Présentation Unix

Matthieu Herrb
LAAS-CNRS

Décembre 1998

Les origines

Unix est né aux laboratoires Bell (Filiale d'ATT).

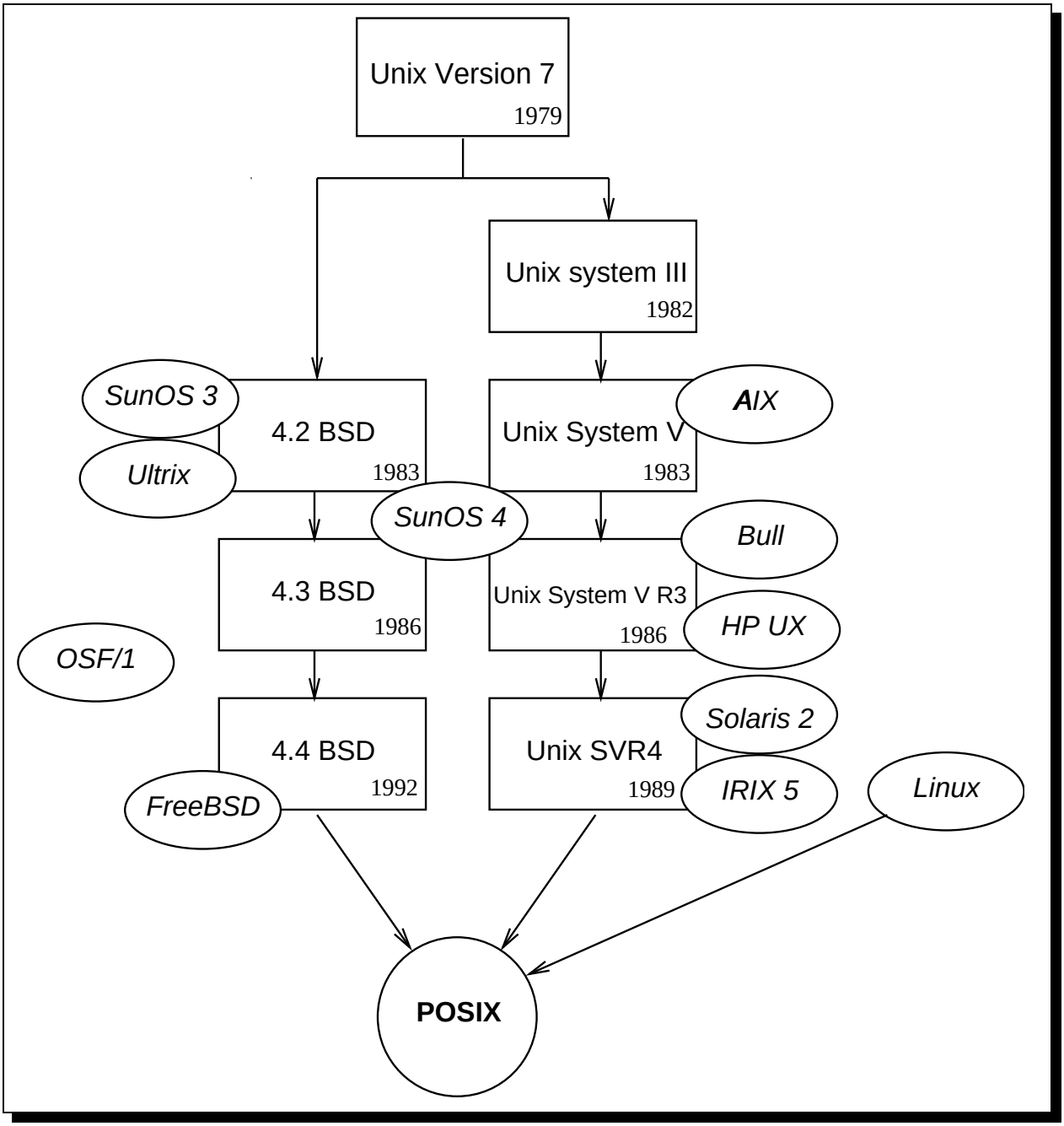
Développé à partir de 1969 par Ken Thompson et Dennis Ritchie pour leurs besoins propres :

Objectif : un système *interactif* pour des petites machines dotés de possibilités *comparables aux grands systèmes*.

Environnement inspiré de Multics et GCOS (Honeywell).

Dès 1973 Unix est réécrit à 90% en langage C (Créé pour l'occasion par Brian Kernighan et Dennis Ritchie).

Évolution


Unix Version 7
1979
Unix system III
1982
SunOS 3
Ultrix
4.2 BSD
1983
Unix System V
1983
AIX
SunOS 4
Bull
4.3 BSD
1986
Unix System V R3
1986
HP UX
OSF/1
Solaris 2
4.4 BSD
1992
Unix SVR4
1989
IRIX 5
Linux
FreeBSD
POSIX

La situation actuelle

Système d'exploitation des stations de travail et des serveurs de base de données.

Fournisseurs :

Digital Equipement, Hewlett Packard, IBM, Silicon Graphics, Sun,. . .

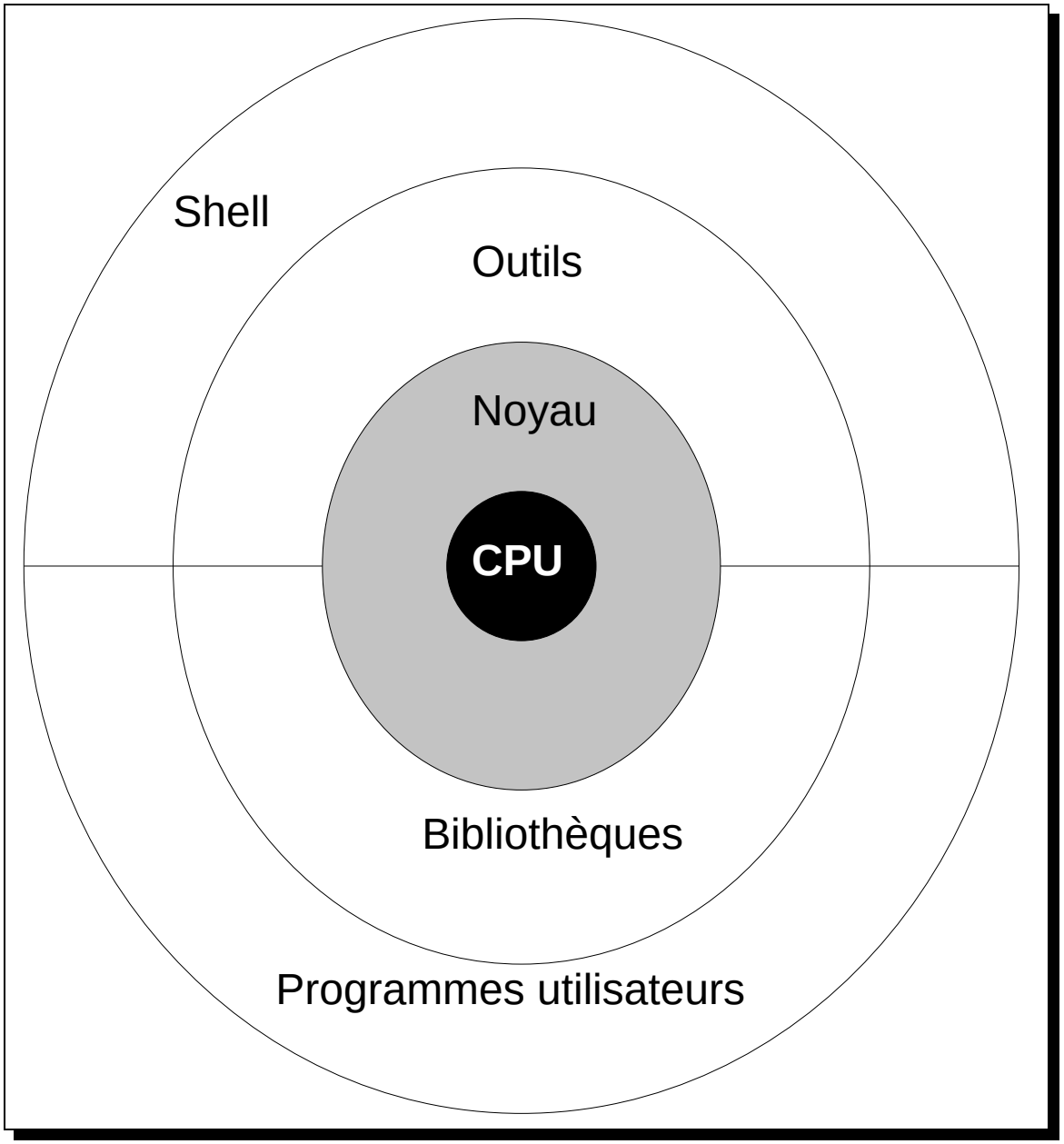
Un consortium : X-Open.

Deux standards : POSIX (IEEE), X-Open.

Un environnement utilisateur : CDE (Common Desktop Environment) X11 + Motif.

Un concurrent : Windows-NT (Microsoft).

Structure générale



Caractéristiques principales

Système d'exploitation

- Multi-tâches en temps partagé
- Multi-utilisateurs
- Interactif
- Intégré aux réseaux

Langages de commande

- Bourne Shell
- Korn Shell
- C-Shell

Plusieurs centaines d'outils.

Outils

- Manipulation de texte
- Développement de logiciels
- Communication
- Documentation
- Bureautique

Le système de fichiers

Partie la plus importante : « Tout est fichier »

Quatre types de fichiers :

ordinaire données, programme

répertoire contient d'autres fichiers ou répertoires

lien symbolique pointe vers un autre fichier

spécial permet l'accès à un périphérique

Un fichier est représenté par une structure (*I-node*) qui stocke les informations sur un fichier: taille, droits d'accès, dates de création, de modification...

Le nom n'est qu'un pointeur sur un I-node.

Un seul type de fichier ordinaire: flot de caractères (8 bits). Les fichiers texte ne sont qu'un cas particulier (lignes séparées par le caractère "Line Feed").

Partage des fichiers en réseau : NFS.

Fichiers et répertoires

Structure arborescente.

- **Répertoire courant**: position dans l'arbre à partir de laquelle on recherche les fichiers.

- Un *chemin absolu* désigne de manière unique un fichier en partant du répertoire racine (/).
  Le caractère « / » sépare les répertoires.

  Exemple : `/home/matthieu/cours/unix/slides.tex`

- Un *chemin relatif* désigne un fichier à partir du répertoire courant.

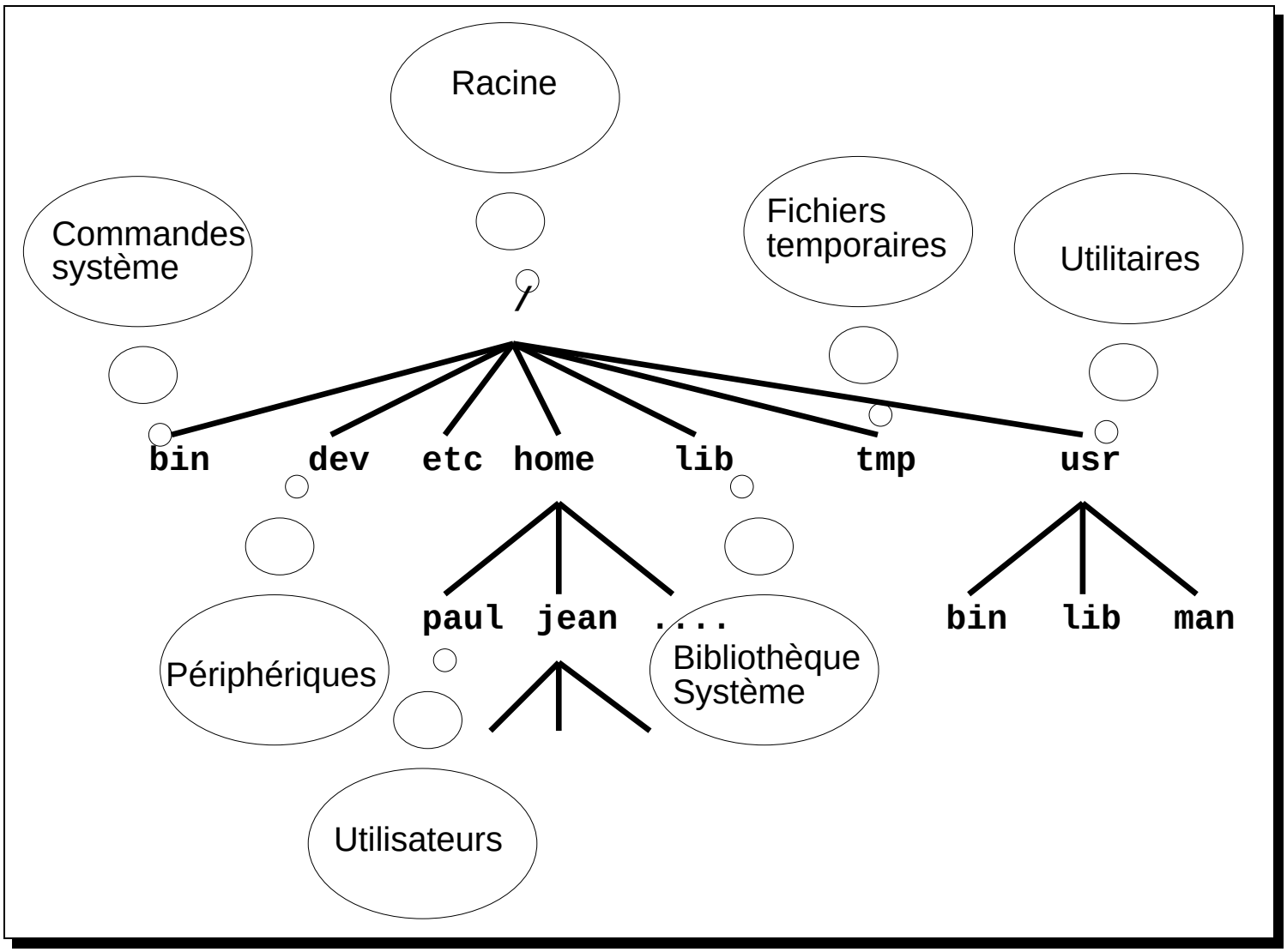
  Caractéristiques des noms de fichiers :

- Majuscules/minuscules ont une signification.

- Tous les caractères sont valides sauf « / » et le code ASCII 0.

- Pas de limitation sur la longueur d'un élément.
  Longueur totale limitée à 1024 caractères.

L'arborescence Unix



".." désigne le répertoire parent.

"." désigne le répertoire courant.

Les utilisateurs

Chaque utilisateur du système est identifié par un nom de *login* auquel est associé :

- un mot de passe
- un identificateur numérique (uid)
- un groupe (gid)
- un commentaire (identité réelle - GCOS)
- un répertoire de travail
- un langage de commandes.

Stockés dans la base de données `passwd`.

Les groupes permettent aux utilisateurs de partager l'accès à certains fichiers.

Le super-utilisateur `root` gère tout le système.

Les droits d'accès

Trois catégories d'utilisateurs :

le propriétaire	u
le groupe	g
les autres	o

Trois types de droits :

Lecture	`r`
Écriture	`w`
exécution	`x`

Pour un fichier les droits sont exprimés par une chaîne de 10 caractères:

tuuugggooo

t: type du fichier :

Fichier ordinaire	`-`
Répertoire	`d`
Lien symbolique	`l`
fichier spécial	`c` ou `b`

uuu: droits du propriétaire
ggg: droits du groupe
ooo: droits des autres

Le super-utilisateur a tous les droits.

Droits d'accès en octal

Droits	valeur octale
`---`	0
`--x`	1
`-w-`	2
`-wx`	3
`r--`	4
`r-x`	5
`rw-`	6
`rwx`	7

L'ensemble des droits est représenté par trois chiffres.

Exemples:

- 600 `rw-------`
- 644 `rw-r--r--`
- 750 `rwxr-x---`

L'accès au système

- depuis un terminal graphique (Clavier, écran, souris)
- depuis un terminal ASCII (console) connecté par une ligne série
- par le réseau

Déroulement:

- Un processus *moniteur de port* attend des demandes de connexion sur chacune de ces entrées en affichant un message.
- le programme `login` demande un mot de passe et vérifie sa validité. Il lance le shell dans le répertoire de travail de l'utilisateur.
- Lorsqu'une session de travail est finie, l'utilisateur la termine par la commande `logout` qui termine le shell et rend la main au moniteur de port.

Le shell

L'interpréteur de commandes est un processus.

- Il affiche un prompt, attend la frappe d'une ligne, analyse cette ligne puis exécute la ou les commande(s).

- Pour exécuter une commande il crée un nouveau processus. Il attend la fin du processus créé puis affiche un nouveau prompt.

- Pendant la saisie d'une ligne les touches DELETE ou BACKSPACE permettent d'effacer les caractères saisis.

- Certains shells (Korn Shell, T-CShell) disposent de possibilités interactives plus étendues (historique, complétion. . .).

Le shell dispose de variables et de structures de contrôle qui en font un langage de commande.

Les commandes

Le shell découpe un ligne en suite de commandes.

Les caractères ;| et les retours chariot séparent les commandes.

La structure générale d'une commande est :

commande arguments

commande est un nom de fichier. Ce fichier est recherché dans une liste de répertoires désignée par la variable PATH. Si aucun fichier n'est trouvé, erreur.

arguments est une liste de paramètres passes a la commande. Trois formes possibles :

-option	option booléenne
-option valeur	option avec valeur
fichier	chemin d'accès à un fichier

L'analyse des arguments est faite par chaque commande $\longrightarrow$ pas de normalisation...

Les caractères spéciaux du shell

Expansion des noms de fichiers :

`?` Un caractère quelconque

`*` N'importe quelle suite de caractères, peut être vide.

`[...]` Un caractère de la liste. Ex: `[A-Z0-9]`.

`[^...]` Un caractère n'appartenant pas à la liste.

`\` supprime la signification spéciale du caractère suivant. (`\*` = `*`).

Exemples :

Si le répertoire courant contient:

fich1.bin fich1.txt fich2.txt fich10.txt fichier.txt readme zzz

Alors:

`fich1*`	fich1.bin fich1.txt fich10.txt
`fich*.txt`	fich1.txt fich2.txt fich10.txt fichier.txt
`fich[0-9]*.txt`	fich1.txt fich2.txt fich10.txt
`???`	zzz

L'historique du c-shell

```
set history=n
```

mémorise *n* commandes.

```
history
```

affiche l'historique.

```
!n
```

réexécute la commande numéro *n*.
$n = !$ → commande précédente.

```
!-n
```

réexécute la *n*ième commande précédente.

```
!xxx
```

réexécute la dernière commande commençant par *xxx*.

```
^xxx^yyy
```

réexécute la commande précédente en remplaçant *xxx* par *yyy*.

Le TCshell

Shell utilisé au LAAS : `/usr/local/bin/tcsh`

Ctrl-P ↑	Commande précédente
Ctrl-N ↓	Commande suivante
ESC-P	Rappelle la commande commençant par le même préfixe
Ctrl-B ←	Curseur vers la gauche
Ctrl-F →	Curseur vers la droite
Ctrl-A	Début de ligne
Ctrl-E	Fin de ligne
Ctrl-T	Transpose les 2 derniers caractères
Ctrl-D	Détruit le caractère sous le curseur
Ctrl-U	Detruit la ligne en cours
Ctrl-R	Réaffiche la ligne en cours
Ctrl-L	Efface l'écran
TAB	Complète le nom de fichier en cours
Ctrl-D	(en fin de ligne) Liste les complétions possibles

L'environnement

Unix dispose d'un mécanisme de variables d'environnement qui peuvent être utilisées pour modifier le comportement des commandes.

`printenv` permet d'afficher la valeur d'une variable.

`setenv` *variable valeur* permet de modifier une variable.

`$VARIABLE`permet d'adresser le contenu d'une variable.

Le fichier `.login` contient les commandes exécutées au début de chaque connexion. Il peut être utilisé pour positionner les variables d'environnement.

Variables standard :

`USER` désigne le nom de login de l'utilisateur

`HOME` désigne le répertoire de login de l'utilisateur

`PATH` spécifie le chemin de recherche des commandes

`TERM` indique le type de terminal ASCII

`DISPLAY` indique le nom du terminal graphique

L'aide en ligne

Unix dispose de toute la documentation de référence en ligne. La doc est organisée en 9 sections :

1	Commandes utilisateur
2	Appels système
3	Fonctions la bibliothèque standard
4	Formats de fichiers
5	Tables
6	Jeux
7	Drivers de périphériques
8	Commandes d'administration système
l	Commandes locales

Visualiser une page du manuel :

```
man [-s section] commande
```

Rechercher les pages qui se rapportent à un mot-clé :

```
man -k mot-clé
```

Naviguer dans l'arborescence

```
pwd
```

Affiche le répertoire courant.

```
cd chemin
```

Change le répertoire courant.

```
cd
```

Retourne dans le répertoire de travail de l'utilisateur.

```
mkdir chemin
```

Crée un répertoire.

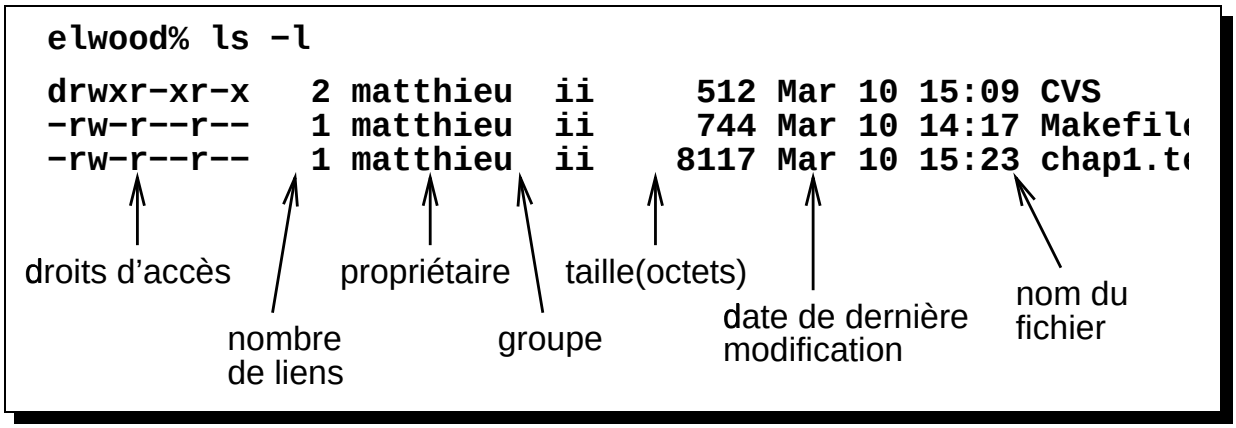
```
rmdir chemin
```

Détruit un répertoire vide.

Visualiser des fichiers

`ls [-l][`*chemin*`]`

Liste le contenu d'un répertoire. `-l` affiche des informations sur chaque fichier.



`cat` *chemin...*

Copie les fichiers spécifiés à l'écran.

`more` *chemin...*

Visualise le fichier écran par écran.

`od [-x]` *chemin...*

Copie le fichier en octal à l'écran. `-x` permet de l'afficher en hexadécimal.

Rechercher de l'information

```
find répertoire -name nom -print
```

Recherche à partir de *répertoire* tous les fichier dont le nom est *nom*.

```
grep expression fichier...
```

Recherche la chaîne *expression* dans les fichiers spécifiés.

Manipuler des fichiers

```
cp [-i] chemin_source chemin_destination
cp [-i] chemin...  répertoire
```

Copie de fichier.

```
mv [-i] chemin_source chemin_destination
mv [-i] chemin...  répertoire
```

Changement de nom ou déplacement de fichiers.

```
rm [-r][-i] chemin
```

Détruit un fichier. Si `-r`, destruction récursive du contenu du répertoire.

`-i` permet de demander une confirmation à l'utilisateur avant de détruire ou d'écraser un fichier existant.

Gérer les droits d'accès

`chmod [-R] droits chemin...`

Change les droits d'accès des fichiers spécifiés. `-R` permet de traverser récursivement tous les sous-répertoires. *droits* est de la forme:

`u|g|o +|- r|w|x`

exemple :

`chmod g+w exemple.data`

Les droits peuvent être indiqués en octal.
Ex: **chmod 644 fichier**.

`chgrp [-R] groupe chemin...`

Change le groupe des fichiers spécifiés. `-R` permet de traverser récursivement tous les sous-répertoires.

Seul **root** peut changer le propriétaire d'un fichier avec **chown**.

Editer un fichier texte avec vi

```
vi chemin
```

2 modes: commande et insertion.

En mode commande

h j k l	déplacement (gauche bas haut droite)
i	passe en mode insertion
x	detruit le caractere sous le curseur
dd	detruit la ligne courante
U	(Undo) Restaure la ligne courante
:w<RET>	sauvegarde le fichier courant
:q<RET>	quitte vi
:q!<RET>	quitte vi sans sauver
ZZ	sauve le fichier en cours et quitte

En mode insertion

<ESC>	repasse en mode commande
<DEL>	detruit le caractère précédant

Les redirections et les pipes

Chaque commande a :

- une entrée standard (clavier)
- une sortie standard (écran)
- une sortie d'erreur (écran)

Redirection de l'entrée :

```
commande < chemin
```

Redirection de la sortie standard :

```
commande > chemin
```

Redirection des sorties standard et d'erreur :

```
commande >& chemin
```

Redirection de la sortie d'une commande vers l'entrée d'une autre :

```
commande1 | commande2
```

Les filtres

Un filtre est une commande qui prend ses donnée d'entrée sur l'entrée standard et fournit ses résultats sur la sortie standard.

Utilisés dans des pipes

Exemple: tri des lignes d'un fichier

```
cat /etc/passwd | sort | more
```

Important : tous les processus formant un pipe tournent en même temps.

Quelques filtres utiles

`sort`

Trie les lignes de l'entrée.

`uniq`

Supprime les lignes identiques dans un fichier trié.

`head -`*n*

Garde les *n* premières lignes.

`tail -`*n*

Garde les *n* dernières lignes.

`grep` *expression*

Garde les lignes qui contiennent l'expression régulière.

`tr` *liste1 liste2*

Remplace chaque caractère de *liste1* par celui correspondant dans *liste2*.

Ex: `tr A-Z a-z` remplace les majuscules par des minuscules.

Autres outils de base

- `awk` langage de recherche de chaînes de de génération de rapports.

- `diff` recherche des différences entre 2 fichiers.

- `sed` éditeur de texte non-interactif (filtre).

- `join` jointure de 2 fichiers sur un champ.

- `comm` affichage des parties communes à 2 fichiers

Imprimantes

```
setenv LPDEST imprimante
```

Choisi l'imprimante par défaut.

```
lp fichier...
```

Envoie des fichiers à l'imprimante par défaut.

```
lpstat -o
```

Visualise la file d'attente des imprimantes.

```
cancel job
```

Enlève le job indiqué de la file d'attente.

```
a2ps fichier ASCII
```

Formatte un fichier ASCII en 2 colonnes par page et l'envoie à l'imprimante.

Le courrier électronique

Lire ses messages

```
mailx
```

`?`	liste des commandes disponibles
num	affiche le message numéro *num*
RET	affiche le message courant et passe au suivant
`d`	détruit le message courant et passe au suivant
`r`	répond à l'auteur du message
`R`	répond à tous les destinataires du message
`s` *file*	sauve le message courant dans un fichier
`q`	sauve la boite aux lettres dans `mbox` et quitte
`x`	quitte sans sauver

Envoyer un message

```
mailx destinataire
```

Terminer par un point (.) en début de ligne.

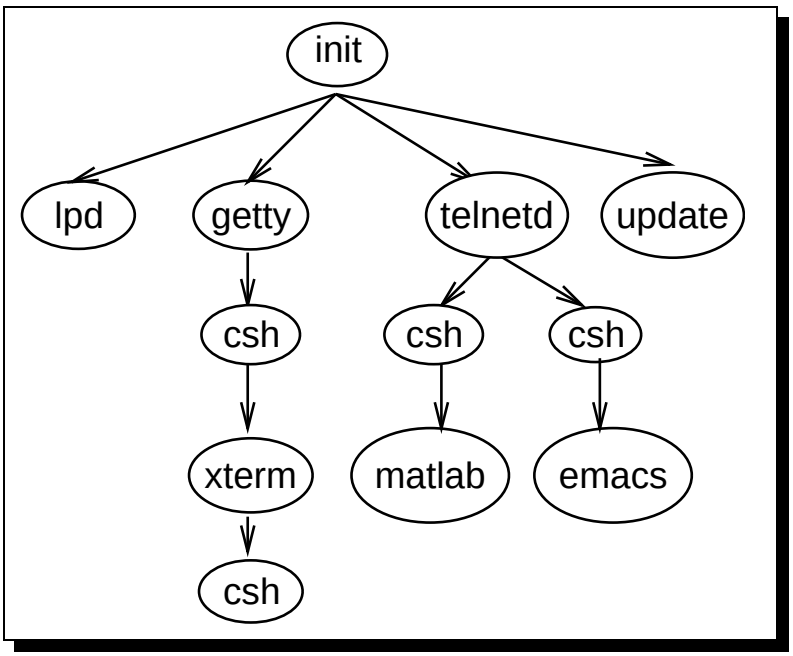
Les processus

Un seul type de processus.

Un processus est créé par un autre processus $\longrightarrow$ arborescence de processus.

Un processus appartient à l'utilisateur qui l'a créé.

Les processus de root sont les processus système.




```
ps -ef
```

Liste tous les processus du système.

Les processus (2)

foreground commande qui garde la main pendant son exécution. Utilise le clavier et l'écran.

background commande qui rend la main une fois le processus lancé. Celui-ci continue sans faire d'entrée/sortie au terminal.

```
commande &
```

Lance un processus en background.

```
jobs
```

Affiche les processus en background.

```
kill %job
```

Termine un processus en background.

```
fg [%job]
```

Ramène un processus en foreground.

Les processus (3)

```
nice commande
```

Exécute la commande avec une priorité plus faible.

```
renice priorité processus
```

Change la priorité du processus indiqué.

```
at date
commande
^D
```

Exécute la commande à la date indiquée.

Les signaux

Unix dispose d'un mécanisme d'interruptions logiciel: les signaux.

2 façons d'envoyer un signal à un processus :

- Caractères de contrôle :

^C	envoie SIGINT (interruption)
^Z	envoie SIGSTOP (supend le process)
^\	envoie SIGQUIT (fin)

- la commande kill :

`kill [-`*signal*`]` *processus*...

Par défaut, envoie SIGTERM.

Développement de logiciels

Cycle :

- Édition :

```
vi fichier.c
```

- Compilation :

```
cc -g -o fichier fichier.c
```

- Exécution :

```
fichier
```

Gestionnaire de compilation : `make`.

SunOS 4.1.x et Solaris 2.x

Sur stations Sun:

transition SunOS 4.1.x (BSD)$\longrightarrow$ Solaris 2.x (SVR4)

Conséquences:

- quelle version d'OS ? `uname -r`
  - `4.1.x` $\longrightarrow$ SunOS 4.1.x = Solaris 1
  - `5.x` $\longrightarrow$ SunOS 5.x = Solaris 2.x
- Variations sur certaines commandes :

SunOS 4.1.x	Solaris 2.x
lpr	lp
lpq	lpstat -o
lprm	cancel
ps -xa	ps -ef
mail	mailx
hostname	uname -n

- Variations sur les logiciels disponibles
- Pour les différencier: variable d'environnement `${HOSTTYPE}`
  - SunOS 4.1.x: `sun4`
  - Solaris 2.x: `sparc`

Bibliographie

- John Levine and Margaret Levine Young,
  Unix for Dummies,
  IDG.

- Harley Hahn,
  A Student's Guide to Unix,
  McGraw Hill.

- Don Libes and Sandy Ressler,
  Life with Unix - A Guide for Everyone,
  Prentice Hall.

- Paul Abrahams and Bruce Larson,
  Unix for the Impatient,
  Addison Wesley.

- Daniel Gilly and O'Reilly staff,
  Unix in a Nutshell,
  O'Reilly.