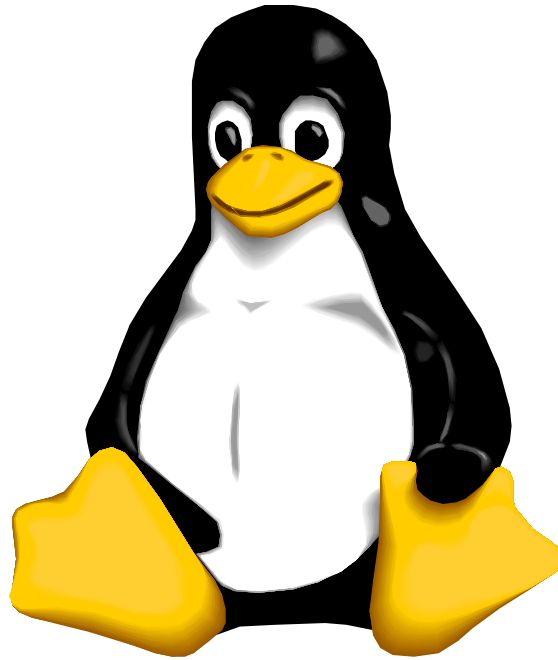


Unix/Linux Utilisateur

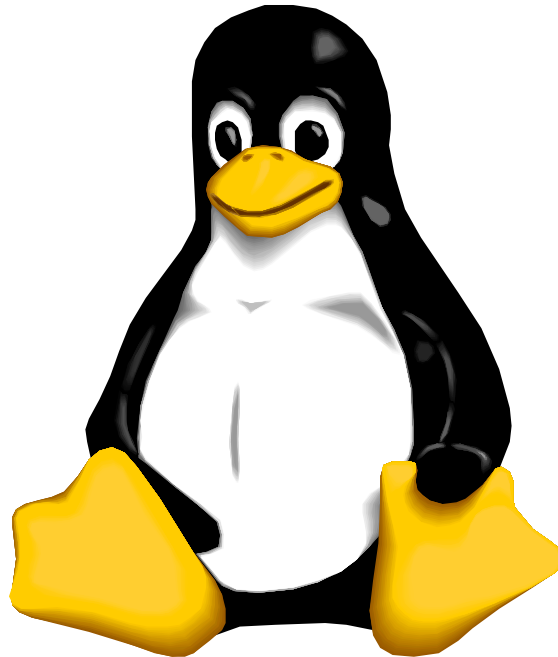


AVANT PROPOS

- UNIX est un système d'exploitation complet intégrant un grand nombre de fonctionnalités.
- Qui utilise les machines UNIX ?
- **L'utilisateur**
- **Le développeur**
- **L'administrateur**

Unix/Linux Utilisateur

Introduction

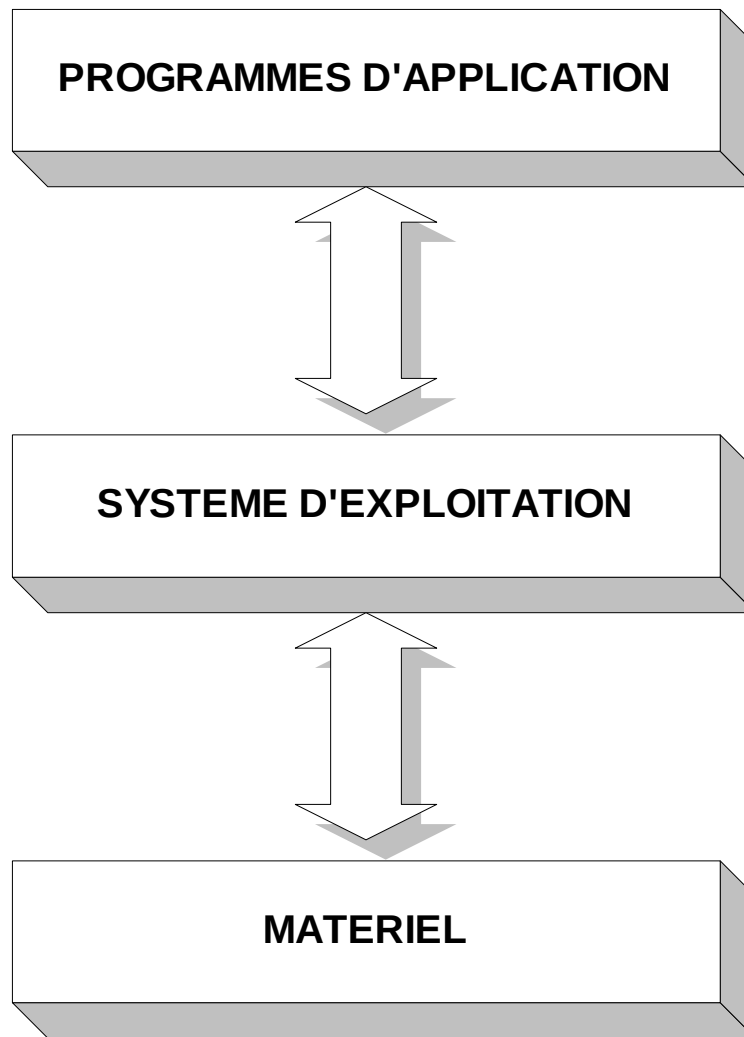


Système d'exploitation

- Un système d'exploitation est, pris dans son sens le plus large, un ensemble de programmes et de sous-programmes dont le rôle est de gérer, de piloter le matériel. C'est l'intermédiaire entre les programmes d'application et le matériel, c'est lui qui intercepte les demandes des applications et les transmet au matériel.

Rôle d'un système d'exploitation

- **Gestion de la mémoire**
- **Accès aux périphériques**
- **Gestion des disques**
- **Gestion des programmes**
- **Protection des données**
- **Collecte de statistiques**

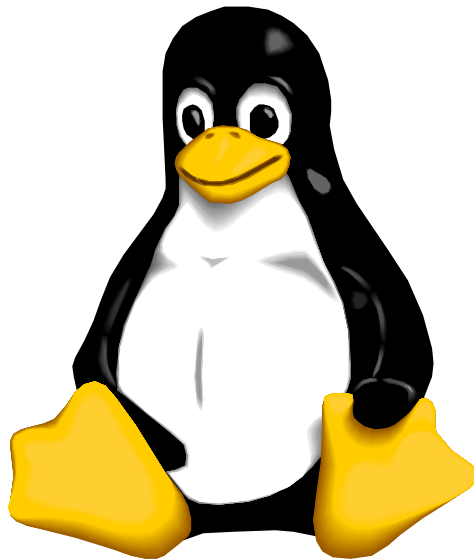


**Système d'exploitation vu comme interface
entre matériel et logiciel**

Afpa - Angers - Linux Utilisateurs

Unix/Linux Utilisateur

Historique





- **En 1964 :**

General Electric, le MIT et les Bell Labs d'AT&T s'associent dans le projet MULTICS (Multiplexed Information and Computing Service). Le but du projet était de créer un système d'exploitation pour ordinateur parfaitement fiable, capable de tourner 24H sur 24, 7 jours sur 7, utilisable par plusieurs personnes à la fois et capable en même temps de faire tourner des calculs en tâche de fond.

- **En 1969 :**

Ken Thompson, l'un des collaborateurs du labo, ayant récupéré un dec PDP 7 inutilisé, écrit alors un jeu, « *Space Travel* » (« voyage spatial »), d'abord sur Multics, puis il se mit à réfléchir à la création d'un nouveau système d'exploitation temps partagé .

Il l'écrit en collaboration avec Dennis Ritchie. Ce système d'exploitation avait pour nom UNICS (Uniplexed Information and Computing Service), il s'inspirait de MULTICS : notion de process, système de fichiers arborescent, interpréteur ligne de commande tournant comme un simple programme utilisateur, représentation simple des fichiers texte et accès généralisé aux périphériques.

- **En 1969 (suite):**

Unics fut rebaptisé UNIX (par Brian Kernighan), un shell, quelques programmes utilitaires, un éditeur et un assembleur furent rapidement mis au point sur le PDP 7. Cette version est connue sous le nom "Unix Time-Sharing System Version 1".



Caractéristiques techniques du **PDP-1**

Processeur 18 bits

Mémoire : 4096 mots de 18 bits

Entrée sortie de données : écran / clavier / cartes perforées

Options : stylo optique, écran haute résolution, bandes magnétiques

Logiciels fournis : Editeur, Assembleur et Débugueur

Prix : 120000 \$

Le PDP 1 et son panneau de contrôle

Ken Thomson
et Dennis
Ritchie en
plein travail !





Ken Thompson

- **En 1970 :**

Ken Thompson, pensant qu'UNIX ne serait pas complet sans un langage de programmation de haut niveau commence à porter le Fortran sur le PDP 7 mais change rapidement d'avis et crée en fait un nouveau langage, le B, un langage interprété (en référence au BCPL dont il s'inspire).

- **En 1971 :**



Dennis Ritchie

Portage du tout nouveau système d'exploitation UNIX sur PDP 11/20. Ken Thompson et Dennis Ritchie ont obtenu cette machine en prétextant le développement d'un logiciel de traitement de textes, les responsables du Bell Labs ne voulant plus entendre parler de systèmes d'exploitation suite à l'abandon du projet MULTICS.

Sur cette machine disposant de 24 Ko de mémoire, le noyau Unix occupait 16 Ko, 8 Ko restant disponibles pour les utilisateurs. Le disque dur avait une taille de 512 Ko et les fichiers une taille maximale de 64 Ko.

En 1971 (suite):

Le formateur de texte roff fût porté sur cette machine, ce qui permit à trois personnes du service des brevets d'utiliser effectivement la machine comme traitement de textes et ce en même temps que Thompson et Ritchie qui continuaient le développement d'applications.

Le succès de cette expérience a prouvé l'utilité d'Unix et a rendu possible la poursuite du développement sur PDP 11/45. Cette version est connue sous le nom "Unix Time-Sharing System Version 2".

- **De 1971 à 1973 :**



B. Kernighan

- Dennis Ritchie et Brian Kernighan du Bell Labs d'AT&T reprennent le langage B écrit par Ken Thompson pour PDP/7 en 1970 pour mieux l'adapter au PDP/11 sur lequel UNIX vient juste d'être porté. Ils font évoluer le langage et le dote d'un vrai compilateur.
- C'est le langage C qui est à la fois proche du matériel, permettant ainsi de réécrire le noyau UNIX en C et suffisamment généraliste, le rendant ainsi facilement portable. Les développements et les succès du langage C et d'UNIX sont intimement liés.

- **En 1973 :**
- Le noyau du système d'exploitation UNIX est entièrement réécrit en langage C par Ken Thompson. Au vu de la qualité du résultat, tous les autres outils utilisés sous Unix vont être réécrits en C. Cette version est connue sous le nom "**Unix Time-Sharing System Version 4**".
- En raison d'une injonction du ministère de la justice qui n'a pas permis à AT&T de vendre le logiciel, le code source d'Unix est distribué librement aux universités. En conséquence, UNIX a gagné la faveur de la communauté scientifique et universitaire. Il a été ainsi à la base des systèmes d'exploitation des principales universités, dont l'Université de Californie à Berkeley.

- **En 1976 :**
- Sortie de "Unix Time-Sharing System Version 6" aux Bell Labs d'AT&T, AT&T peut enfin commercialiser le logiciel.
- En marge, Les laboratoires Bell d'AT&T développent UUCP (Unix to Unix Copy Program). Il s'agit du premier protocole d'échanges de données largement disponible et qui sera énormément utilisé avant l'avènement de TCP/IP et d'Internet.

En 1977 :

- Première expérience de portage d'UNIX sur un autre type d'ordinateur, l'Interdata 8/32, par Ken Thompson, Dennis Ritchie et Steve Johnson.
- A cette date, environ 600 machines tournent déjà sous UNIX. Beaucoup sont des machines de recherche et de développement aux Bell Labs. Une partie est utilisée pour contrôler des autocommutateurs téléphoniques. Une dernière partie est utilisée dans les universités pour l'enseignement.

- **En 1978 :**

L'université de Berkeley produit ses propres versions d'UNIX appelées **BSD** (Berkeley Software Distribution) :

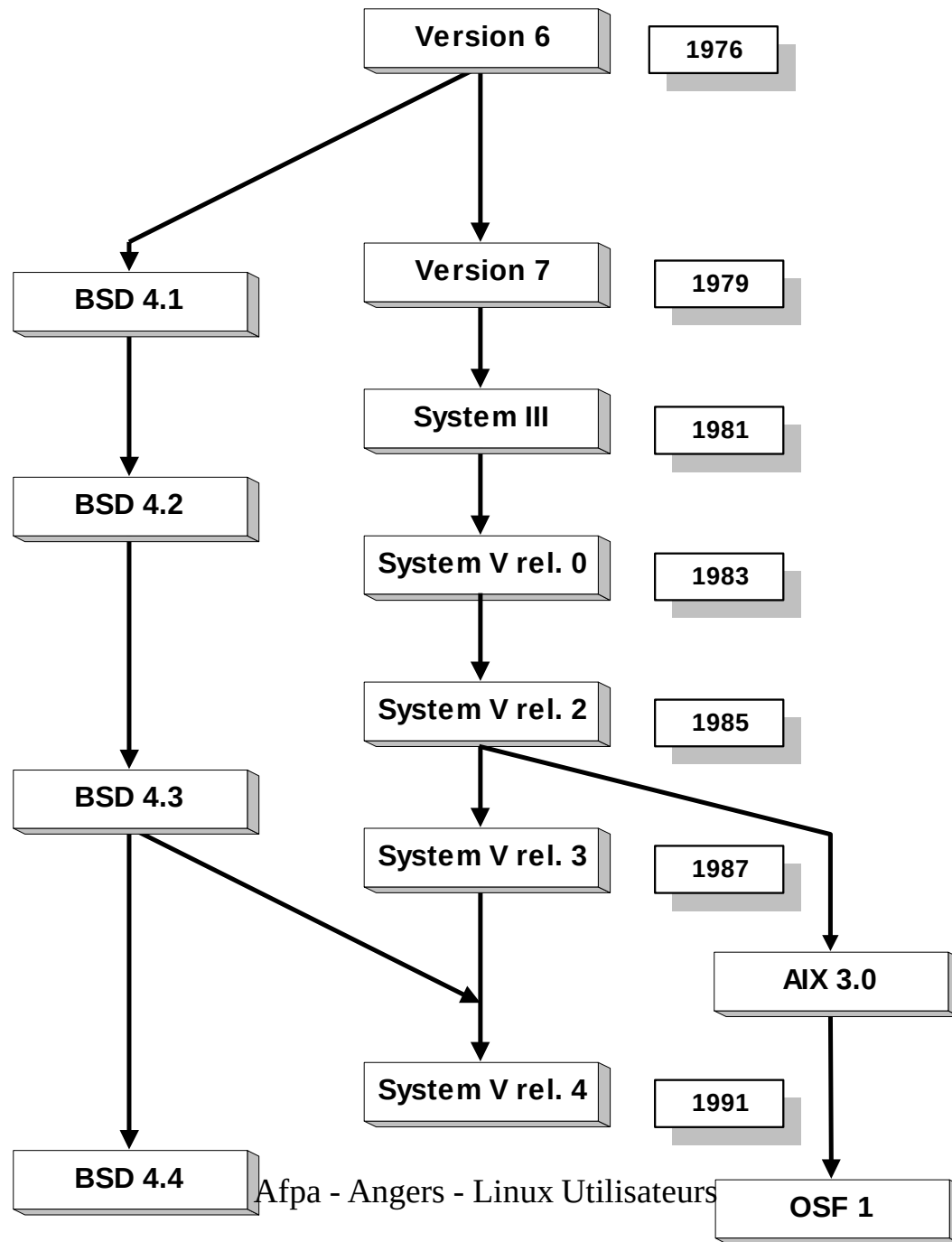
Apparition de 1BSD une nouvelle distribution d'UNIX réalisée à l'université de Berkeley principalement par Bill Joy (l'actuel n°2 de Sun Microsystems) et basée sur les sources d'Unix Time-Sharing System V6.

En 1979 :

- Sortie de "Unix Time-Sharing System V7" aux Bell Labs d'AT&T. Cette version est la première à disposer, en standard d'UUCP.

Elle tournait sur PDP/11 et VAX.

Histoire D'UNIX



Le projet GNU, la Free Software Foundation (FSF)

- Fin 1983, Richard Stallman, alors chercheur au laboratoire d'intelligence artificielle du MIT aux États-Unis, lance le [projet GNU](#), en réaction à la pratique alors émergente qui consistait à garder secret le code source des logiciels et à contraindre au paiement d'une licence pour leur utilisation.





Le projet GNU, la Free Software Foundation (FSF)

- Son objectif est de développer un système d'exploitation complet, semblable à Unix et qui soit un logiciel libre : le système **GNU**. (« GNU » est l'acronyme récursif the « GNU's Not Unix »)
- Afin de récolter des fonds, il crée la **Free Software Foundation**, et commence à développer de nombreux logiciels.

LINUX



- **En 1991 :**
- Un étudiant finlandais, nommé Linus Torvalds, a acheté un micro-ordinateur de type PC, afin d'étudier, la programmation du microprocesseur i386 (ancêtre du pentium). Ne voulant pas être limité par MS/DOS, il a tout d'abord utilisé Minix un clone d'Unix pour x86. Conscient des limitations de Minix, il décide de lui apporter quelques améliorations. Chemin faisant, il propose à certains internautes de le rejoindre dans son projet.
- Pour cela il lance le mail qui suit sur la Mailing-List de Minix (en Août 1991) :

Vous regrettez les beaux jours de Minix-1.1, lorsque les hommes étaient des hommes et écrivaient leurs propres pilotes de périphériques ? Vous manquez d'un superbe projet et vous languissez après un système que vous pourriez modifier à votre convenance ? Vous êtes frustrés que tout fonctionne sous Minix ? Plus de nuits blanches passées à tenter de faire fonctionner un programme récalcitrant ? Alors ce message pourrait bien être pour vous.

Comme signalé il y a un mois, je travaille actuellement sur une version libre et gratuite d'un système ressemblant à Minix pour les ordinateurs AT-386. J'ai finalement atteint un stade où il est utilisable (bien qu'il puisse ne pas l'être pour vous, selon ce que vous désirez), et je compte diffuser les sources pour une diffusion plus large. Il s'agit juste de la version 0.02... mais j'ai pu exécuter bash, gcc, gnu-make, gnu-sed, compress, etc. avec succès sous ce système.

Linux

- Août 1991
- C'est le départ de L'aventure linux, qui vous mène aujourd'hui ici. De nombreux développeurs de part le monde se joignent à lui et Linux 0.01 est disponible en Septembre de la même année.

Linux

- Au sens strict, Linux n'est pas un Unix puisqu'il ne comprend pas de code provenant de l'original. D'ailleurs, Linux à proprement parler n'est que le *noyau*, le cœur du système d'exploitation. Le système d'exploitation est **GNU/Linux**, le noyau plus les outils basiques fournis par le projet **GNU** (pour *Gnu's Not Unix* : « Gnu N'est pas Unix ») de la *Free Software Foundation*.

Linux

- Néanmoins, GNU/Linux a en commun avec Unix une bonne part de son fonctionnement et de son comportement. Pour un utilisateur lambda, il est bien difficile de faire la différence entre un Linux et un Unix. Pour éviter toute pinaillerie, certains parlent d'« Unixoides » (« *Unix-like* » en anglais) pour évoquer la grande famille des Unix et Linux.

Définitions

- **logiciel libre et logiciel propriétaire.**

Un logiciel *libre*, selon la définition de la *Free Software Foundation*, garantit aux utilisateurs 4 libertés fondamentales :

1. la liberté d'exécuter le programme, pour tous les usages ;
2. la liberté d'étudier le fonctionnement du programme, et de l'adapter à ses besoins ;
3. la liberté de redistribuer des copies ;
4. la liberté d'améliorer le programme et de publier ses améliorations.

Un logiciel qui ne garantit pas ces 4 libertés est dit *propriétaire*.

Polémique

- « UNIX » ou « Unix » ? Comme c'est une marque déposée, on peut dire que la bonne manière de l'écrire est celle des détenteurs de la marque, l'Open Group. Celui-ci écrit « UNIX® ». On peut néanmoins admettre que le mot est devenu un nom commun, et que ça s'écrit « Unix ». Faites votre choix !

Qu'est-ce que Linux ?

- Linux™ est un système UNIX 32 bits multi-tâches, Multi-Utilisateurs, sachant gérer le mode multiprocesseurs
- Développé par Linus Torvalds à l'origine pour plateforme Intel x86, il a été porté sur de nombreuses architecture : Alpha, Sparc, PowerPC, ...
- Il est compatible avec le standard POSIX 1003.1 et il inclut un bon nombre des fonctionnalités d'UNIX System V et BSD 4.3
- Il est développé selon les termes de la **GNU General Public License** et ses sources sont librement distribuables

Autres distributions majeures de Linux



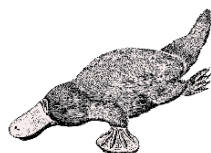
Ubuntu

- dernière version : 7.10
- page d'accueil : www.ubuntufr.org
- site FTP :



Debian

- dernière version : 3.1
- page d'accueil : www.debian.org
- site FTP : www.debian.org/distrib/ftplist



Slackware

- dernière version : 12
- page d'accueil : www.slackware.com
- site FTP : ftp.cdrom.com/pub/linux/slackware



Suse

- dernière version : 10.3
- page d'accueil : www.suse.com
- site FTP : ftp.suse.com

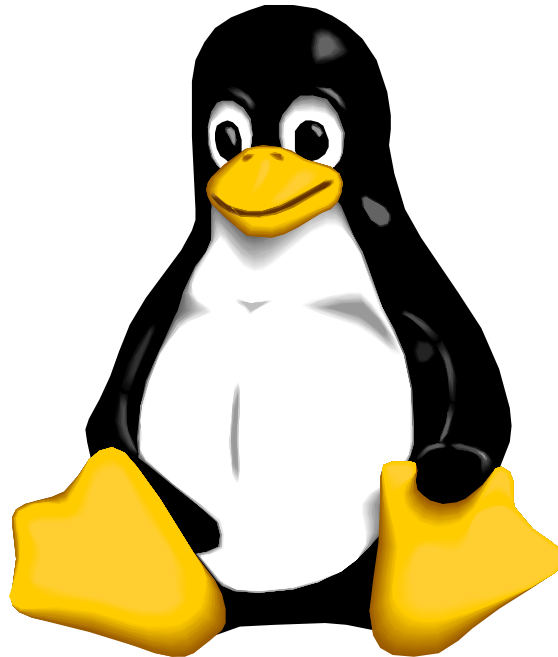


Redhat /Fedora

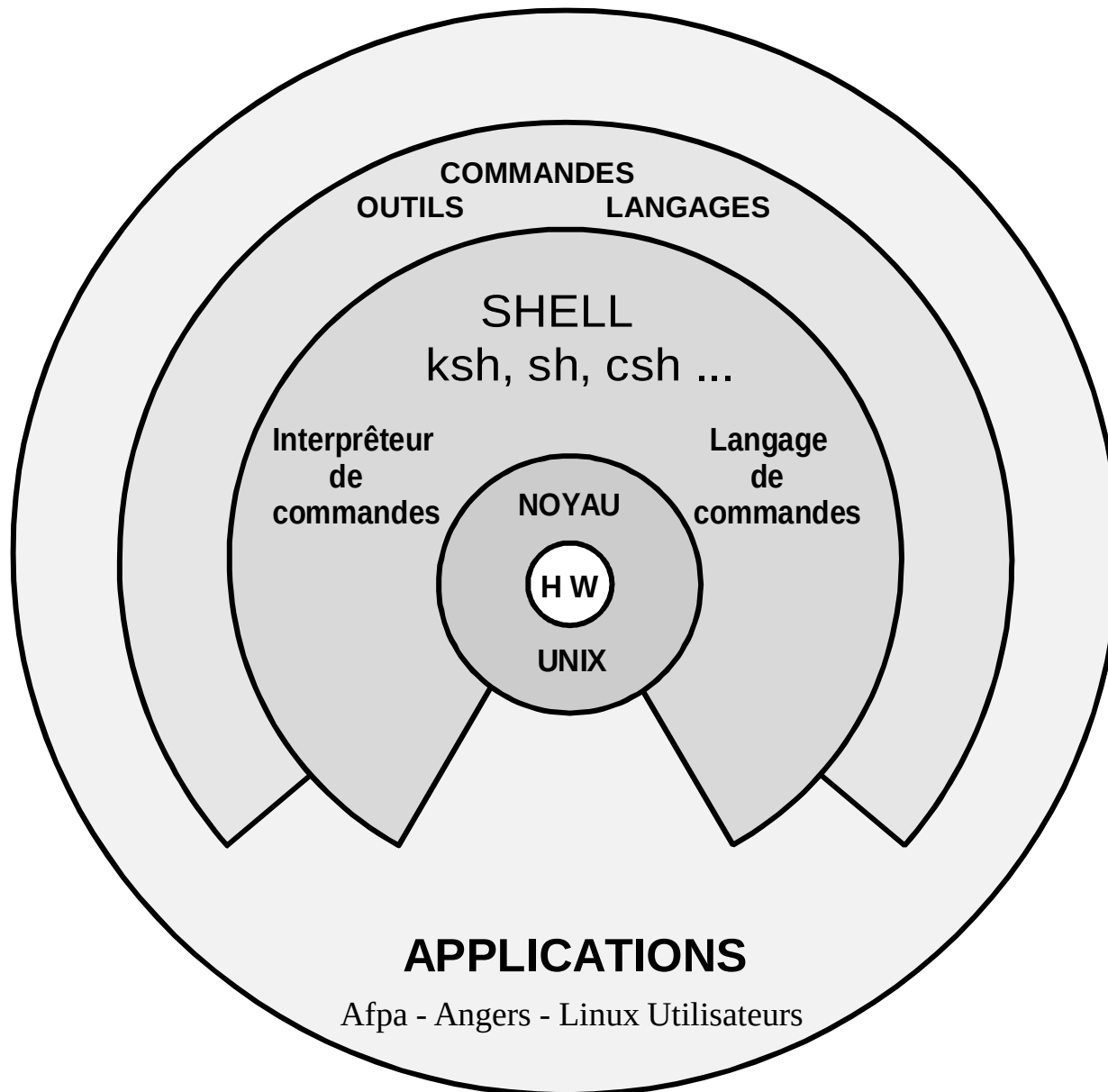
- dernière version : 9.0
- page d'accueil : www..fr.redhat.com

Unix/Linux Utilisateur

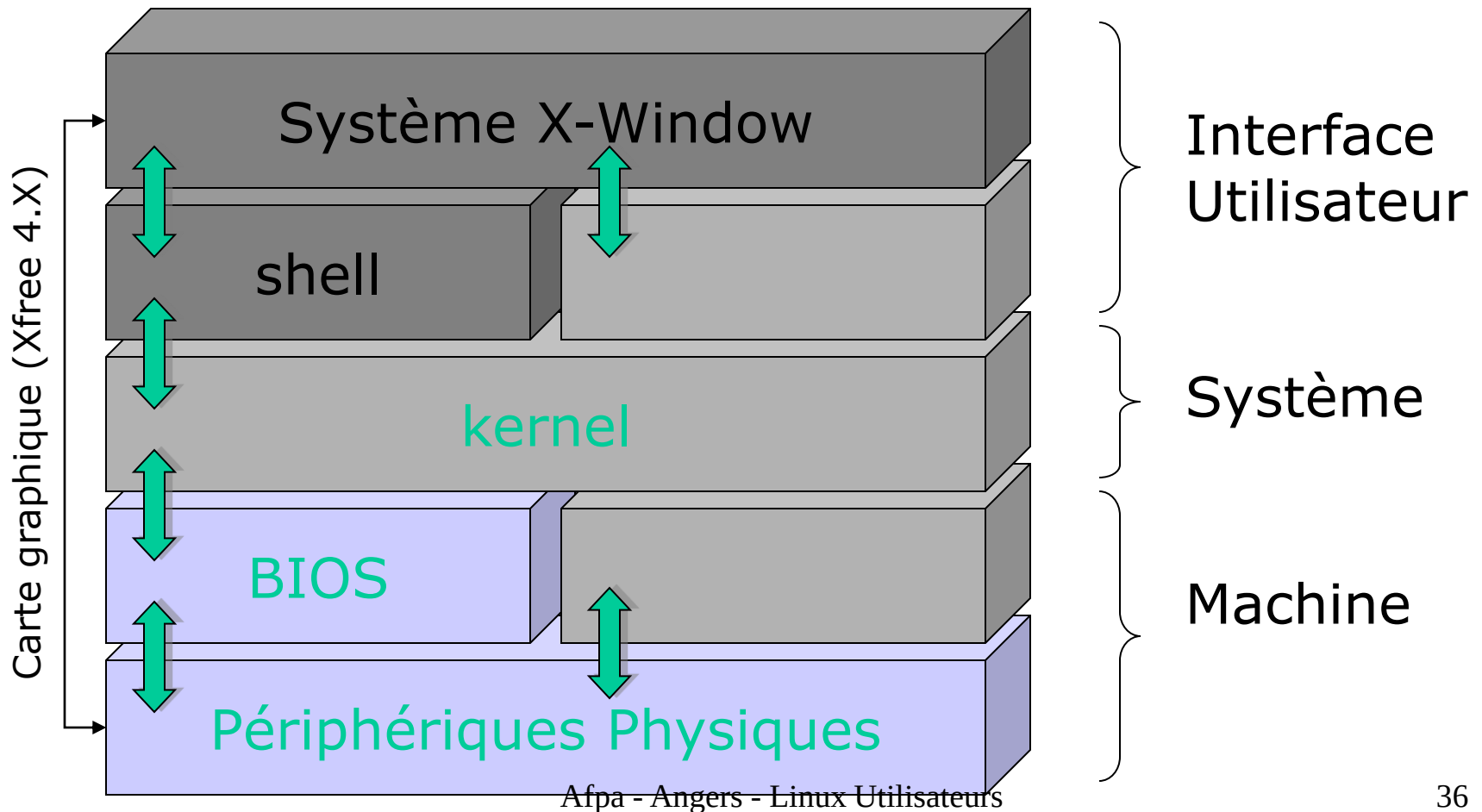
Structure de l'O.S. UNIX



Le modèle en couches UNIX



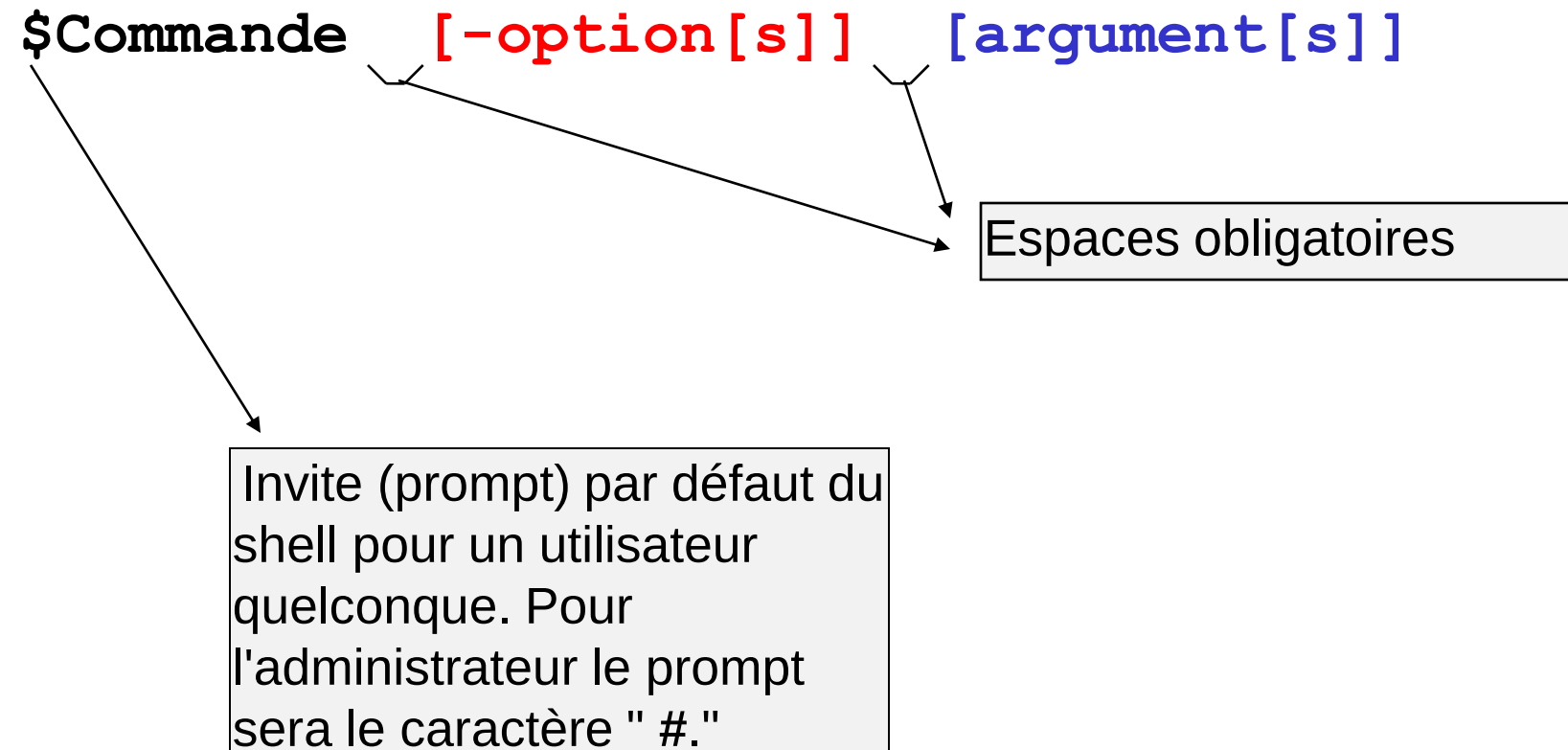
Architecture d'un système GNU/Linux:



Les interpréteurs de commande

- Le ***Bourne shell*** est le premier shell fourni avec UNIX
- Le ***Korn shell*** développé à BERKELEY, offre une syntaxe proche de celle du langage C.
- Le ***C shell***
- Le ***Bourne Again SHell***, spécifique à Linux et compatible Bourne Shell est disponible sur Linux

Syntaxe générale des commandes UNIX



Aide sur les commandes:

man

- La commande ***man*** permet de consulter l'aide en ligne pour les commandes du système ainsi que la définition de certains fichiers.
 - Obtenir de l'aide sur la commande ***ls***
- \$man ls***
- Pour quitter ***man*** taper ***q*** .
 - Aide simplifiée sous Linux, pour une commande:

\$« commande » -- help

Quelques commandes UNIX

COMMANDE	ACTION
passwd	Permet de changer le mot de passe. Deux possibilités sont offertes: • L'utilisateur définit lui même son mot de passe. • Le système génère automatiquement un mot de passe.
who	Renvoie la liste des utilisateurs connectés sur la machine.
who am i	Affiche le nom de l'utilisateur qui a lancé la commande.
date	Affiche la date et l'heure.
cal	Affiche le calendrier
calendar	Gestion d'agenda sur certains systèmes.
finger	Similaire à la commande who mais fournit des informations plus complètes.
banner	Affiche un texte en grands caractères.
exit	Termine la session de travail.

Terminer une session de travail

- Pour mettre fin à une session de travail il suffit d'entrer la commande suivante:

\$exit

Quelques secondes après réapparaît le message de connexion:

Login:

- Il est aussi possible de mettre fin à la session, si cette possibilité est validée sur le système, en frappant la combinaison de touches suivante sur la ligne de commande:

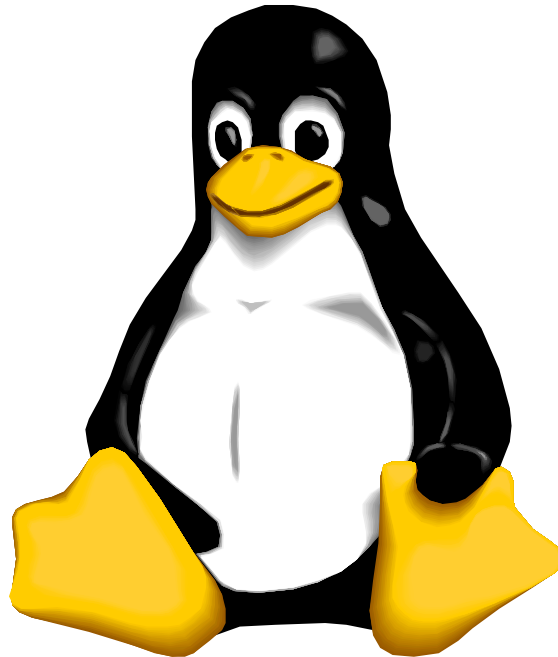
^D (Contrôle D)

- Qu'il s'agisse d'un terminal, d'un PC en émulation de terminal ou d'un PC sous UNIX, il ne suffit pas d'éteindre la machine. UNIX n'interprète pas cela comme une fin de session.

Vous devez d'abord quitter la session de travail !!!

Linux Utilisateur

Le système de fichiers



Le système de fichiers

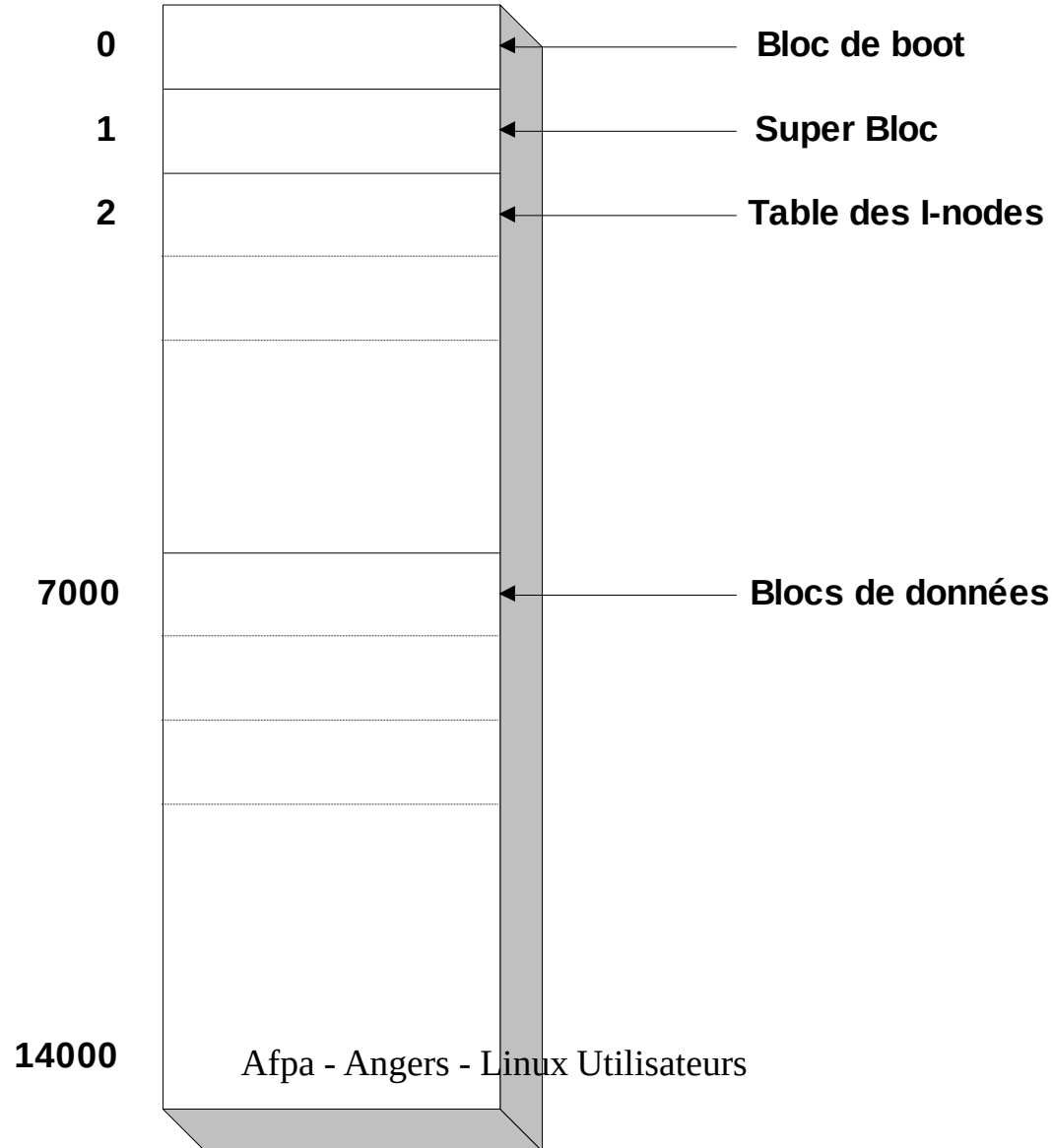
- Regroupe les méthodes et les structures de données qu'un système d'exploitation utilise pour gérer les fichiers sur un disque ou une partition.
- Il ne faut pas confondre **système d'exploitation** et **système de fichiers**
- Les systèmes de fichiers sont généralement associés à un système d'exploitation (Windows 9x avec la FAT, Linux avec ext2) mais le système d'exploitation peut être capable de reconnaître plusieurs systèmes de fichiers

Les principaux systèmes de fichiers reconnus par Linux

- **minix** : le plus ancien, mais limité dans ses caractéristiques
- **xia** : une version modifiée de minix. Très répandu, fonctionne très bien.
- **ext/ext2** : le plus riche des systèmes de fichiers Linux et le plus répandu (système de fichiers par défaut).
- **ext/ext3** : le successeur de Ext2FS. Il conserve une compatibilité avec celui-ci mais ajoute une fonctionnalité très intéressante: la journalisation.
- **ReiserFS** : également journalisé comme Ext2FS, mais sa structure est radicalement différente.
- **msdos/vfat** : compatibilité avec les systèmes de fichiers FAT de MSDOS (d'OS/2 et de Windows NT).
- **nfs** : système de fichiers permettant le partage des données entre plusieurs ordinateurs en réseau.
- **hpfs** : système de fichiers d'OS/2.

Composantes d'un système de fichiers

N° de bloc



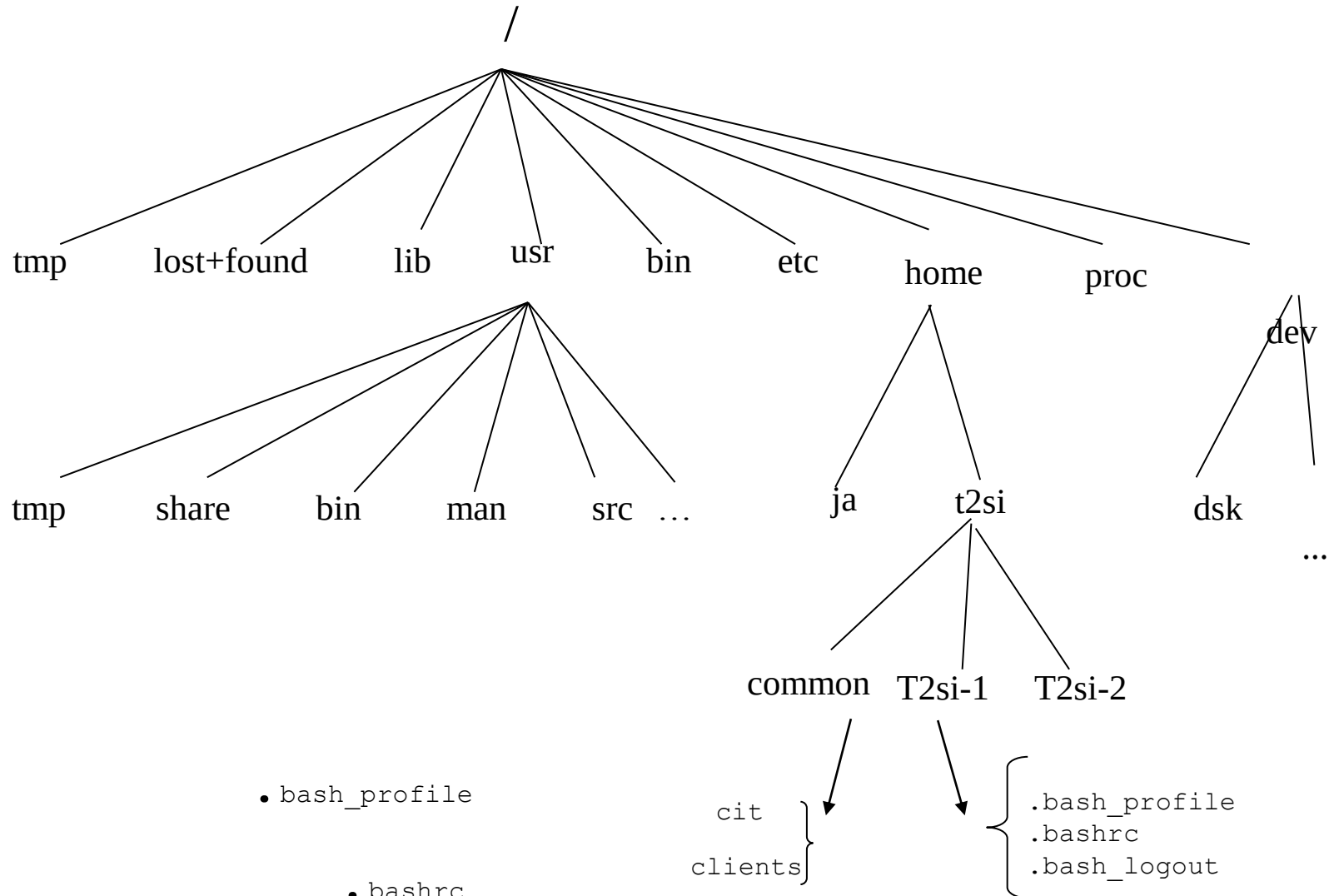
TOUT est fichier :

- un disque dur, une partition sur ce disque dur, un port parallèle, une connexion à un site Web , une carte Ethernet, même les répertoires sont des fichiers pour Linux.
- un fichier, qu'il soit une image, un fichier binaire ou autre, n'est fondamentalement qu'une quantité d'octets. La différenciation des fichiers en fonction de leur contenu est du ressort des applications.

L'arborescence du système de fichiers Linux

- L'arborescence Linux est un ensemble de systèmes de fichiers raccordés les uns aux autres
- La racine (unique) est le « / » et se dénomme **root**
- On peut distinguer plusieurs catégories de fichiers :
 - **partageables** : pouvant être partagé entre plusieurs machines différentes (répertoires des utilisateurs)
 - **non partageables** : spécifiques à une machine particulière (fichiers de blocage de périphérique, les *locks*)
 - **variables** : tout ce qui change sans l'intervention de l'administrateur système
 - **statiques** : tout ce qui ne change pas sans l'intervention de l'administrateur système (les binaires, les bibliothèques, la documentation ...)

L'arborescence



Les répertoires principaux de Linux (1)

- Répertoire racine **/** : contient l'ensemble des répertoires
/boot : contient les fichiers nécessaires au démarrage du système, notamment le noyau `vmlinuz` ou `vmlinuz`
- **/bin** : contient les commandes majeures souvent utilisées (`ls`, `mkdir`, ...). Les autres commandes sont dans `/usr/bin`
- **/dev** : regroupe les fichiers de périphérique ; ce sont des fichiers spéciaux qui constituent des interfaces avec les pilotes de périphérique (ils n'ont pas de contenu)
- **/etc** : contient les fichiers de configuration du système

Les répertoires principaux de Linux (2)

- **/home** : est destiné à recevoir les répertoires personnel des utilisateurs
- **/lib** : regroupent les bibliothèques partagées (notamment la *libc6* sur laquelle reposent la majorité des programmes)
- **/proc** : c'est un **pseudo-système** de fichiers ; les fichiers qui se trouvent sous /proc pointent vers des programmes en mémoire vive qui lisent des informations système centrales ; ils n'occupent pas de place sur le disque
- **/root** : contient le répertoire personnel de l'administrateur (/root figure dans le système de fichier racine)

Les répertoires principaux de Linux (3)

- **/sbin** : contient les commandes d'administration
- **/tmp** : regroupent les fichiers temporaires (créés pendant l'utilisation de certains programme, comme les éditeurs)
- **/var** : est l'emplacement des données variables (travaux d'impressions en attente, courrier électronique reçu ...)
- **/usr** : contient le reste des données et programmes du système, le tout regroupé dans une sous-arborescence :
 - /usr/include : fichiers d'entête C ou C++
 - /usr/doc : fichiers de documentation de Linux
 - /usr/src : source du noyau
 - /usr/X11R6 : répertoire racine de toutes les données de l'environnement graphique X Windows
 - ...

Les fichiers

Pour Linux, Ext3 FS, gère les noms de fichiers longs. Il utilise des entrées de répertoire de taille variable. La taille maximale d'un nom de fichier est de 255 caractères. Tout caractère est autorisé sauf le /

- **Les majuscules et les minuscules sont différenciées!**
- Les fichiers bonjour, Bonjour et BONJOUR sont considérés par UNIX comme trois fichiers différents.
- Le caractère . (point) utilisé en 1er caractère du nom d'un fichier à une signification spéciale: il désigne un fichier caché.

Les types de fichiers

```
$ ll wellcome
-rw-r--r-- 1 chris chris 284 wellcome
```

- - Fichier ordinaire
 - d Fichier répertoire
 - l Lien symbolique ou physique
 - b Périphérique bloc
 - c Périphérique caractère
- } Fichiers spéciaux associés aux dispositifs d'ES physiques

Les commandes associées au système de fichiers

- Création d'un répertoire.

\$ **mkdir** nomderekp<cr>

- Suppression de répertoire

\$ **rmdir** nomderekp <cr>

Ceci n'est possible que si le répertoire est vide

\$ **rm -Rf** nomderekp (récuratif)

Les commandes associées au système de fichiers

Gestion des répertoires

- Répertoire racine: « / »
- Affichage du répertoire courant:
\$ **pwd**
- Changer de répertoire:
\$ **cd** *répertoire*
- Répertoires spéciaux:
 - « **.** »: répertoire courant
 - « **..** »: répertoire parent
 - « **~** »: répertoire personnel de l'utilisateur
 - « **-** »: répertoire précédent

Les commandes associées au système de fichiers

- **Gestion des Fichiers**

- Lire le contenu d'un fichier:

- \$ **cat** fichier

- \$ **vi** fichier

- \$ **less** fichier

- \$ **more** fichier

- Créer un lien symbolique

- \$ **ln** -s source lien

Les commandes associées au système de fichiers

- **Suppression d'un fichier**

\$ rm nomdefic <cr>

Options

- i interactif la commande rm demande confirmation
- f forcer suppression du fichier sans message
- r récursif si le nom de fichier est un répertoire, rm détruit la totalité des fichiers du répertoire et le répertoire lui-même

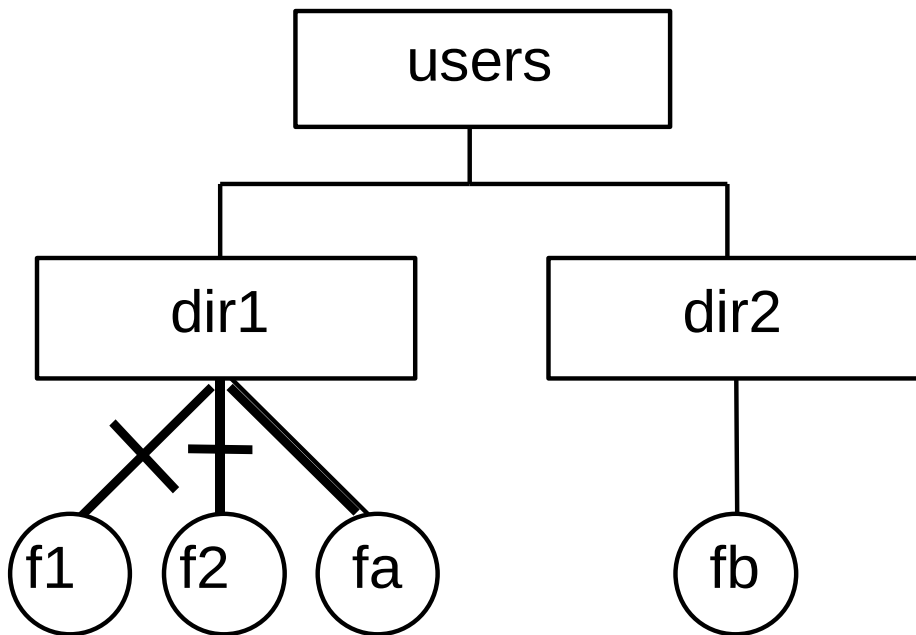
- **Copie d'un fichier**

\$ cp origine destination <cr>

Les commandes associées au système de fichiers

- Déplacer, renommer un fichier

\$ mv origine destination <cr>(move)



\$ mv f1 fa<cr>

(changement de nom)

\$ mv f2 ../dir2/fb<cr>

(déplacement et changement de nom)

Expressions régulières

Caractères génériques pour la génération de fichiers

***** correspond à n'importe quelle chaîne de caractères, y compris la chaîne nulle

Seule elle signifie liste des fichiers et répertoires du répertoire courant

? correspond à n'importe quel caractère, mais un seul.

[abnx] n'importe quel caractère appartenant à l'ensemble défini entre crochets.

[a-z] n'importe quel caractère compris entre a et z

Expressions régulières

- Il existe un caractère spécial `\` qui permet d'annuler la signification d'un caractère spécial.
- `$ rm \*.c` n'effacera que le fichier `*.c` et non tous ceux qui ont l'extension `.c`

Expressions régulières

- `$ls -C`

binaire01 essai2 1search

binaire02 essaiA 2search

bindery essaiAB 4827.set

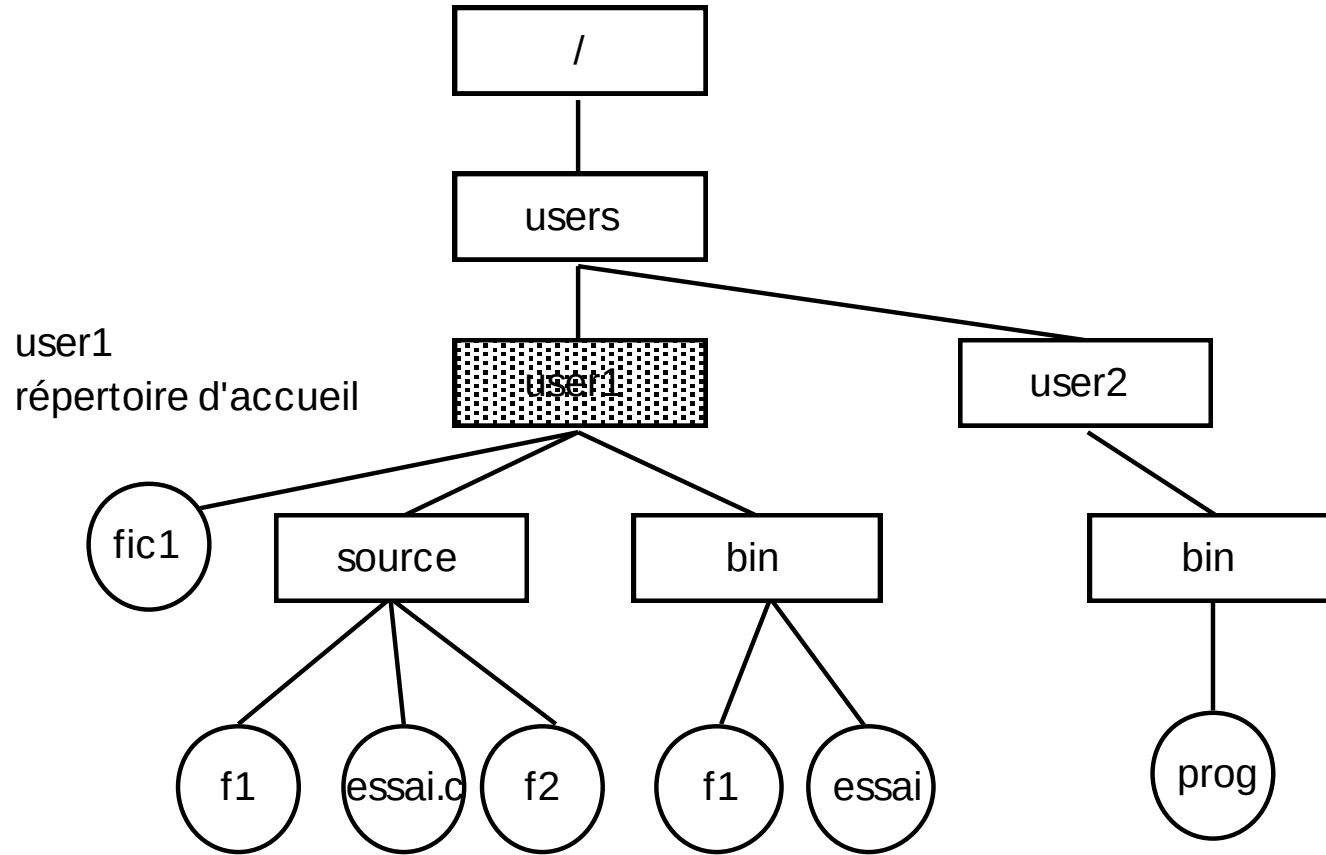
essai1 Essai5 05tar

essai11 Bindery

essai12 Fastsearch

- `$rm essai?`
- `$rm bin*`
- `$rm [A-Z]*`
- `$rm [!a-d]*`

Référencement d'un fichier



la commande **cat** permet de visualiser le contenu d'un fichier ASCII

- `$ cat fic1`<cr> à partir de user1 le chemin est direct.
- `$ cat /users/user1/fic1`<cr> même fichier en référence absolue.
- `$ cat ./source/essai.c`<cr> accès à partir de user1 à essai.c en référence relative.

(Le point signifie que la recherche démarre du répertoire courant.)

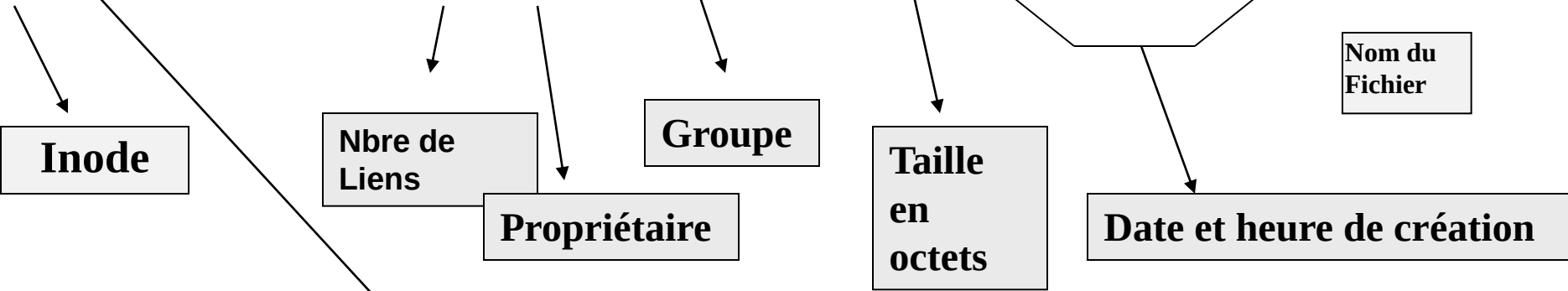
Espace disque occupé
par les fichiers en
nombre de blocs de
512 Octets

Nota:
L'espace minimum d'allocation sur
disque est de 1 KO. Un fichier de 10
Octets occupe donc sur disque un
minimum de 1 KO.

\$ls -ail

Total 282

330	-rwx-----	1	bin	bin	3918	Mars	16	1993	cmos
332	drwxr-xr-x	12	root	sys	192	Juin	08	21:20	conf
337	drwxr-xr-x	2	root	root	96	Mars	16	1993	co*s



Type de fichier:

- fichier normal	c driver de périphérique en mode caractères
d répertoire	b driver de périphérique en mode bloc
p pipe nommé	l lien

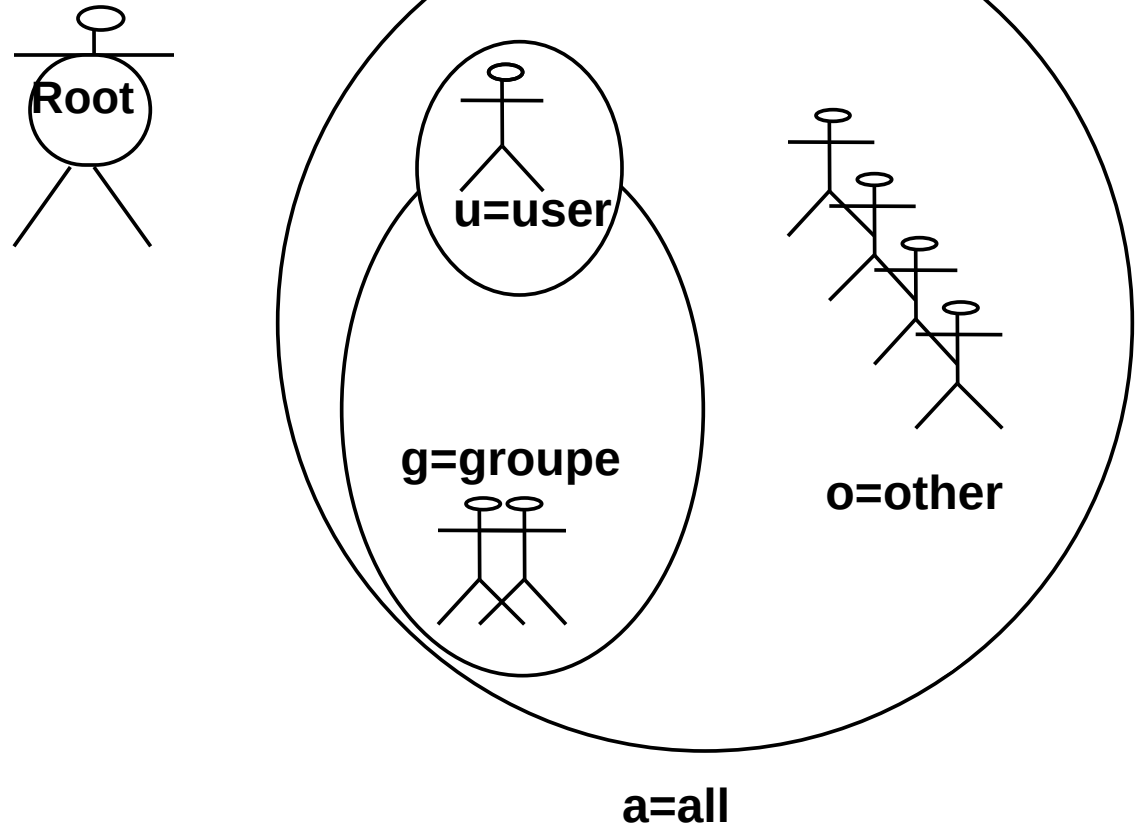
Droits d'accès pour les fichiers

A chaque fichier (ou répertoire) sont associés des droits constitués d'un titre de propriété et des opérations possibles. Pour un fichier donné, l'utilisateur peut être:

Type d'utilisateur	Définition
User	le propriétaire du fichier
Group	un membre du même groupe que le propriétaire du fichier
Other	un autre utilisateur.

Droits d'accès pour les fichiers

Classification des utilisateurs d'UNIX



Droits d'accès pour les fichiers

Les opérations sont en fait 3 actions élémentaires, associées à chaque type d'utilisateur, qui sont gérées par le système:

- **r** (**R**ead : lecture)

le contenu du fichier peut être consulté.

- **w** (**W**rite : écriture)

le contenu du fichier peut être modifié.

- **x** (**eX**ecution : exécution)

désigne un fichier exécutable (binaire ou script).

User	Group	Other
r w x	r w x	r w x

Droits d'accès pour les fichiers

Un fichier repéré par:

-rwxr-xr-- ... **hello**

Est pour:

User	r w x	lisible, modifiable, exécutable
Group	r - x	lisible, exécutable
Others	r - -	lisible

Droits d'accès étendus pour les fichiers exécutables binaires

- **SUID**

Le **Set User ID**entifieur indique que le processus correspondant dispose des droits du propriétaire et non de l'utilisateur qui le lance.

- **SGID**

Le **Set Group ID**entifieur a le même rôle que le précédent, mais relativement au groupe.

- **Sticky bit**

Lorsqu'un programme est exécuté, son code est chargé en mémoire. La mémoire occupée est libérée à la fin de l'exécution. Pour un exécutable utilisé très souvent, comme *ls*, on positionne le Sticky bit afin de le maintenir en mémoire secondaire après son exécution. On évite ainsi un nouveau chargement.

Droits d'accès étendus pour les fichiers exécutables binaires

Droits étendus			User	Group	Other
suid	sgid	sticky	rwX	rwX	rwX

```
-rwxr-Sr-x 1 root tty      8332 jan 30 2004 write*  
---S--x--x 1 root root    86136 oct 21 2003 sudo*  
-rwS--x--x 1 root root    11768 jan 30 2004 chsh*  
-r-S--x--x 1 root root    15760 jui 21 2003 passwd*
```

Droits d'accès pour les répertoires

On retrouve les mêmes 3 actions élémentaires que pour les fichiers mais avec une signification différente:

- **r**

Le contenu du répertoire peut être consulté à l'aide de la commande **ls**.

- **w**

Le contenu du répertoire peut être modifié.

- **x**

Donne le droit d'accès au répertoire à l'aide de la commande **cd**.

Droits d'accès pour les répertoires

Un répertoire repéré par:

drwxr-xr-- ... bin

est pour:

User	r w x	consultable, modifiable, accessible
Group	r - x	consultable, accessible
Others	r - -	consultable

Modification des droits d'accès

-
- Syntaxe littérale

\$chmod [**u|g|o|a**] [**+|-**] [**r|w|x**] [**fichier|répertoire**]

- **u** affecte les droits de l'entité User
- **g** affecte les droits de l'entité Group
- **o** affecte les droits de l'entité Other
- **a** affecte les droits de toutes les entités (peut être omis)
- **+** ajoute un droit
- **-** retire un droit

Modification des droits d'accès

- Changement des droits d'accès en octal

Utilisateur (u)			Groupe (g)			Autres (o)		
R	W	X	R	W	X	R	W	X
400	200	100	40	20	10	4	2	1

- Exemple:


```
$ chmod u+r fichier
```

```
$ chmod og-rwx fichier
```

}

$\Leftrightarrow$ \$ `chmod 400 fichier`

Modification des droits d'accès

- Changement des droits d'accès en mode octal
 - Droits courants:

\$ chmod 444 *fichier* ⇔ r-- r-- r--

\$ chmod 555 *fichier* ⇔ r-x r-x r-x

\$ chmod 155 *fichier* ⇔ --x r-x r-x

Modification des droits d'accès par défaut

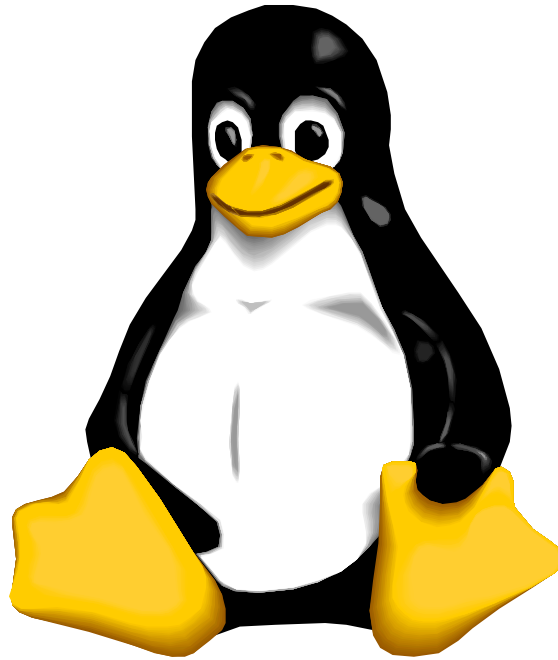
- Mode par défaut: commande « umask »
 - \$ umask u=rwx,g=rx,o=r
 - Mode absolu du masque
 - Différent de celui des droits d'accès
 - La signification des bits est inversée
 - Méthode: calculer le masque des droits et soustraire 7 à chacune des valeurs trouvées
 - Exemple:
rwx r-x --- ⇔ droits 750 ⇔ \$ umask 027

Changement de propriétaire et de groupe d'un fichier

- Commande de changement de propriétaire
`$Chown` nouveau_propriétaire fichier
- Commande de changement de groupe
`$Chgrp` nouveau_groupe fichier

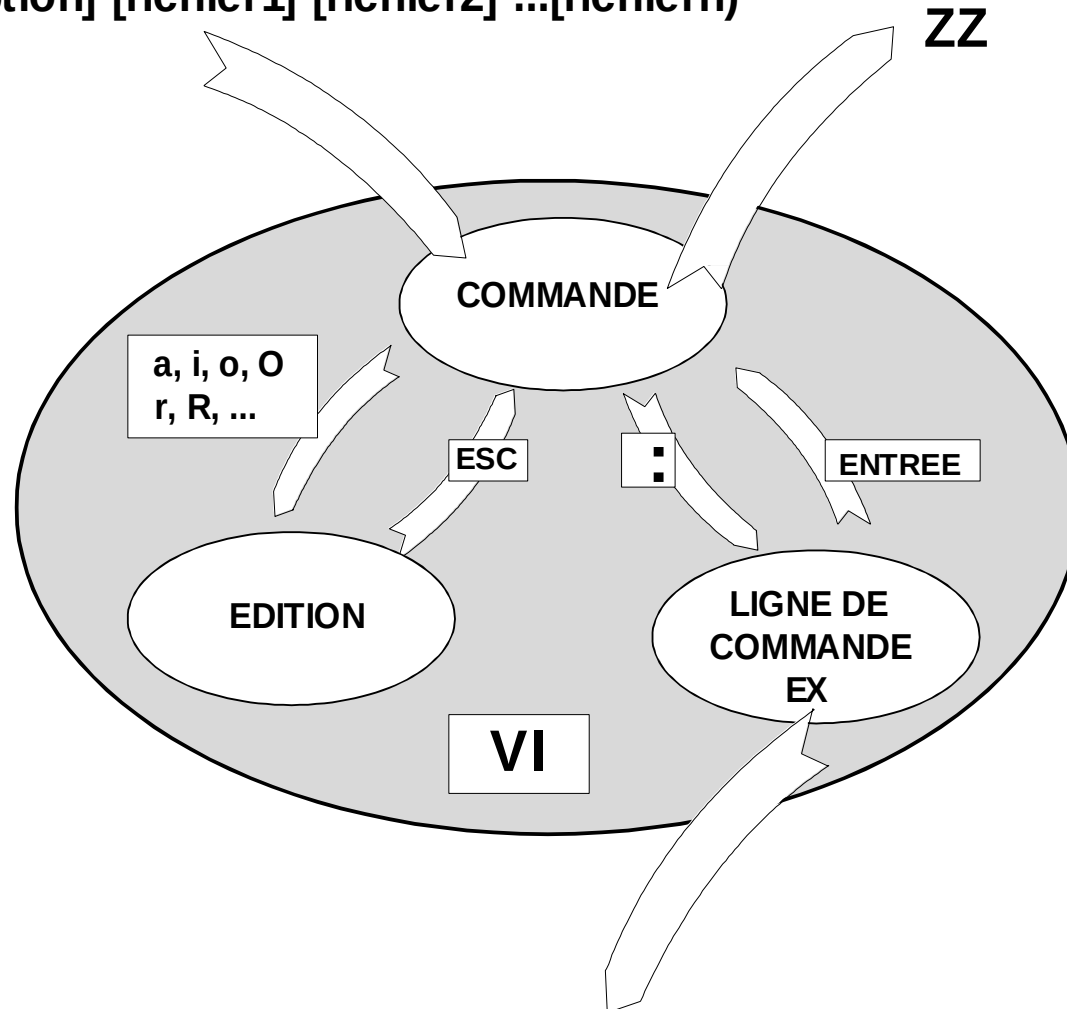
Linux Utilisateur

Edition de textes sous UNIX



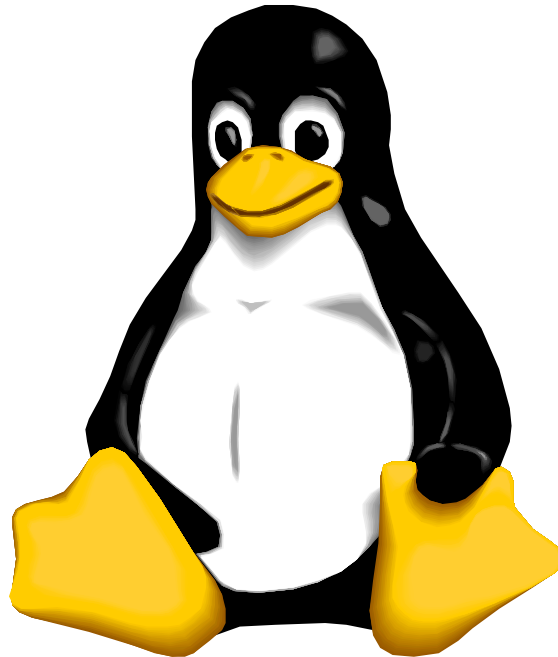
L'éditeur VI

vi [-option] [fichier1] [fichier2] ...[fichierN]



Linux Utilisateur

Gestion des processus



Les processus

On appelle processus ou tâche, un programme en cours d'exécution.

Il existe deux types de processus:

- Les processus systèmes
- Les processus utilisateurs

Les processus

Processus systèmes (démons):

- Rôle: assurer un certain nombre de services généraux accessibles à tous les utilisateurs du système (hôte ou distant)
- Ne sont sous le contrôle d'aucun terminal
- Propriétaire: root (en général)
- Répertoire de travail: / (en général) afin de permettre le démontage de systèmes de fichiers sans avoir à les interrompre
- Créés au lancement du système ou par root en cours d'utilisation du système

Les processus

Processus utilisateurs

- Dédiés à l'exécution d'une tâche particulière
- Durée de vie limitée en général
- Exemples:
 - Phase de connexion au système
 - Exécution de commandes Shell
 - Exécution de programmes « lourds » (StarOffice,...)
- Liés en général à un terminal (Attention: ce n'est pas une obligation)

Les processus

Chaque processus dispose:

- D'un espace d'adressage

 - segment de code

 - segment de données

 - segment de pile

- D'un environnement

 - Identificateurs

 - répertoires

 - registres

 - variables

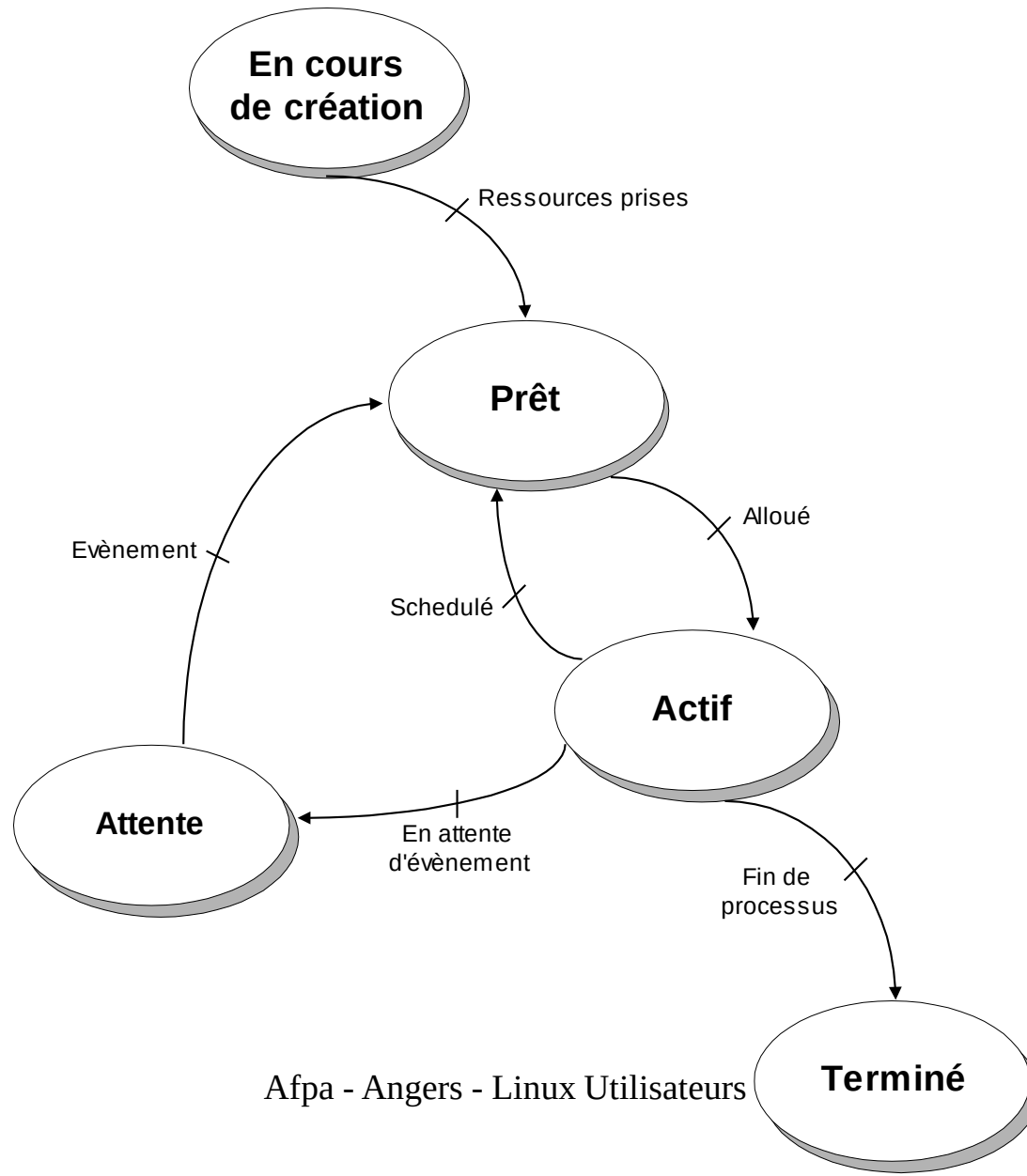
 - fichiers

Les processus

Caractéristiques d'un processus

- numéro de processus
- numéro de processus parent
- Numéro d'utilisateur et de groupe
- Durée de traitement et priorité
- Référence du répertoire de travail actif
- Table des fichiers ouverts
- Traitement en tâche de fond

Les différents états d'un processus



L'état d'un processus

ABBREVIATION	SIGNIFICATION
O	non existant
S	endormi, attente d'un événement (sleeping)
R	executable (running)
I	(intermediate)
Z	terminé (terminated)
T	arrêté (stopped)
B	en attente (waiting)

Les commandes liées aux processus

- **ps** : afficher la listes des processus en cours d'execution
- **kill** : envoyer des signaux aux processus (par exemple pour le terminer)
- **nice** : modifier la priorité d'execution des processus
- **trap** : ignorer des signaux en provenance d'autre processus

Informations affichées

par la commande `ps al`

```
$ ps al
```

FLAGS	UID	PID	PPID	PRI	NI	SIZE	RSS	WCHAN	STA	TTY	TIME	COMMAND
100100	0	396	1	0	0	1496	964	117341	S	1	0:00	/bin/login
100100	0	397	1	0	0	1496	964	117341	S	2	0:00	/bin/login
100100	501	398	1	0	0	1496	964	117341	S	3	0:00	/bin/login
100100	0	399	1	0	0	744	316	189a5a	S	4	0:00	/sbin/min
100100	0	400	1	0	0	744	316	189a5a	S	5	0:00	/sbin/min
100100	0	401	1	0	0	744	316	189a5a	S	6	0:00	/sbin/min
100	0	404	396	18	0	1316	876	117341	S	1	0:00	-bash
100	0	597	404	1	0	1316	752	12e0dd	S	1	0:00	vi

Les processus

■ Interruption de processus

■ Depuis un terminal père:

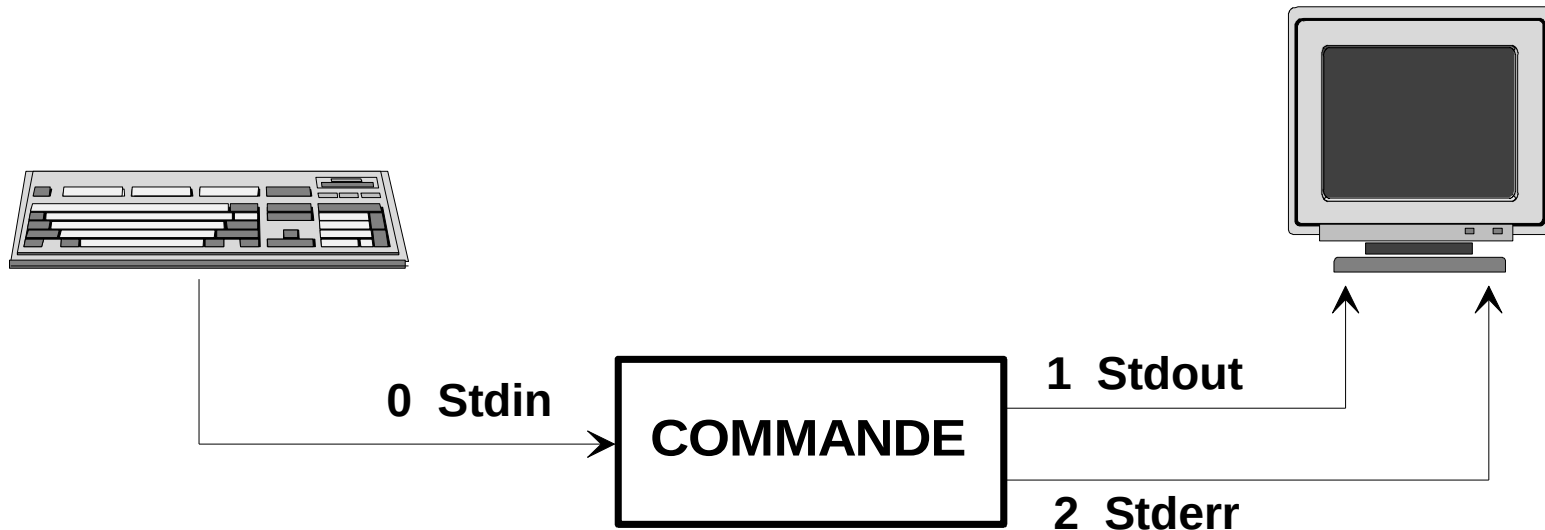
- <CTRL><C>: stoppe le processus
- <CTRL><Z>: suspend le processus
- <CTRL><D>: fin du terminal (et donc du processus)

■ Enchaînement de processus:

■ Séquentiel: le premier n'influe pas sur le second et ainsi de suite (séparation par des « ; »)

■ Communication par tubes (pipes): la sortie d'un processus est reliée à l'entrée d'un autre (séparation par le caractère « | »)

Redirection des entrées / sorties



UN FLUX D'ENTREE STANDARD: le clavier du terminal.

stdin numéro 0 Entrée des informations

DEUX FLUX DE SORTIE STANDARDS: l'écran du terminal

stdout numéro 1 affichage des résultats

stderr numéro 2 messages d'erreurs

Redirection de sortie

Syntaxe

\$commande > nom_de_fichier

Nota:

Le canal de sortie par défaut est le canal 1 (stdout). Il n'est donc pas nécessaire de le préciser dans la commande. Si tel n'était pas le cas la syntaxe serait:

```
$commande 1> nom_de_fichier
```

Redirection de sortie

Dans l'exemple suivant le résultat de la commande `ls` ne sera pas affiché sur l'écran mais écrit dans le fichier `result.ls` dans le répertoire courant:

\$ls > result.ls

Nota:

Si le fichier n'existe pas il est créé, s'il existe déjà son contenu est écrasé par le résultat de la commande.

Redirection de sortie

```
$ls > result.ls
```

```
$who >> result.ls
```

Le contenu du fichier **result.ls** est le résultat des commandes **ls** et **who**.

Nota:

Si le fichier n'existe pas il est créé, s'il existe déjà le résultat de la commande est ajouté au fichier.

Redirection des message d'erreur

```
$ls -ail 2>erreurs.log
```

Le résultat de la commande ***ls -ail*** est affiché à l'écran, les messages d'erreurs éventuels sont écrits dans le fichier ***erreurs.log***.

```
$find / -name core -print 2>/dev/null
```

Le résultat de la commande ***find*** est affiché à l'écran et les messages d'erreur sont expédiés à la "poubelle".

Redirection des message d'erreur

\$ls -ail > result.ls 2>&1

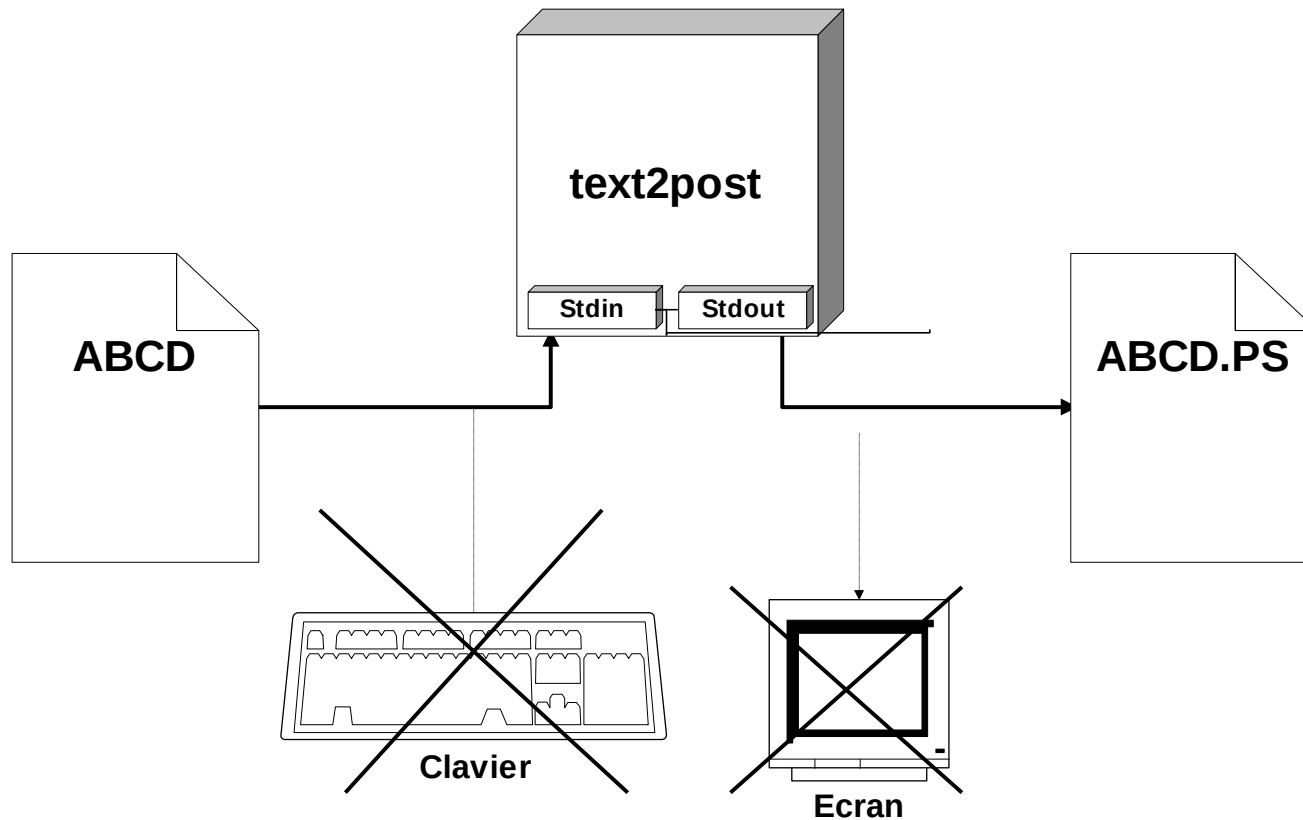
Le fichier result.ls contient à la fois le résultat de la commande ls -ail et les messages d'erreur éventuels.

Redirection de l'entrée standard

\$wc -l < result.ls

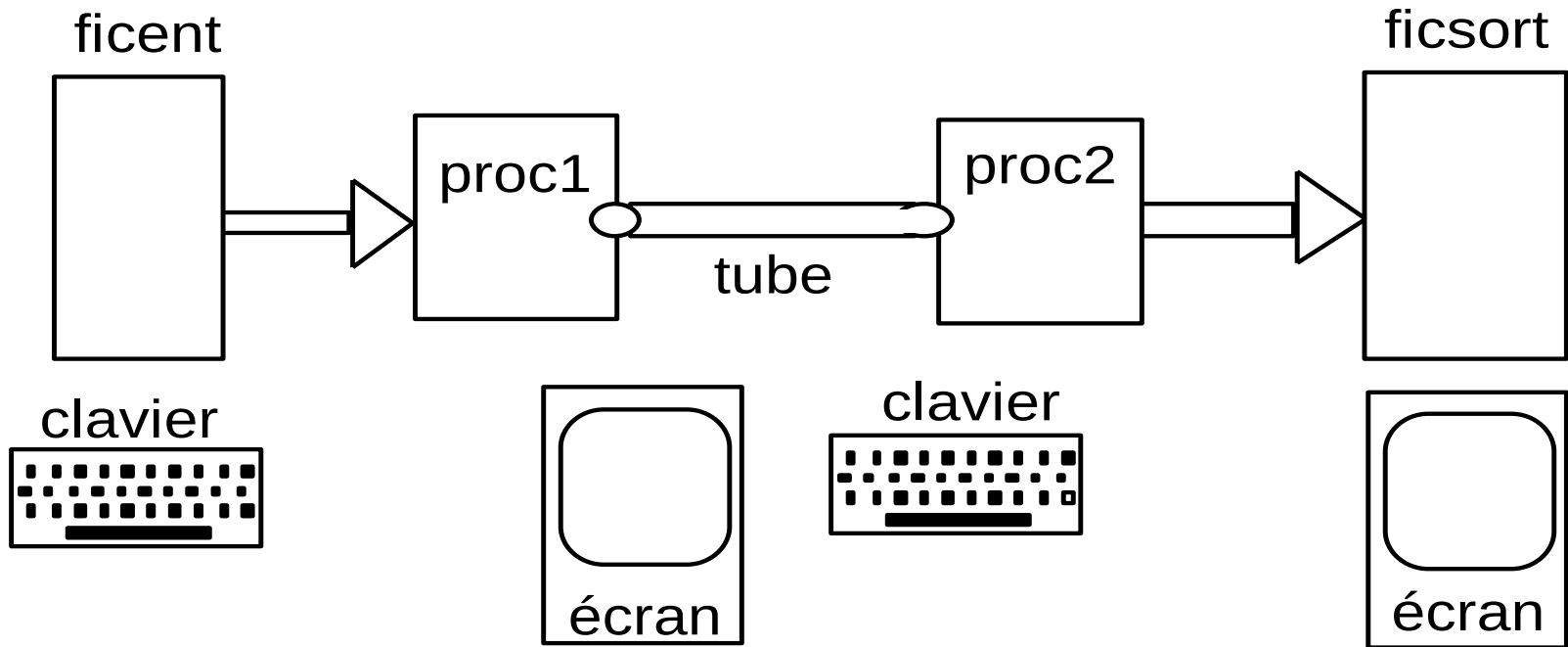
Le symbole " < " permet la redirection de l'entrée standard (stdin) d'une commande depuis un fichier.

Redirection de l'entrée et de la sortie standard



`$text2post < abcd > abcd.ps`

Enchaînement de commandes



Un **tube** ou (**pipe**) est une liaison entre la sortie standard d'un processus et l'entrée standard d'un autre processus.

Enchaînement de commandes

- Sous UNIX, on utilise un mécanisme appelé tube de communication ou pipe.

C'est une liaison entre la sortie standard d'un processus et l'entrée standard d'un autre processus.

<code>\$ls > fic</code>	}	
<code>\$wc -l < fic</code>		
		<code>\$ ls wc -l</code>

- Enchaînement de 3 commandes: (La commande **tee** copie dans un fichier les données sortant du premier processus)

`$ ls | tee fic1 | wc -l`

