



IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

TC131C - Introduction à Unix/Linux

Christophe Lohr
Automne 2018

- ▶ Consulter les fiches programme :
Intranet – Enseignement – Portail
- ▶ Module module TC131C :
Maitrise de l'environnement numérique de travail
«Il s'agit d'apporter en tout début de cursus les compétences indispensables à une utilisation efficace et raisonnée de l'environnement numérique de travail tout au long de la scolarité, et notamment dès le premier semestre.»
- ▶ Intégré dans l'UV TC131 : *Outils de l'ingénieur*
«... assimiler un socle d'outils trans-disciplinaires de l'Ingénieur, quelle que soit sa spécialité.»

- ▶ Savoir utiliser
 - ▶ Connaître les commandes de base
 - ▶ Savoir gérer son environnement de travail
- ▶ Connaître les concepts fondamentaux
 - ▶ Système de fichiers
 - ▶ Processus
 - ▶ Notion de Shell
- ⊕ Pour pouvoir aller plus loin par soi même

Introduction

Le système de fichiers

Utilisation courante

Les processus

L'interface graphique X-Window

Introduction

Le système d'exploitation

Historique

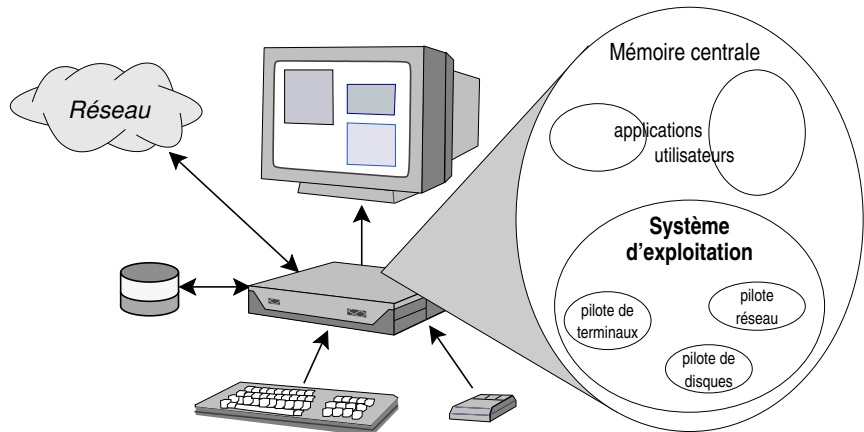
De Unix à Linux

Le système de fichiers

Utilisation courante

Les processus

L'interface graphique X-Window



Introduction

Le système d'exploitation

Historique

De Unix à Linux

Le système de fichiers

Utilisation courante

Les processus

L'interface graphique X-Window

- ▶ Unix naît officiellement le 1^{er} janvier 1970 dans les laboratoires Bell AT&T : Ken Thompson et Dennis Ritchie
- ▶ Années 70 : développement d'Unix : 1973 langage C...
- ▶ Années 80 : deux filières
 - ▶ Univ. Berkeley : système Unix BSD (Berkeley Software Development)
 - ▶ AT&T : Unix Système V (déjà...), version commerciale standard
 - ▶ Sun (création 1983) et Digital (Dec) choisissent BSD : SunOS (jusqu'à la version 4) et Ultrix (Dec)
 - ▶ 1984 : Richard Stallman crée la Free Software Foundation et la Licence Publique Générale (GNU GPL)

- ▶ Années 90 : La version Système V s'impose, Sun s'y rallie (Solaris ou encore SunOS-5), Digital adopte une autre version développée par le consortium OSF, HP-UX (HP), AIX (IBM) sont des systèmes V.
- ▶ Un trublion apparaît : Linus Torvalds qui écrit le noyau Linux
- ▶ Les versions BSD continuent en logiciel libre : FreeBSD, OpenBSD...

Introduction

Le système d'exploitation

Historique

De Unix à Linux

Le système de fichiers

Utilisation courante

Les processus

L'interface graphique X-Window

- ▶ Philosophie d'Unix :
 - ▶ (presque) tout s'utilise comme un fichier
 - ▶ "Do one thing, do it well" (Doug McIlroy, l'inventeur des *pipes Unix*) :
 - ▶ Write programs that do one thing and do it well.
 - ▶ Write programs to work together.
 - ▶ Write programs that handle text streams, because that is a universal interface.
- ▶ Caractéristiques d'un système d'exploitation Unix :
 - ▶ Multitâche (multi processus)
 - ▶ Multi utilisateurs
- ▶ Spécificités (de Linux et de tous les Unix) :
 - ▶ Son Système de Gestion de Fichiers
 - ▶ La gestion des processus
- ▶ → Linux c'est (une implémentation) Unix

Linux (p.ex. Linux 3.2.0)

Le noyau, uniquement !

GNU/Linux

+ commandes Unix de base (implémentation de GNU)
copier un fichier, répertoire, permissions utilisateur..

une distribution Linux (p.ex. Ubuntu 12.04, Debian Wheezy)

+ organisation des fichiers, outils d'administration,
applications

Introduction

Le système de fichiers

Structure, nommage, droits

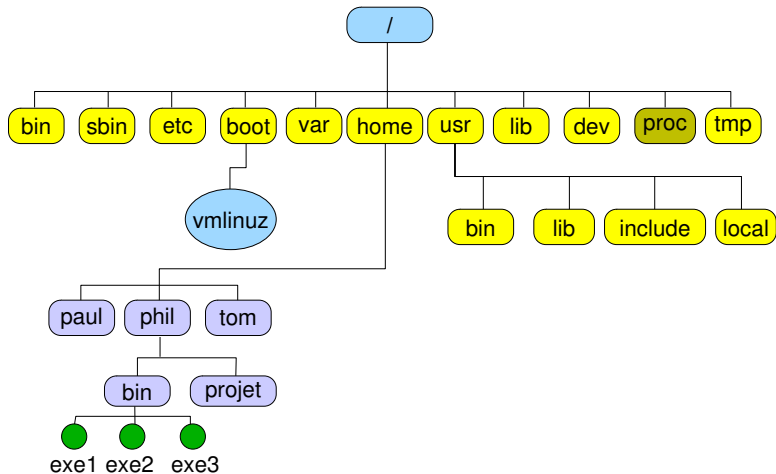
Organisation sur disques

Utilisation courante

Les processus

L'interface graphique X-Window

- ▶ Fichier ordinaire
 - ▶ Simple suite d'octets parfois réduite à 0 (fichier vide)
- ▶ Répertoire
 - ▶ «Fichier» contenant des références sur des fichiers
 - ▶ Permet de créer une arborescence de fichiers et répertoires
- ▶ Liens
 - ▶ Références multiples sur des fichiers qui n'existent réellement que dans une seule copie
- ▶ Fichiers spéciaux
 - ▶ Références sur des périphériques



▶ Nommage absolu

- ▶ par rapport à la racine, le nom commence par /
- ▶ `/home/phil/bin/exe1`

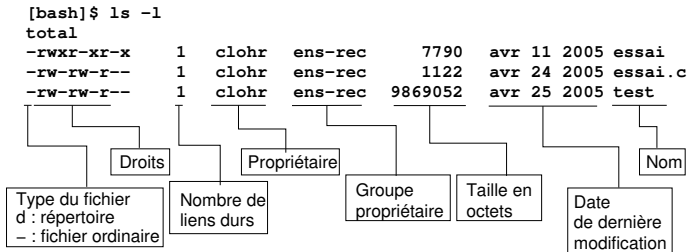
▶ Nommage relatif

- ▶ relatif au répertoire dans lequel on est :

<code>home/phil/bin/exe1</code>	si on est dans /
<code>phil/bin/exe1</code>	si on est dans <code>/home</code>
<code>bin/exe1</code>	si on est dans <code>/home/phil</code>
<code>exe1</code>	si on est dans <code>/home/phil/bin</code>

▶ La commande ls

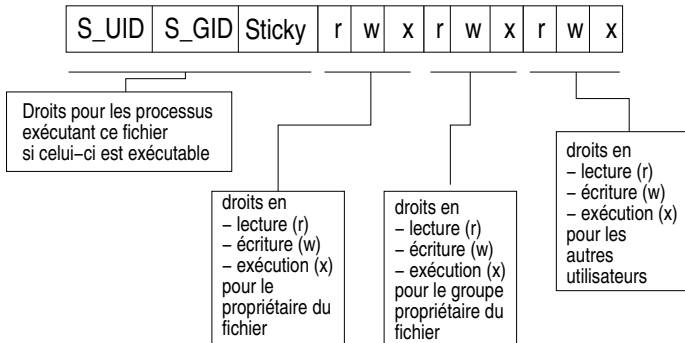
▶ Exemple :



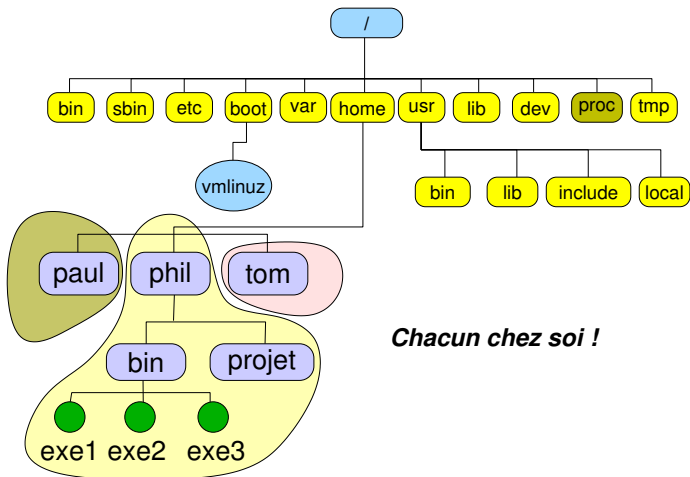
- ▶ Ce sont les fichiers dont le nom commence par un point
 - ▶ Exemple : `.bashrc`
- ▶ Par défaut les outils d'affichage du contenu des répertoires n'affichent pas les fichiers cachés
- ▶ Ce sont généralement des fichiers de configuration d'applications
- ▶ On peut les comparer à la base de registres sur des systèmes concurrents à Unix/Linux
- ▶ Il existe aussi des répertoires cachés, leur nom commence aussi par un point

- ▶ Un répertoire n'est jamais vide, même à sa création, il contient déjà deux références sur des répertoires de nom « . » et « .. »
 - ▶ Le répertoire « . » (point) constitue une référence sur le répertoire lui-même
 - ▶ Le répertoire « .. » constitue une référence sur le répertoire immédiatement au dessus dans l'arborescence (le répertoire *père* en quelque sorte)
- ▶ Utilisation
 - ▶ Nommage sans ambiguïté d'un fichier local : `./test` par exemple
 - ▶ Nommage rapide d'un fichier au dessus : `../fichier` par exemple

- ▶ Dans le champ mode de l'inode
 - ▶ 12 bits



- ▶ Ce sont les mêmes types de droits que pour les fichiers : **r**, **w** et **x** et même **t**
- ▶ La sémantique associée est toutefois différente
- ▶ Droits :
 - ▶ **r** : le répertoire est lisible, on peut lister son contenu
 - ▶ **w** : le répertoire est «écrivable», on peut y créer des fichiers ou des répertoires
 - ▶ **x** : le répertoire n'est pas exécutable, il est accessible : on peut aller dedans, ou le traverser pour accéder à ce qu'il contient (fichiers, sous-répertoires)
 - ▶ **t** : le *sticky bit*, valable pour un répertoire ouvert en **w** à tout le monde, indique que seul un propriétaire de fichier peut supprimer ce fichier.



- ▶ Référencent des périphériques
 - ▶ Permettent les échanges (lectures/écritures) avec les pilotes des périphériques
 - ▶ Permettent le contrôle de ceux-ci
- ▶ Deux types
 - ▶ Les périphériques en mode bloc
 - ▶ les échanges se font par bloc d'octets (par «pages»)
 - ▶ Les périphériques en mode caractère appelé encore mode transparent (on dit plutôt mode *raw*)
 - ▶ les échanges se font octet par octet
 - ▶ Les disques sont plutôt en mode bloc, les terminaux en mode *raw*

► Exemple d'entrée dans /dev

```
[bash]$ cd /dev; ls -l hda1  
brw-rw---- 1 root  disk   3,  1 mar 24 2001 /dev/hda1
```

Indique le mode
b : bloc
c : caractère

Nombre majeur
indique le numéro
du pilote (driver)
dans le noyau

Nombre mineur
indique le numéro
du périphérique
piloté par le driver

Introduction

Le système de fichiers

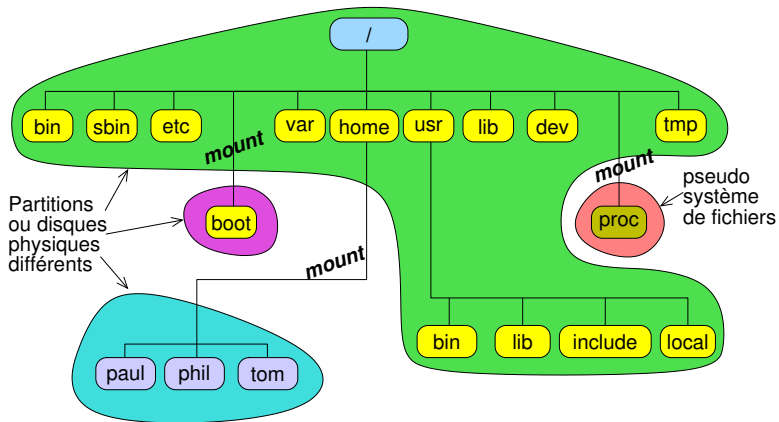
Structure, nommage, droits

Organisation sur disques

Utilisation courante

Les processus

L'interface graphique X-Window



Introduction

Le système de fichiers

Utilisation courante

Les commandes et leur syntaxe

La documentation

Les processus

L'interface graphique X-Window

- ▶ Manière moderne
 - ▶ Via l'interface graphique
- ▶ Manière moins moderne... mais ô combien efficace !
 - ▶ Via un émulateur de terminal (xterm, gnome-terminal, ...)
 - ▶ Dans un terminal virtuel de console (accessible via Ctrl-Alt-F1 à F6 depuis l'interface graphique)
 - ▶ Ces méthodes lancent un processus interpréteur de commandes associé au terminal, appelé *Shell*
 - ▶ Le *Shell* est une interface entre le clavier et le système, il offre des facilités pour entrer les commandes mais aussi un vrai langage de programmation

► Syntaxe générale

`[invite-shell] nom_commande [-options] [arguments...]`

affiché par le shell
Indique que le shell
est prêt à recevoir
une nouvelle commande

Le nom de la commande
en notation absolue
(/bin/ls par exemple)
ou relative (ls)

liste d'arguments éventuels

options, séparées (-r -p par exemple)
ou concaténées (-rp)

- ▶ Le **PATH** est une variable d'environnement
- ▶ Contient la liste des répertoires où se trouvent les commandes que l'on peut appeler par leur nom relatif le plus court : `ls` au lieu de `/bin/ls` par exemple
- ▶ Ne pas croire le système lorsqu'il vous dit : *«command not found»*, il se peut que votre variable PATH soit mal configurée...
- ▶ Configuration dans le shell directement ou dans `$HOME/.bashrc` :
 - ▶ `export PATH=$PATH:/usr/local/bin` (par exemple)

- ▶ Les commandes en arrière-plan (*background*)
 - ▶ la ligne de commande se termine par le caractère `&`
 - ▶ le shell lance la commande et redonne la main aussitôt sans attendre qu'elle se termine. La commande poursuit son exécution en arrière-plan
- ▶ Les redirections
 - ▶ redirection du fichier standard de sortie
 - ▶ *commande* `> fichier`
 - ▶ Redirection du fichier standard d'erreur
 - ▶ *commande* `2> fichier`
 - ▶ redirection du fichier standard de sortie et d'erreur
 - ▶ *commande* `> fichier 2>&1`

► Pour réaliser des filtres

► *commande1* | *commande2* | *commande3* ...

► le texte normalement affiché par *commande1* est redirigé vers *commande2* (il est lu par *commande2* et n'apparaît pas à l'écran). Le résultat est envoyé vers *commande3* et ainsi de suite

► Exercice :

► afficher le contenu du répertoire /usr à l'aide de la commande `ls -l`

► en utilisant un tube de communication et la commande `head`, n'afficher que les 5 première lignes du résultat précédent

► en utilisant un autre tube et la commande `tail`, n'afficher que la dernière ligne du résultat précédent

► Voir le manuel en ligne pour savoir comment utiliser ces commandes

- ▶ Un caractère spécial qui en remplace plusieurs...
 - ▶ Pour les noms de fichiers
 - ▶ Un genre d'expression rationnelle (mais non POSIX)
- ▶ Exemple : `rm *` (efface les fichiers du répertoire)
- ▶ Reconnus par le shell :
 - ▶ `?` : n'importe quel caractère
 - ▶ `*` : zéro ou plusieurs caractères
 - ▶ `[A1x]` : le caractère A ou 1 ou x
 - ▶ `[a-z]` : les caractères de a à z (code ASCII)
 - ▶ `{ab,ac}` : la chaîne ab ou ac

(liste non exhaustive...)

- ▶ Créer, naviguer parmi les fichiers et répertoires
 - ▶ `ls cd pwd cp mv rm mkdir rmdir`
- ▶ Afficher — éditer des fichiers
 - ▶ `more less — vi emacs touch`
- ▶ Filtres texte
 - ▶ `echo cat grep sort uniq sed tail tee head cut
tr split paste printf`
- ▶ Comparaison de fichiers
 - ▶ `comm cmp diff patch`

...

- ▶ Administration basique (niveau utilisateur)
 - ▶ `chmod chown ps su w who`
- ▶ Communication
 - ▶ `mail telnet ftp finger ssh`
- ▶ Shells
 - ▶ `sh csh ksh zsh bash tcsh`

Introduction

Le système de fichiers

Utilisation courante

Les commandes et leur syntaxe

La documentation

Les processus

L'interface graphique X-Window

- ▶ La commande `man`... Savoir lire la syntaxe de `man`...
- ▶ Par exemple : `man rm`

```
RM(1)          Manuel de l'utilisateur Linux      RM(1)
NOM
    rm - Effacer des fichiers.
SYNOPSIS
    rm [-dfirvR] nom...
```

Nom de la
commande

Liste des options
Entre [] signifie que ces options
ne sont pas obligatoires.
(Si on les indique cependant,
on ne mettra pas les crochets)

Le ou les arguments.
Ici les caractères ... indiquent que
nom peut être répété plusieurs fois

- ▶ La commande **info**
 - ▶ alternative à `man`
 - ▶ généralement plus à jour
 - ▶ interface texte «à la emacs» avec menu
- ▶ La commande **apropos**
 - ▶ recherche les commandes «à propos de `xxxx`» (recherche dans les mots clefs des pages de `man`)

- ▶ La commande `locate`
 - ▶ recherche les occurrences de la chaîne de caractères qui lui est passée en arguments dans une base de données mise à jour via la commande `updatedb`
- ▶ La commande `whereis`
 - ▶ Recherche le nom de commande passé en argument dans un certain nombre de répertoires standards
- ▶ La commande `which`
 - ▶ Recherche dans le `PATH` où se trouve la commande indiquée en argument
- ▶ La commande `find`
 - ▶ Recherche n'importe quoi n'importe où

Introduction

Le système de fichiers

Utilisation courante

Les processus

Environnement, cycle de vie

L'interface graphique X-Window

- ▶ Le concept de processus
 - ▶ Dans une première approche on peut dire qu'un processus est un programme en cours d'exécution, dans un environnement donné, dans la mémoire centrale
- ▶ Notion d'environnement
 - ▶ Ensemble d'informations complémentaires au programme en exécution qui viennent paramétrer l'exécution
 - ▶ Essentiellement trois types d'information d'environnement
 - ▶ Les variables d'environnement
 - ▶ L'identité de l'utilisateur au nom duquel s'exécute le processus
 - ▶ Les fichiers ouverts (notamment ceux d'entrée/sortie standard)

- ▶ Un processus est toujours créé par le noyau, à la demande d'un autre processus
- ▶ Le processus qui demande la création est appelé le père, le processus créé est appelé le fils
- ▶ Le processus fils est créé en mémoire centrale dans une zone mémoire distincte du processus père
- ▶ Le processus fils est la copie intégrale du processus père. Mais un détail technique permet au développeur de différencier les instructions exécutées par le père de celles exécutées par le fils

- ▶ Chaînes de caractères, en majuscules par coutume
 - ▶ Syntaxe NOM=valeur
 - ▶ Regroupées dans un espace mémoire appelé «tableau des variables d'environnement»
 - ▶ Ce tableau est hérité par les processus fils et résiste à l'exécution d'une commande (voir plus loin)
- ▶ Quelques variables standard
 - ▶ PATH, HOME, USER, LOGNAME, DISPLAY, ...

- ▶ Deux identités d'utilisateur !
 - ▶ L'utilisateur réel : celui qui a lancé le processus, identifié par son numéro d'utilisateur dans `/etc/passwd` (*ruid : real user ID*)
 - ▶ L'utilisateur effectif :
 - ▶ Dans la majorité des cas il s'agit de l'utilisateur réel
 - ▶ Si le fichier exécuté (qui a donné naissance au processus) a le bit `S_UID` positionné (une lettre `s` apparaît à la place du `x` des droits du propriétaire du fichier) alors l'utilisateur effectif est le propriétaire du fichier exécuté (*eid : effective user ID*)
 - ▶ Attention : trou de sécurité potentiel, surtout si le fichier appartient à `root`
- ▶ Deux identités de groupe
 - ▶ Concept identique à ci-dessus : groupe réel, groupe effectif (bit `S_GID`)

- ▶ Trois fichiers standards
 - ▶ Le fichier standard d'entrée (descripteur 0, FILE Pointer stdin)
 - ▶ Le fichier standard de sortie (descripteur 1, FILE Pointer stdout)
 - ▶ Le fichier standard de sortie d'erreur (descripteur 2, FILE Pointer stderr)
- ▶ Ouverts par défaut lors du lancement d'un exécutable
- ▶ Associés virtuellement au clavier pour l'entrée standard et à l'écran pour les deux autres
- ▶ Ils peuvent être redirigés vers des fichiers réels ou des tubes de communication

- ▶ Lister les processus
 - ▶ La commande `ps`
 - ▶ Nombreuses options : `ax`, `axl`, `axf`, `-ef`, etc.
 - ▶ La commande `top`
 - ▶ Comme `ps axu`, avec réaffichage régulier, plus des informations sur la charge et l'occupation mémoire
- ▶ Arrêter un processus
 - ▶ Si on a le contrôle (processus en premier plan dans un terminal)
 - ▶ Sans le tuer (arrêt momentané) : `<Ctrl-Z>`
 - ▶ En le supprimant : `<Ctrl-C>`
 - ▶ La commande `kill` (voir pages suivantes)

- ▶ Un signal est une sorte d'interruption logicielle envoyée à un processus par le noyau après qu'un événement particulier soit intervenu
- ▶ L'événement peut être :
 - ▶ Une faute logicielle (division par 0, manipulation d'une adresse mémoire interdite, erreur d'alignement de donnée)
 - ▶ Terminaison d'un processus fils : par défaut (mais paramétrable) le père est prévenu
 - ▶ Intervention de l'utilisateur via le shell ou l'interface graphique pour tuer le processus ou le stopper ou autre (modification de la taille d'une fenêtre par exemple)
- ▶ Dans la plupart des cas le signal est fatal au processus

Signal	Numéro	Fonction
HUP	1	Signal envoyé au processus en premier plan associé à un terminal lorsque celui-ci est fermé
INT	2	Envoyé depuis le clavier avec la combinaison de touches <CTRL-C> (par défaut)
QUIT	3	Envoyé depuis le clavier avec la combinaison de touches <CTRL- > (par défaut)
KILL	9	Ne peut être intercepté, envoyé depuis le clavier via la commande kill (kill -9 ou kill -KILL)
TERM	15	Envoyé via le clavier par la commande kill simple
SEGV		Erreur de segmentation, accès à une zone mémoire interdite
CLD		Terminaison d'un fils
WINCH		Modification de la taille de la fenêtre associée à l'application
STOP		Arrêt du processus sans le terminer. Envoyé via la combinaison de touches <CTRL-Z>
URG		Une données urgente a été reçue via le protocole TCP et est en attente de lecture (voir le cours sur la programmation réseau)
IO		Des données réseau sont arrivées et sont en attente de lecture. (voir le cours sur la programmation réseau)
USR1		Nom de signal utilisable par le développeur, à son gré
USR2		Nom de signal utilisable par le développeur, à son gré

- ▶ `kill [numéro_ou_nom_de_signal] numéro_processus / numéro_job`
- ▶ le numéro ou le nom de signal sera en général omis sauf si le résultat est négatif, auquel cas on pourra essayer le signal KILL (-9) qui ne peut pas être intercepté par le processus
- ▶ Le numéro de processus sera obtenu par `ps`
- ▶ Le numéro de job n'est valable que pour les processus en arrière plan (background) ou les processus stoppés. On peut le connaître avec la commande `jobs`
- ▶ Le programmeur d'application peut gérer l'arrivée des signaux (sauf le signal 9) et les ignorer ou les traiter de manière à ce que le processus se termine proprement ou ne se termine pas

- ▶ Lorsqu'un processus fils se termine, son père doit acquitter la terminaison. C'est un problème de programmeur, pas d'utilisateur. Si l'acquiescement n'est pas fait, le processus terminé reste dans la liste des processus (état Z), il est appelé «zombie»
- ▶ Un processus zombie est un processus fils pour lequel son père n'a pas acquiescé la terminaison
- ▶ Le processus zombie est vidé de sa substance mais reste dans la liste des processus de la machine et peut être listé par ps
 - ▶ On ne peut plus le supprimer, il faut supprimer le père pour que le zombie disparaisse
 - ▶ Il est généralement dû à une erreur de programmation

Introduction

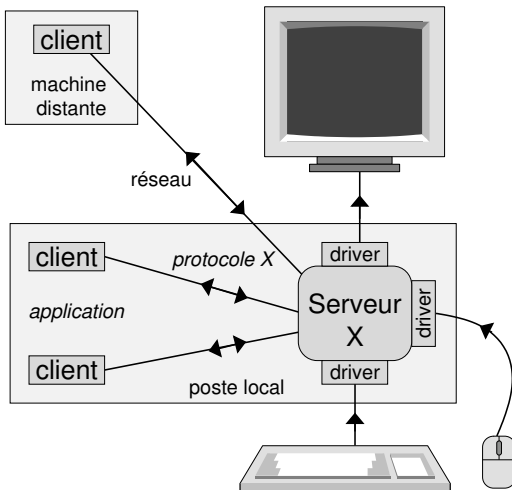
Le système de fichiers

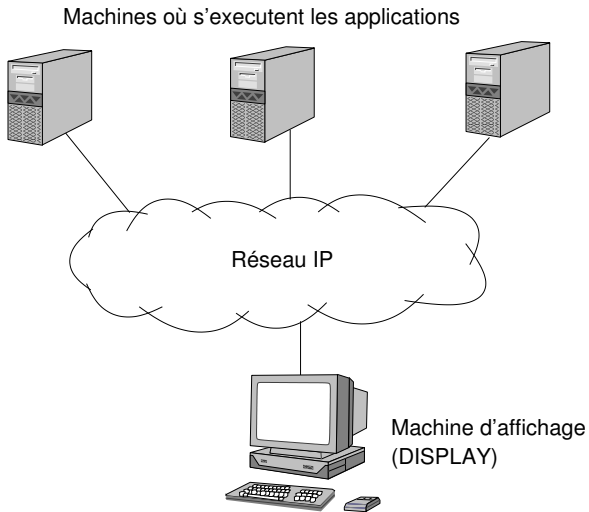
Utilisation courante

Les processus

L'interface graphique X-Window

Client-serveur, authentification, bureau





- ▶ Toute application X peut s'exécuter sur une machine et s'afficher sur une autre
- ▶ La machine d'affichage est indiquée dans une option (`-display`) ou une variable d'environnement (`DISPLAY`)
 - ▶ Exemples :
 - ▶ `$ xterm -display lune:0.0`
 - ▶ `$ export DISPLAY=lune:0.0`
 - ▶ `$ xterm`
 - ▶ format du display :
adresseOuNomMachine :numServ.numEcran

- ▶ Par défaut il n'est pas possible d'afficher des fenêtres sur un «display» utilisé par un autre utilisateur si celui-ci n'a pas donné l'autorisation
- ▶ Les autorisations sont possibles avec la commande `xhost`
 - ▶ `xhost +terre`
 - ▶ autorise les «connexion graphiques» depuis la machine terre
 - ⊕ authentification par adresse IP
- ▶ Autorisations avec la commande `xauth`
 - ▶ plus complexe mais plus *sûr*
 - ▶ fichier `.Xauthority`
 - ⊕ authentifie un utilisateur (qui doit posséder le bon *cookie* de 128 bits)

- ▶ Le serveur X sait gérer l'affichage mais il ne sait pas quoi ni comment afficher !
 - ▶ ce sont les applications qui l'informent via le protocole X
 - ▶ des bibliothèques applicatives pour dessiner boutons, menu, assesseurs, etc. (*graphic toolkits*)
- ▶ Le serveur X reçoit tous les événements clavier et souris
 - ▶ il informe alors les applications de l'événement si celles-ci ont demandé qu'il leur soit envoyé
 - ▶ les applications traitent l'événement et décident de ce qu'il faut faire, elles demandent éventuellement des modifications d'affichage au serveur

- ▶ Le gestionnaire de fenêtre ou *Window Manager*
 - ▶ le serveur X **ne sait pas gérer** les fenêtres!!!!
 - ▶ une application spécifique est nécessaire pour
 - ▶ offrir des menus de fond d'écran
 - ▶ permettre de déplacer, iconifier, restaurer, supprimer les fenêtres et aussi modifier leur taille
 - ▶ C'est le **Window Manager**
 - ▶ Les Window Managers existent en grand nombre, il y en a pour tous les goûts
 - ▶ Les applications sont normalement compatibles avec tous les Window Managers (vœu pieux!)