Structure *inode*

- Tous les types de fichiers Unix sont gérés par le SE au moyen des inodes,
- Un inode est une structure de contrôle qui contient les informations clés nécessaires au SE,
  - Plusieurs noms de fichiers peuvent être associés au même inode,
  - Un inode est associé à exactement un seul fichier,
  - Un fichier est contrôlé par exactement un seul inode.

File Mode	16-bit flag that stores access and execution permissions associated with the file		**Link Count**	Number of directory references to this inode
	12-14	File type (regular, directory, character or block special, FIFO pipe)	**Owner ID**	Individual Owner of file
	9-11	Execution flags	**Group ID**	Group owner associated with this file
	8	Owner read permission	**File size**	Number of bytes in file
	7	Owner write permission	**File Adresses**	39 bytes of adress information
	6	Owner execute permission	**Last Accessed**	Time of last file access
	5	Group read permission	**Last Modified**	Time of last file modification
	4	Group write permission	**Inode Modified**	Time of last inode modification
	3	Group execute permission		
	2	Other read permission		
	1	Other write permission		
	0	Other execute permission		

Information in a UNIX Disk-Resident Inode

Les Liens Symboliques et Physiques

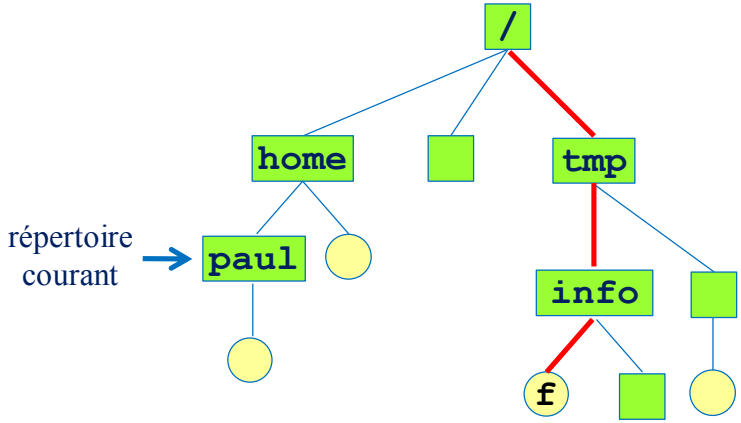


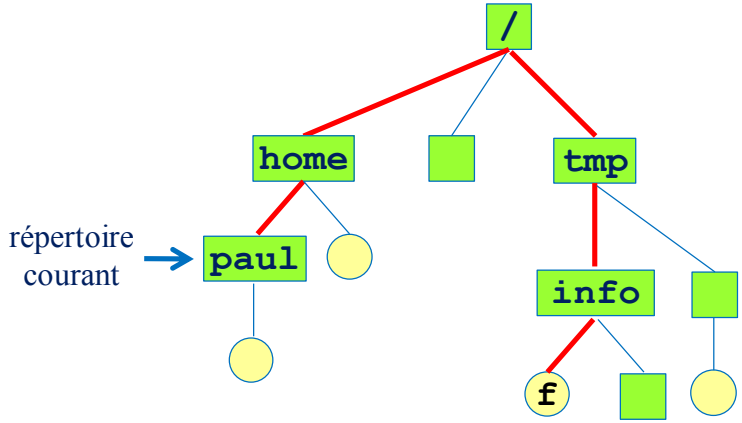
- Les liens symboliques (**a** et **b**)
  - objets contenant l'adresse du fichier ou répertoire qu'ils pointent. Constituent une sorte de redirection.

- Les liens physiques (**c**)
  - correspondent à l'ajout d'un nom à l'objet pointé. Ils ne s'appliquent qu'aux fichiers et ne peuvent exister que dans le même système de fichiers.

```
$ ln -s f a
$ ln -s d b
$ ln f c
$ ls -il
12546556  -rw-rw-rw-  2  paul  paul  2 sep 46 22:10 f
12546572  -rw-rw-rw-  1  paul  paul  2 sep 92 22:12 d
12546602  lrwxrwxrwx  1  paul  paul  2 sep  4 22:15 a->f
12546774  lrwxrwxrwx  1  paul  paul  2 sep  4 22:17 b->d
12546556  -rwxrwxrwx  2  paul  paul  2 sep 46 22:25 c
```

Chemins Absolu et Relatif





- Chemin absolu du fichier **f**
  - chemin qui mène au fichier **f** à partir du sommet de l'arborescence, cad à partir de la racine. Chemin commençant donc toujours par "**/**".
  - chemin toujours valide, quelque soit le répertoire courant.

```
/tmp/info/f
```

- Chemin relatif du fichier **f**
  - chemin qui mène au fichier **f** à partir du répertoire courant.
  - chemin qui change donc en fonction du répertoire courant.

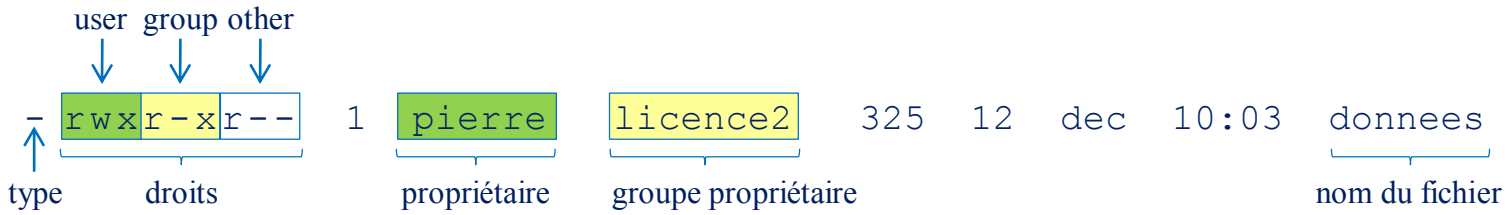
```
../../tmp/info/f
./../../tmp/info/f
```

Remarque :
"**.**" signifie "répertoire courant" et "**..**" signifie "répertoire parent" (pas d'ambiguïté car unique).

Les Droits sur les Fichiers et Répertoires

- Attributs d'un fichier
  - l'`UID` de son propriétaire,
  - le `GID` de son groupe propriétaire,
  - les droits pour les différentes catégories d'utilisateurs.
- Les catégories d'utilisateurs
  - le propriétaire (***u**ser*),
  - les membres du groupe propriétaire (***g**roup*),
  - les autres (***o**ther*), c'est à dire ceux qui ne sont ni propriétaire ni membres du groupe.
- Droits attribués pour chaque catégorie
  - lecture (*read*, ***r***),
  - écriture (*write*, ***w***),
  - exécution (*execute*, ***x***).

  Le symbole "**-**" indique l'absence de droit. Les droits sont donnés dans l'ordre **rwx** pour chaque catégorie (*user*, *group*, *other*).



Interprétation des Droits

Droits	Fichier	Répertoire
Lecture (**r**)	Permet de voir le contenu du fichier.	Permet de voir le contenu du répertoire, cad la liste des fichiers et sous-répertoires qu'il contient.
Écriture (**w**)	Permet de modifier le contenu du fichier.	Permet d'ajouter ou de supprimer des fichiers (ou sous-répertoire) au répertoire. [1]
Exécution (**x**)	Permet d'exécuter le fichier s'il s'agit d'un binaire ou d'un script shell (dans ce cas le droit **r** est aussi nécessaire).	Permet de se déplacer dans un répertoire ou de le traverser. [2]

[1] un utilisateur ayant le droit **w** sur un répertoire pourra supprimer les fichiers qu'il contient même s'ils ne lui appartiennent pas. Utilisation du `sticky-bit` pour limiter cette suppression aux seuls propriétaires des fichiers ou du répertoire.

[2] afin d'atteindre un fichier quelconque, il faut que l'utilisateur ait les droits **x** sur tous les répertoires qu'il faut traverser pour l'atteindre.

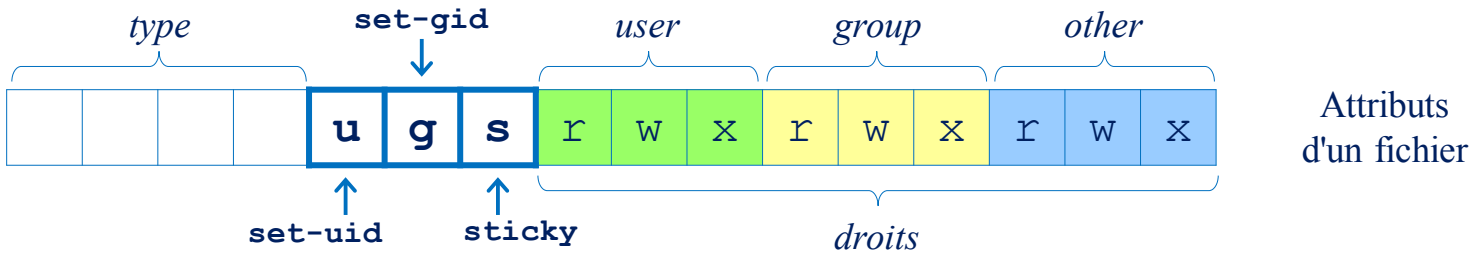
Test des Droits d'Accès

Chaque fois qu'un processus ouvre, crée ou supprime un fichier, le noyau procède aux tests suivants :

1. si ID user effectif (`EUID`) du processus est 0 (super user) l'accès est autorisé,
2. si ID user effectif (`EUID`) du processus est égal à ID propriétaire (`UID`) du fichier :
   a. si le mode d'accès demandé est compatible aux droits attribués au propriétaire (*`user`*), l'accès est autorisé,
   b. sinon l'accès est refusé,
3. si ID groupe effectif (`EGID`) du processus, ou l'un des IDs groupes supplémentaires du processus, est égal à ID groupe (`GID`) du fichier :
   a. si le mode d'accès demandé est compatible aux droits attribués aux membres du groupe (*`group`*), l'accès est autorisé,
   b. sinon l'accès est refusé,
4. si le mode d'accès demandé est compatible aux droits attribués aux autres utilisateurs (*`other`*), l'accès est autorisé, sinon l'accès est refusé.

Si un utilisateur est propriétaire d'un fichier, alors on lui applique uniquement les droits correspondant au propriétaire. Si ces derniers ne sont pas suffisants pour la tâche souhaitée, on ne regarde pas s'il a plus de droits dans les autres catégories.

Les Droits d'Endossement



Les droits d'endossement permettent de changer l'identité d'un utilisateur lors de l'exécution d'un programme.

- Le *Set-UID* bit (*Set User ID bit*)

  Appliqué à un exécutable, il permet à un utilisateur (ayant le droit) d'exécuter ce dernier avec l'identité de son propriétaire. Durant l'exécution, l'EUID du processus est égal au propriétaire de l'exécutable. L'EGID reste celui de l'utilisateur. Ce droit est représenté par un "**s**" en lieu et place du droit **x** de la catégorie *user*.

```
$ -r-s--x--x  1 root      root   13254 jun 25  2008 /usr/bin/passwd
$ -rw-r--r--  1 root      root    2540 sep  5 08:16 /etc/passwd
$ ----------  1 root      root     786 sep  5 08:17 /etc/shadow
```

- Le *Set-GID* bit (*Set Group ID bit*)

  Appliqué à un exécutable, il permet à un utilisateur (ayant le droit) d'exécuter ce dernier avec un identifiant de groupe égale à son groupe propriétaire. Durant l'exécution, l'EGID du processus est égal au groupe propriétaire de l'exécutable. L'EUID reste celui de l'utilisateur. Ce droit est représenté par un "**s**" en lieu et place du droit **x** de la catégorie *group*.

```
$ -rwx--s--x  1 root      root   12184 jun 25  2008 /usr/local/bin/prog
```

Les Droits d'Endossement 2

Pour bénéficier du droit **s** (*Set-UID* ou *Set-GID* bit), il faut nécessairement avoir en même temps le droit **x** sur la catégorie correspondante.

- Affichage, avec la commande `ls -l`, d'un exécutable selon les droits positionnés :

	Set-UID bit non positionné	*Set-UID* bit positionné
Droit **x** pour *user* non positionné	`rw-r-xr-x`	`rwSr-xr-x`
Droit **x** pour *user* positionné	`rwxr-xr-x`	`rwsr-xr-x`

	Set-GID bit non positionné	*Set-GID* bit positionné
Droit **x** pour *group* non positionné	`rwxr--r-x`	`rwxr-Sr-x`
Droit **x** pour *group* positionné	`rwxr-xr-x`	`rwxr-sr-x`

- Le *Set-GID* bit appliqué à un fichier non exécutable

  Lorsque le `Set-GID` bit est appliqué à un fichier ordinaire (données), cela indique aux système que le verrouillage éventuel du fichier doit être de type impératif et non pas consultatif.

- Le *Set-GID* bit appliqué à un répertoire

  Le système BSD a défini un nouveau rôle au `Set-GID` bit lorsqu'il est appliqué à un répertoire (repris par Linux). Dans ce cas, les fichiers qui y sont créés appartiennent au groupe propriétaire du répertoire, et non à celui de l'utilisateur.

Le `sticky` bit

A l'origine, le *`sticky`* bit était appliqué à un exécutable, et il servait à maintenir ce dernier en mémoire de façon à accélérer ses exécutions ultérieures. Fonctionnement délaissé depuis.

- Le *`sticky`* bit appliqué à un répertoire

  Lorsqu'il est appliqué à un répertoire, le *`sticky`* bit permet de contrôler la suppression des fichiers qu'il contient, en limitant cette action au propriétaire du fichier.

  Souvent appliqué aux répertoires publics, tel le répertoire `/tmp`, où tout le monde peut écrire (créer/déposer/supprimer des fichiers) mais chacun ne pouvant supprimer que ses propres fichiers.

  Lorsqu'il est appliqué, le *`sticky`* bit est représenté par un "**`t`**" en lieu et place du droit **`x`** de la catégorie *`other`*.

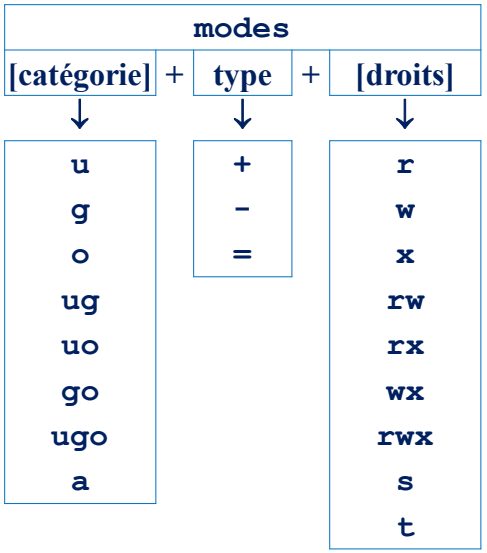
```
$ ls -ld /tmp
drwxrwxrwx  2 root root   5236 apr 15 21:15 /tmp
$ ls -l /tmp
-rw-r—r--   1 paul paul    128 sep  2 22:18 /donnees
            /* tout le monde peut supprimer le fichier donnees contenu dans /tmp */

$ ls -ld /tmp
drwxrwxrwt  2 root root   5236 apr 15 21:15 /tmp
$ ls -l /tmp
-rw-r—r--   1 paul paul    128 sep  2 22:18 /donnees
            /* seuls paul (propriétaire du fichier) et root (propriétaire du répertoire) peuvent supprimer le
                  fichier donnees contenu dans /tmp */
```

Gestion des Droits

La modification des droits se fait par la commande

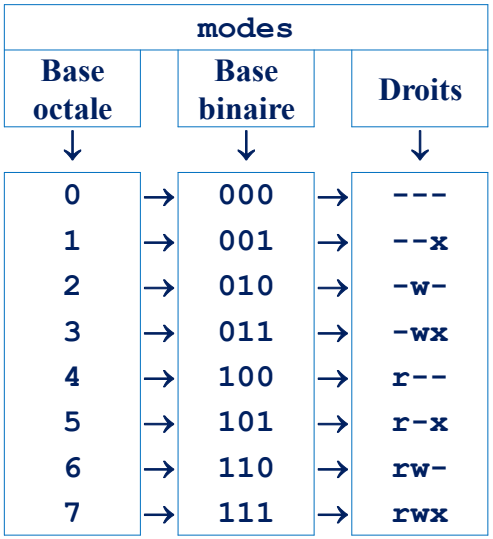
```
chmod -options modes fichier
```

- Notation symbolique




```
$ chmod u+x prog
$ chmod ug+w,o-w fichier
$ chmod -R go-w Exercices
$ chmod +x prog
$ chmod o= test
```

- Notation octale




```
$ chmod 755 prog
$ chmod -R 644 Exercices
$ chmod 4755 prog
```

Codage octale	Interprétation	Equivaut à
4000	Set-UID bit	u=s
2000	Set-GID bit	g=s
1000	Sticky bit	o=t
0400	read pour le propriétaire	u=r
0200	write pour le propriétaire	u=w
0100	execute pour le propriétaire	u=x
0040	Read pour le groupe propriétaire	g=r
0020	write pour le groupe propriétaire	g=w
0010	execute pour le groupe propriétaire	g=x
0004	read pour les autres	o=r
0002	write pour les autres	o=w
0001	excute pour les autres	o=x

La notation octale modifie toujours les 3 catégories d'utilisateurs en même temps.

Gestion des Droits par Défaut

Lorsqu'un fichier ou un répertoire est créé par un processus, ce dernier spécifie quels seront les droits souhaités pour ce fichier ou répertoire.

Au cours d'une session, il existe un filtre qui mémorise les droits devant être interdits, pour chaque catégorie d'utilisateurs, lors de la création d'un fichier ou répertoire.

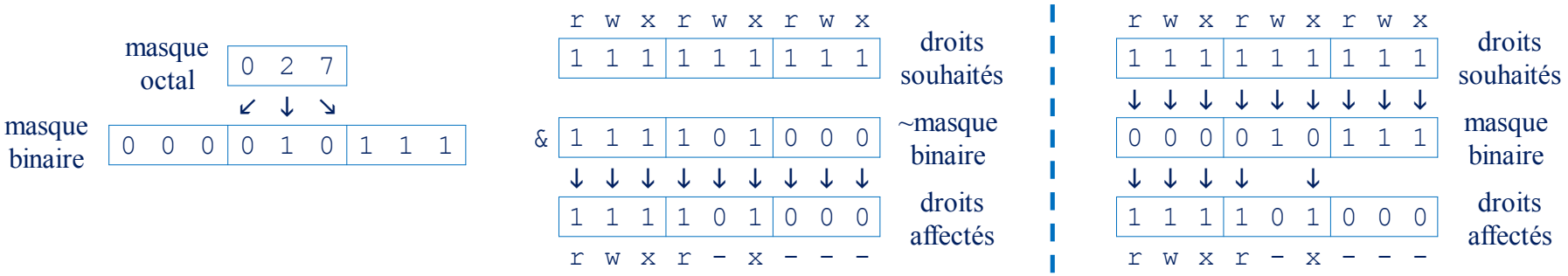
Ce filtre est donné, et peut être modifié, par la commande `umask` :

`umask -options mode`

```
$ umask                          $ umask -S
022                              u=rwx,g=rx,o=rx
$ umask 027                      $ umask u=rwx,g=rx,o=
$ umask                          $ umask
027                              027
```

Les droits finalement affectés au fichier ou répertoire nouvellement créé sont définis par :

`droits affectés = (droits souhaités) & ~(masque de filtrage)`



Commandes de Base sur les Fichiers et Répertoires

Action	Fichier	Répertoire
Afficher le contenu	`cat` ou `more`	`ls`
Copier	`cp`	`cp -R` ou `cp -r`
Changer le nom	`mv`	`mv`
Se déplacer dans		`cd`
Créer		`mkdir`
Détruire	`rm`	`rmdir` ou `rm -r`
Déplacer	`mv`	`mv`
Afficher le répertoire courant		`pwd`

Remarque : le système UNIX ne prévoit aucune commande de création de fichier. Cependant, il est possible de le faire en détournant certaines commandes de leur fonction initiale. Ainsi, les deux commandes suivantes créent le fichier `donnees.txt` :

```
$ cp /dev/null donnees.txt
$ touch donnees.txt
```

Autres Commandes

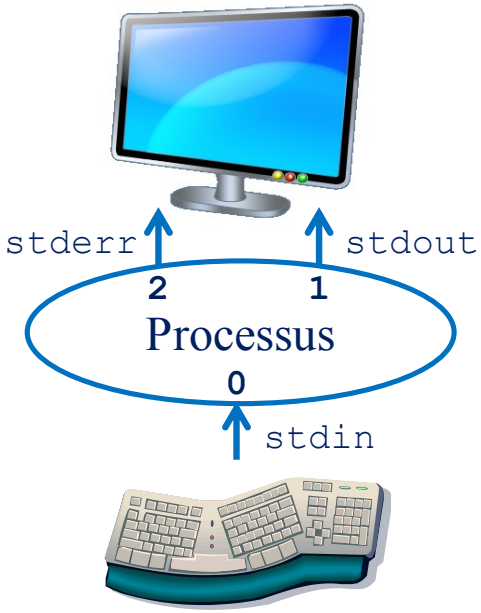
Commande	Interprétation
`file`	Détermine le type d'un fichier (text, executable, data).
`touch`	Modifie la date d'accès et la date de modification du fichier spécifié.
`find`	Recherche des fichiers ou répertoires selon certains critères.
`diff`	Trouve les différences entre des fichiers.
`du`	Donne des statistiques sur l'utilisation du disque.
`df`	Indique les espaces disques utilisés et disponibles des systèmes de fichiers.

Commande	Interprétation
`grep`	Recherche les lignes de fichiers correspondant à un motif.
`sort`	Affiche la concaténation triée de fichiers textes.
`cut`	Affiche des portions spécifiques de lignes de fichiers
`wc`	Affiche le nombre de lignes, de mots ou de caractères des fichiers.

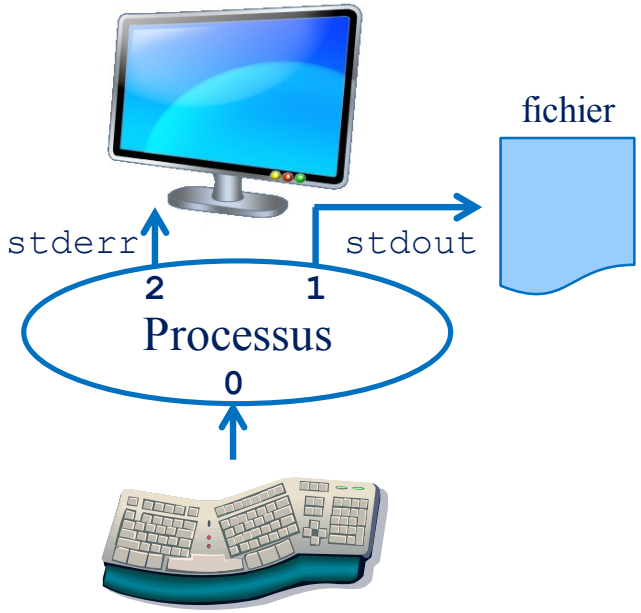
Les Entrée/Sorties Standards

Lorsqu'un processus est créé, trois fichiers spéciaux sont ouverts automatiquement :

- Entrée standard (`stdin`) : représentée par le descripteur `0`, fait référence par défaut au clavier
- Sortie standard (`stdout`) : représentée par le descripteur `1`, fait référence par défaut à l'écran
- Sortie d'erreurs standard (`stderr`) : représentée par le descripteur `2`, fait référence par défaut à l'écran



Les entrée/sorties standards peuvent être redirigées afin que la lecture/écriture se fasse à partir/vers d'autres fichiers.



Les Redirections

Redirection	
`>`	Rediriger (écrire) vers un fichier
`>>`	Rediriger (écrire) vers la fin d'un fichier
`<`	Lire depuis un fichier
`<<`	Lire progressivement depuis le clavier
`2>`	Rediriger la sortie d'erreurs standard vers un fichier.
`2>>`	Rediriger la sortie d'erreurs standard vers la fin d'un fichier.
`2>&1`	Fusionne les sorties standards (rediriger la sortie d'erreurs standard au même endroit et de la même manière que la sortie standard).
`tube '|'`	Redirection (`pipe`). Connecte la sortie d'une commande à l'entrée d'une autre commande.

Exemples de Redirections

Problème : compter le nombre de mots uniques dans un fichier.

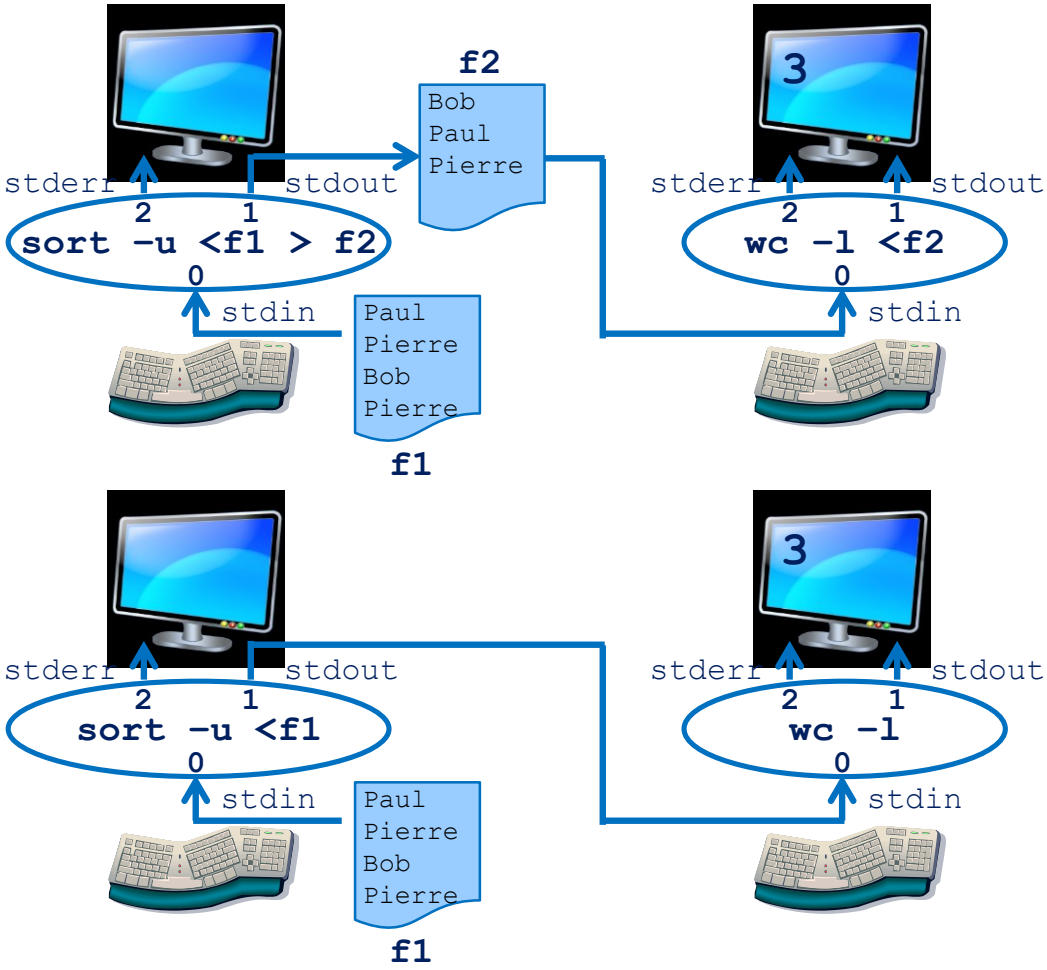
Solution 1 :

Par défaut, `sort` lit les lignes introduites au clavier et les affiche triées à l'écran.

```
sort -u <f1 >f2
wc -l <f2
```

Solution 2 :

```
sort -u <f1 | wc -l
```



Flot de données entre processus sans création de fichier intermédiaire.

Les Expressions Régulières

- Une expression régulière (Regular Expression) décrit un motif qui sert à identifier une chaîne de caractères répondant à un certain critère.

- Les expressions régulières sont couramment utilisées dans les systèmes d'exploitation pour :
  - la recherche de fichiers dans une arborescence,
  - la recherche et la modification de sous-chaines de caractères dans des fichiers.

- De nombreux utilitaires (appelés filtres) utilisent les expressions régulières pour accomplir leur tâche, parmi eux `grep`, `sed`, `awk` …

Exemples

```
$ ls *.jpg                    /* liste tous les fichiers du répertoire courant ayant pour
                                 extension ".jpg"                                    */

$ grep 'sys[0-9]' donnees     /* affiche toutes les lignes du fichier donnees contenant la sous-
                                 chaine "sys" suivie d'un chiffre                    */
```

Les Métacaractères

Les métacaractères sont des caractères spéciaux qui servent d'opérateurs aux expressions régulières.

Métacaractère	Interprétation
`.`	Remplace dans une expression régulière un caractère unique, à l'exception du caractère retour chariot (`\n`).
`?`	Caractère de répétition (zéro ou une occurrence du motif précédent).
`*`	Caractère de répétition (zéro ou plusieurs occurrences du motif précédent).
`+`	Caractère de répétition (une ou plusieurs occurrences du motif précédent).
`{ }`	Permettent la répétition du motif qui précède (`{n}`, `{n,}`, `{n,m}`)
`( )`	Délimite un composant (atome) associé à une partie de l'expression régulière.
`[ ]`	Permettent de désigner des ensembles de caractères.
`-`	Permet de désigner des caractères compris dans un intervalle.
`^`	Identifie un début de ligne.
`$`	Identifie une fin de ligne.
`|`	Exprime l'alternative.
`\`	"Déspécialise" un caractère spécial.

Exemples d'Expressions Régulières

ER atomiques	Signification	Exemples de motifs	Séquences atomiques correspondant au motif
Un caractère non spécial quelconque	Se décrit lui-même	`a`	a
Un caractère spécial précédé de '\'	Décrit ce même caractère	`\$`	$
Le caractère spécial '.'	Décrit un caractère quelconque	`.`	a, b, c, …, A, …, Z, ^, <tab>, …
Suite de caractères entre '[' et ']'	Décrit un ensemble de caractères	`[ab09]`	a, b, 0, 9
Deux caractères séparés par '-' et compris entre '[' et ']'	Décrit tous les caractères dont le code ascii est dans l'intervalle indiqué	`[a-d]`	a, b, c, d
ER étendues	**Signification**	**Exemples de motifs**	**Séquences atomiques correspondant au motif**
Concaténation d'ER	Décrit toutes les concaténations de séquences décrites par la 1ère ER et la 2ème ER	`[a-z][0-9]`	a0, a1, …, b0, b1, …, z0, …, z9
Alternative d'ER séparées par '|'	Décrit toutes les séquences décrites par la 1ère ER ou la 2ème ER	`ab|cd`	ab, cd
Une ER entre '(' et ')'	Décrit le même ensemble de mots que l'ER à l'intérieure des parenthèses, mais permet de marquer les frontières d'un motif	`(ab)|(cd)`	ab, cd
Une ER suivie de '?'	Décrit **zéro ou une** occurrence du motif précédent	`((ab)|(cd))?`	séquence vide, ab, cd
Une ER suivie de '*'	Décrit **zéro ou plusieurs** occurrences du motif précédent	`a(b|c)*`	a, ab, ac, abb, abc, acb, acc, …
Une ER suivie de '+'	Décrit **une ou plusieurs** occurrences du motif précédent	`(ab)+`	ab, abab, ababab, …

Exemples d'Expressions Régulières suite

ER étendues	Signification	Exemples de motifs	Séquences atomiques correspondant au motif
Une ER suivie de "{n}"	Décrit **n** occurrences du motif précédent	`(ab){2}`	abab
Une ER suivie de "{n,}"	Décrit **n** occurrences ou plus du motif précédent	`(ab){2,}`	abab, ababab, abababab, …
Une ER suivie de "{n, m}"	Décrit entre **n** et **m** occurrences du motif précédent	`(ab){1, 3}`	ab, abab, ababab
Une ER suivie de '$'	Décrit les mêmes séquences que celles décrites par l'ER mais seulement si elles apparaissent en fin de ligne	`ab$`	toute ligne se terminant par ab
Une ER précédée de '^'	Décrit les mêmes séquences que celles décrites par l'ER mais seulement si elles apparaissent en début de ligne	`^ab`	toute ligne commençant par ab

Bibliographie

- Unix & Linux, Utilisation et administration
  Jean-Michel Léry, Pearson Education

- The magic garden explained, The internals of Unix System V Release 4
  Berny Goodheart & James Cox - Prentice Hall

- Operating Systems, Internals and Design Principles,
  Willam Stallings - Printice Hall

- Principes des Systèmes d'Exploitation,
  A. Silberschatz, P.B. Galvin, G. Gagne - Vuibert

- Introduction aux Système Unix, Support de cours,
  Eric Gressier - CNAM-CEDRIC