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

F2B002C - Introduction à Unix/Linux

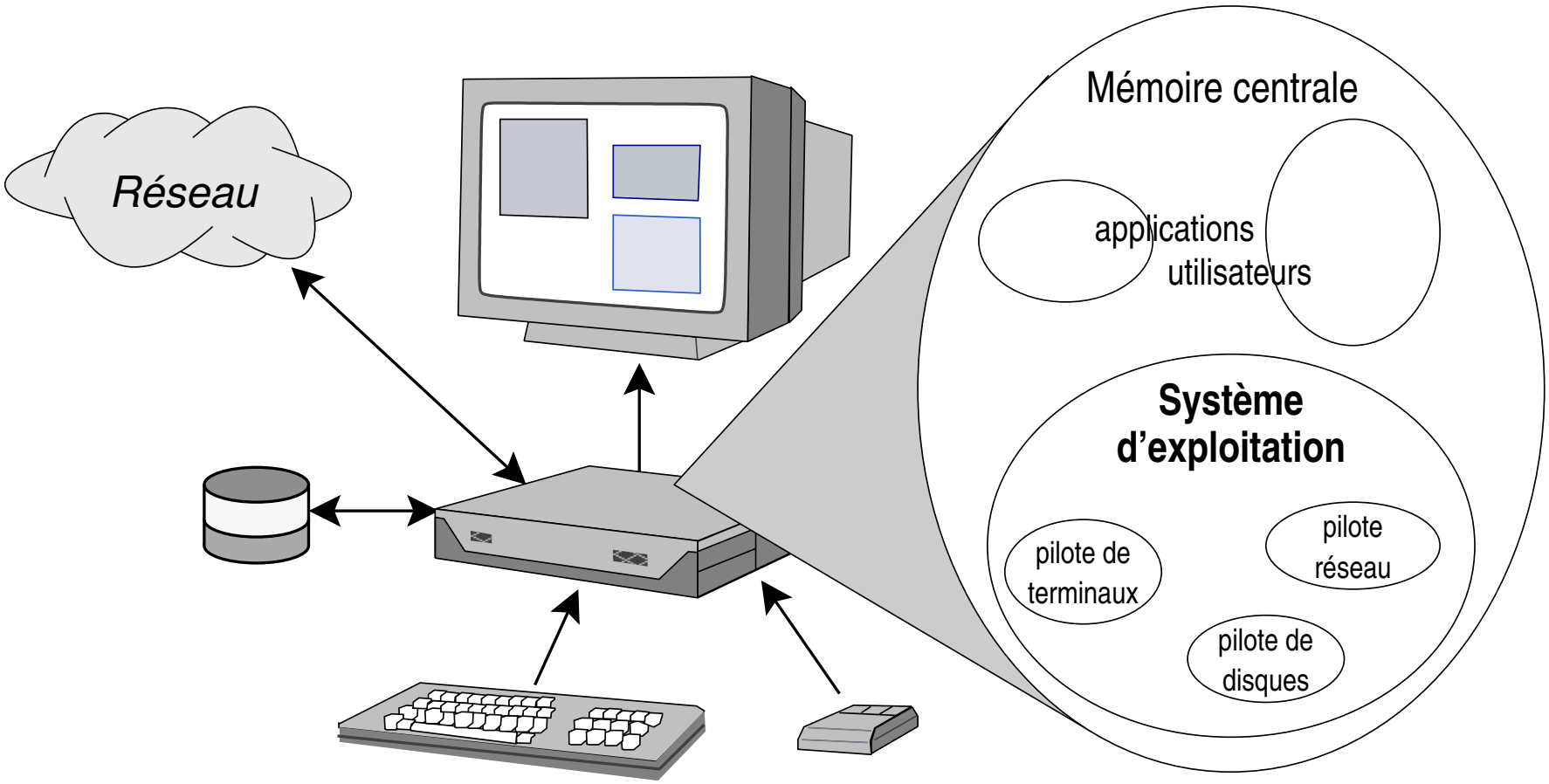
Christophe Lohr

Automne 2018

Sommaire

Sommaire

Unix/Linux : un système d'exploitation



Sommaire

Historique

- Unix naît officiellement le 1er 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)

Historique

- 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...

Historique

- Le bazar ou la cathédrale (livre d'Eric S. Raymond)
  - Cathédrale : développement encadré
  - Bazar : on part dans toutes les directions et on choisit la meilleure (Linus Torvalds)
- Le développement de Linux appartient au deuxième style
  - Tout le monde peut participer
  - Les codes sont libres, ouverts à tous et lisibles par tous et donc sûrs...
- Résultats
  - De nombreuses distributions
  - Les logiciels évoluent très rapidement
  - Il faut suivre ! ! ! ! !

Linux : un système Unix

- 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
- $\rightarrow$ 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

Les distributions

- Linux : un systèmes aux multiples couleurs... Comment s'y retrouver ?
- Plus d'une centaine de distributions !
- Que contient une distribution ?
  - Une méthode et un utilitaire d'installation
  - Un noyau parmi les noyaux stables les plus récents
  - Une bibliothèque libc parmi les plus récentes
  - Les commandes Unix standard
  - Des outils spécifiques d'administration
  - Des interfaces graphiques diverses et variées mais essentiellement Gnome et KDE

Compatibilité des distributions

- Noyau *interchangeables*
  - On doit pouvoir charger un nouveau noyau sans qu'il y ait d'impact sur les commandes du système et ses outils
- Organisation des fichiers *standardisée*
  - *Filesystem Hierarchy Standard* pour l'arborescence de base
- Commandes de base *standard*
  - Outils GNU, pas d'impact
- Administration *souvent spécifique !*
  - Installation
  - Outils d'administration courante
  - Gestion des paquetage (mais il existe des familles de distribution utilisant les mêmes outils)
  - Cependant les procédures d'initialisation sont généralement basées sur Unix Système-V (/etc/inittab), il est donc relativement aisé de comprendre «comment ça marche»

Distributions

- Stations de travail
  - Debian, Ubuntu, RedHat, Mandrake, Suse, etc.
- Spéciales temps réel
  - RT-Linux, RTLinux
- Mini distributions
  - Tiennent sur quelques disquettes voire une seule
  - FloppyFW, Tomsrtbt, Linux Router, LOAF, ...
- Embarqué
  - Uclinux, μLinux, PeeWeeLinux, ...
- Live-CD
  - Knoppix, Morphix, Kanotix, ...

Les familles de gestionnaires de paquetages

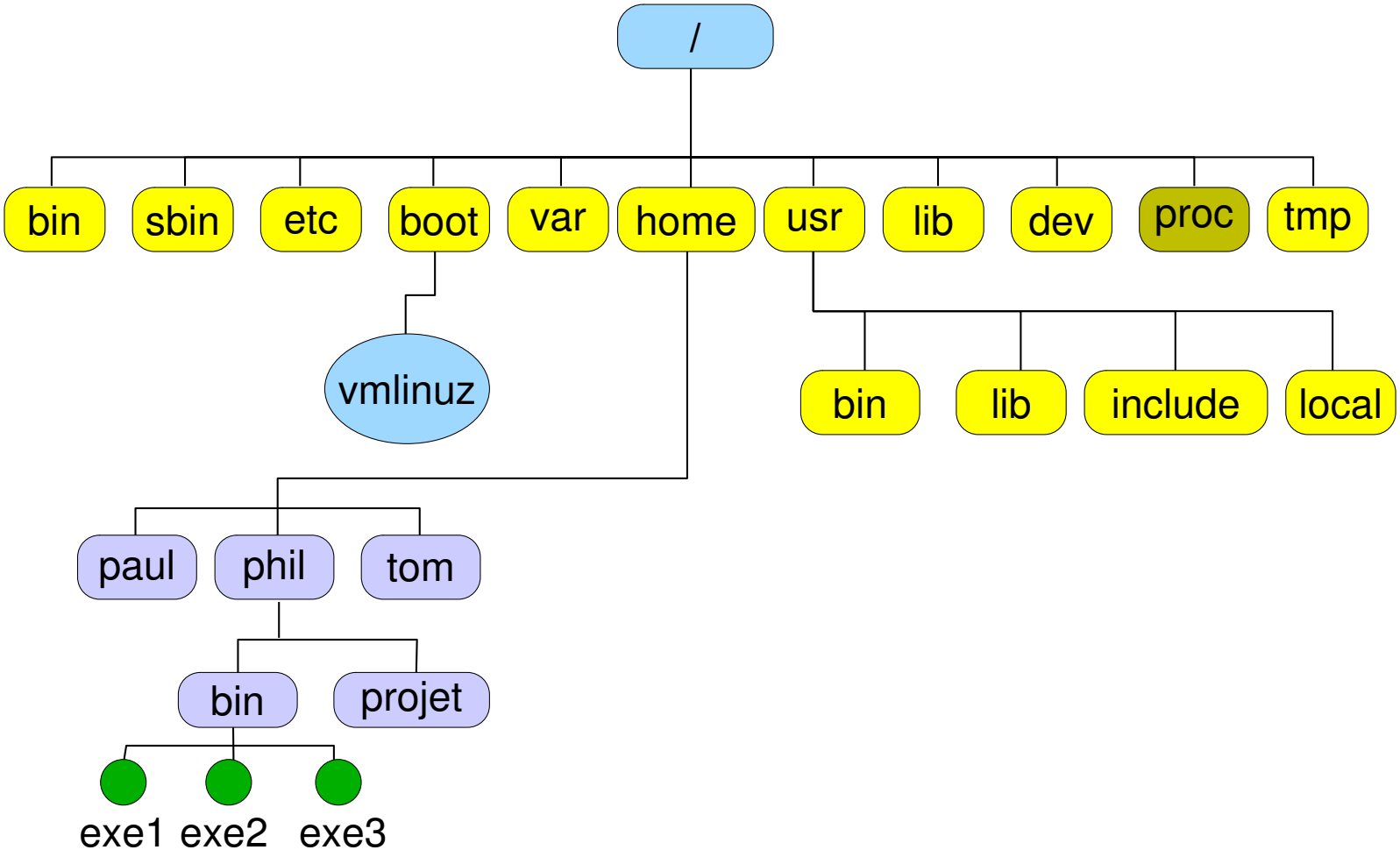
- Famille Redhat
  - le gestionnaire de paquetages `rpm`
  - **Red Hat Enterprise**, **Fedora**, **CentOS**, **Mandriva/Mandrake**, **Suse**, ...
- Famille Debian
  - Gestionnaire de paquetage `dpkg` / `apt` (complémentaires)
  - **Debian**, **Ubuntu**, **Knoppix**, **Morphix**, ...
- Paquetage au format tar compressé (`.tar.gz`, `.tgz`)
  - L'utilisateur final compile les *sources*
  - Format compatible avec toutes les distributions
  - Certaines n'acceptent que cela
  - Distribution **Slackware**
  - Variante **Gentoo** : paquetages `ebuild` (outil `emerge`)

Sommaire

Les fichiers Unix

- 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

L'arborescence du système de fichiers



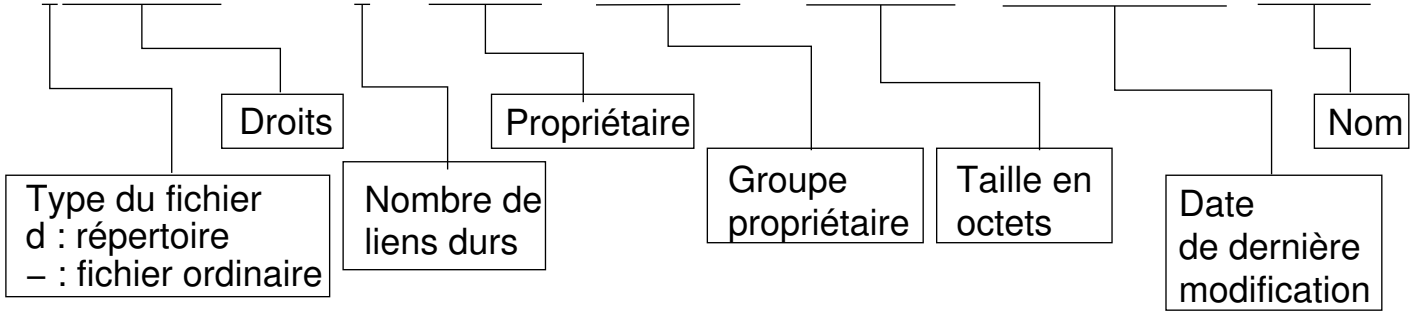
- Nommage abolu
  - par rapport à la racine, le nom commence par /
  - /home/phil/bin/exe1
- Nommage relatif
  - relatif au répertoire dans lequel on est :

home/phil/bin/exe1	si on est dans /
phil/bin/exe1	si on est dans /home
bin/exe1	si on est dans /home/phil
exe1	si on est dans /home/phil/bin

Visualiser le contenu d'un répertoire

- La commande ls
  - Exemple :

```
[bash]$ ls -l
total
-rwxr-xr-x    1   clohr   ens-rec     7790   avr 11 2005 essai
-rw-rw-r--    1   clohr   ens-rec     1122   avr 24 2005 essai.c
-rw-rw-r--    1   clohr   ens-rec  9869052   avr 25 2005 test
```



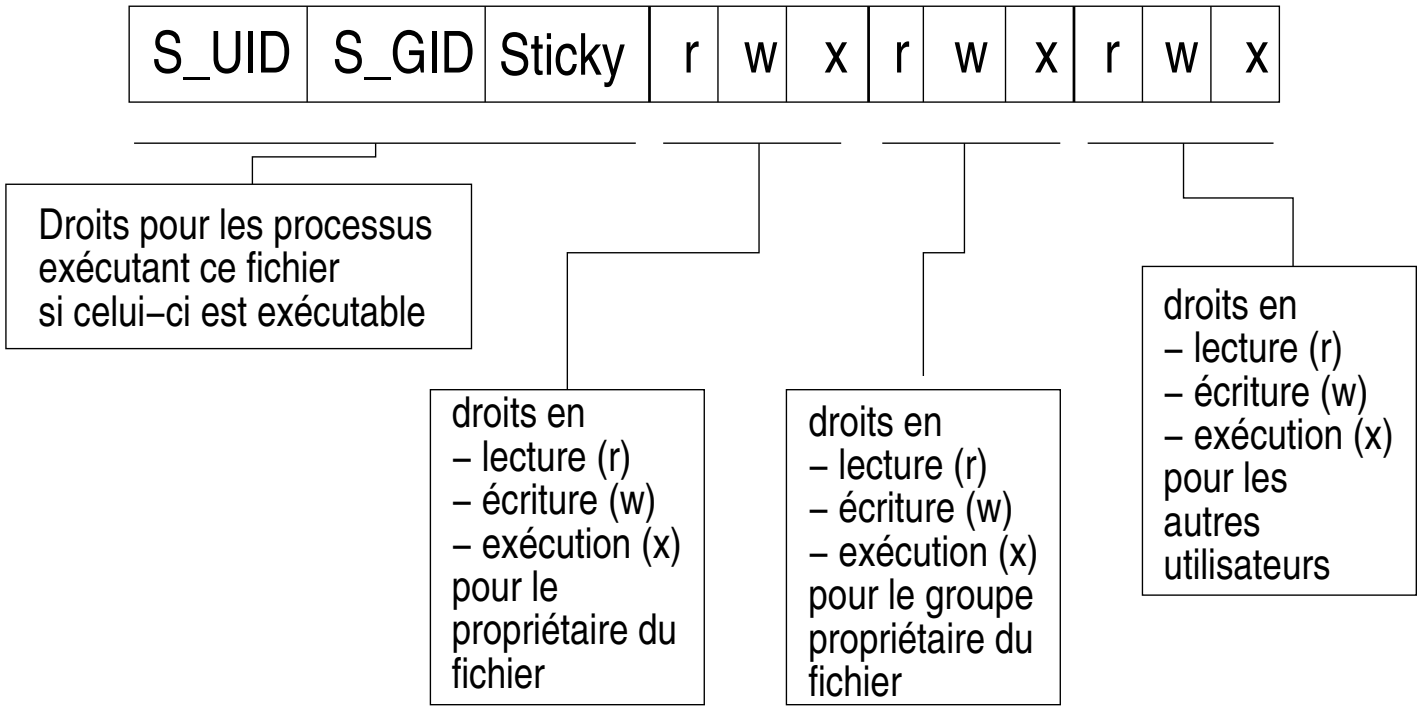
Les fichiers cachés

- 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

Les répertoires « . » et « .. »

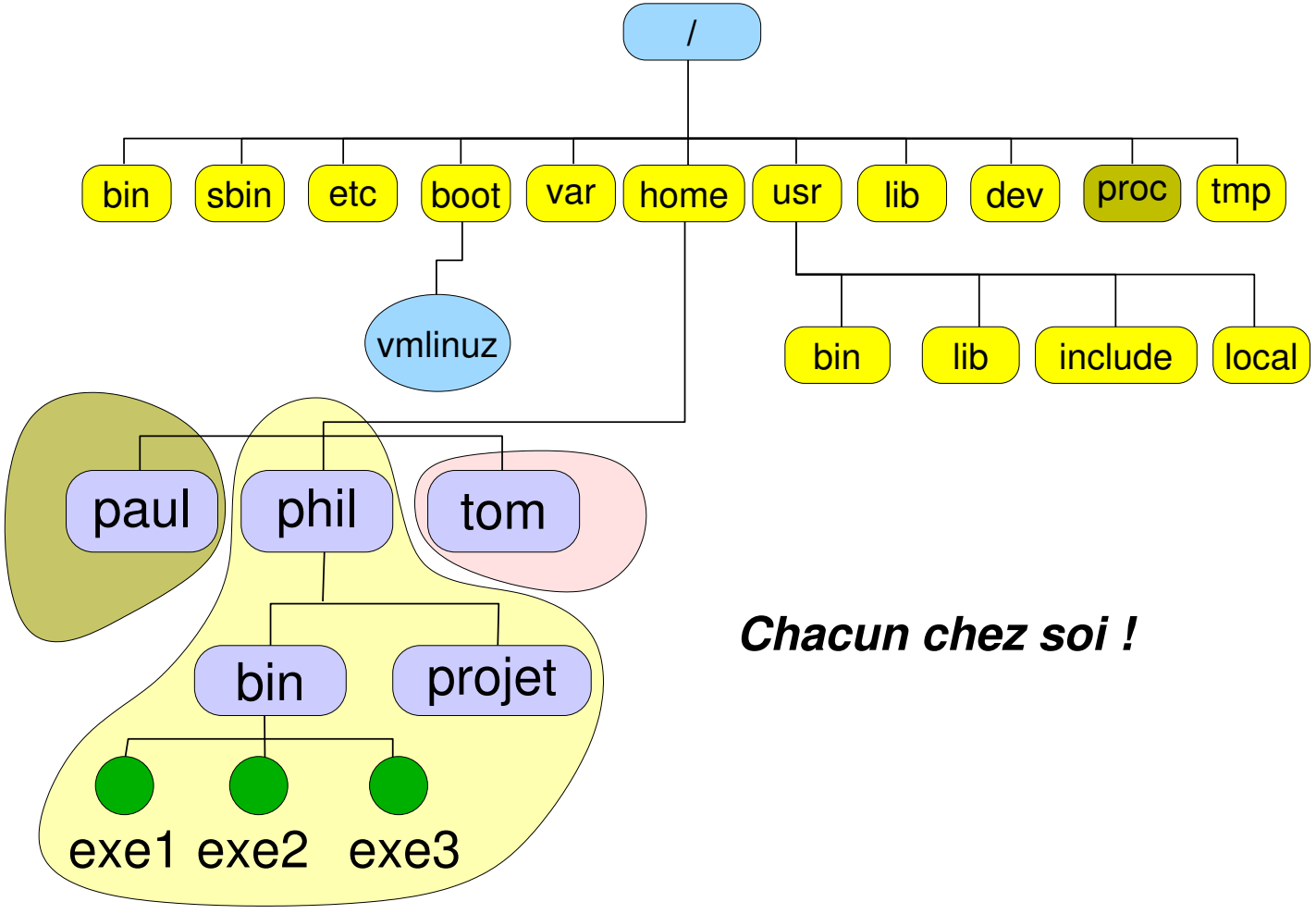
- Un répertoire n'est jamais vide, même à sa création, il contient déjà deux références sur des répertoires de nom « . » et « .. »
  - Le répertoire « . » (point) constitue une référence sur le répertoire lui-même
  - Le répertoire « .. » constitue une référence sur le répertoire immédiatement au dessus 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



Les droits sur les répertoires

- 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.



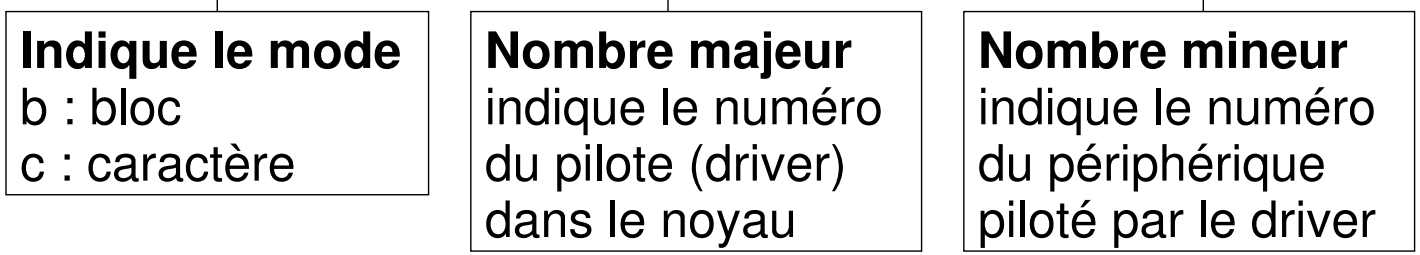
Chacun chez soi !

Les fichiers spéciaux

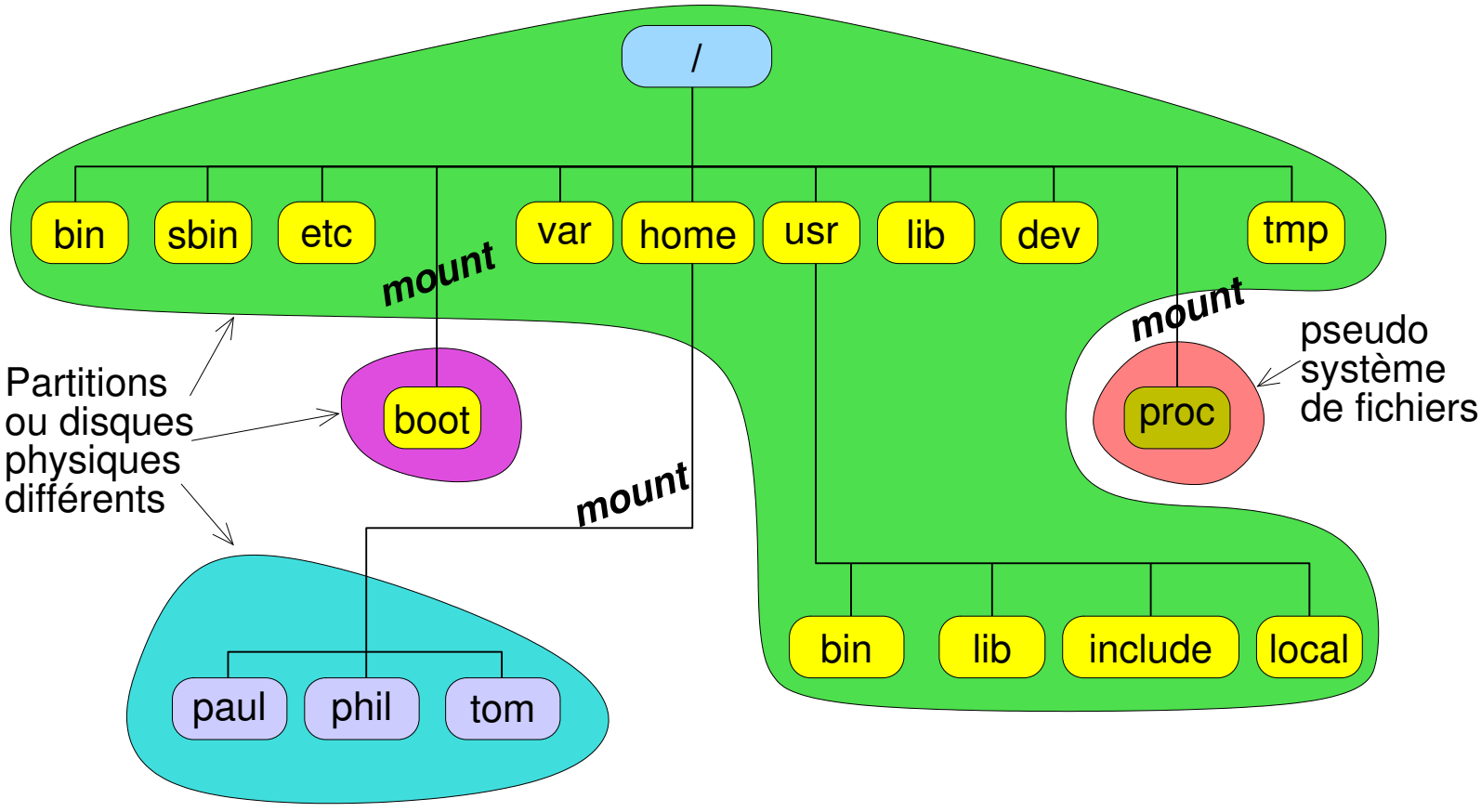
- Référencent des périphériques
  - Permettent les échanges (lectures/écritures) avec les pilotes des périphériques
  - Permettent le contrôle de ceux-ci
- Deux types
  - Les périphériques en mode bloc
    - les échanges se font par bloc d'octets (par «pages»)
  - Les périphériques en mode caractère 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
```



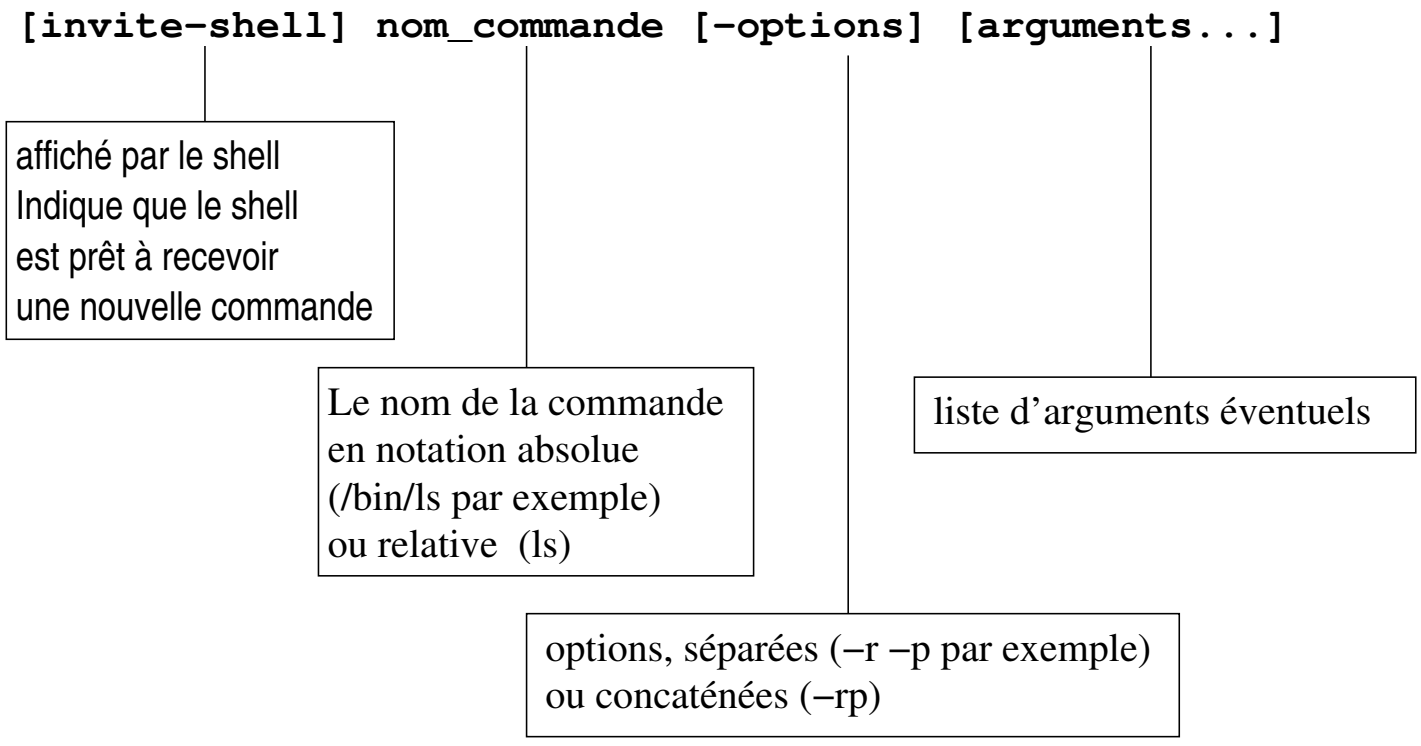
Le système de fichiers global



Comment lancer une commande

- 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



- 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 caractère de `a` à `z` (code ASCII)
  - `{ab,ac}` : la chaîne `ab` ou `ac`

Les commandes Unix classiques

(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`

Les commandes Unix classiques

...

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

- 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 opttions 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)

Les outils de recherche de fichiers

- 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

- 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)

Création d'un processus

- 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

Les variables d'environnement

- 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`, ...

Identité utilisateur associé au processus

- 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 (*euid : 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*)

Les fichiers standards d'entrée-sortie

- 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

Contrôle sur les processus

- 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)

Contrôle sur les processus : les signaux

- 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

Les principaux signaux

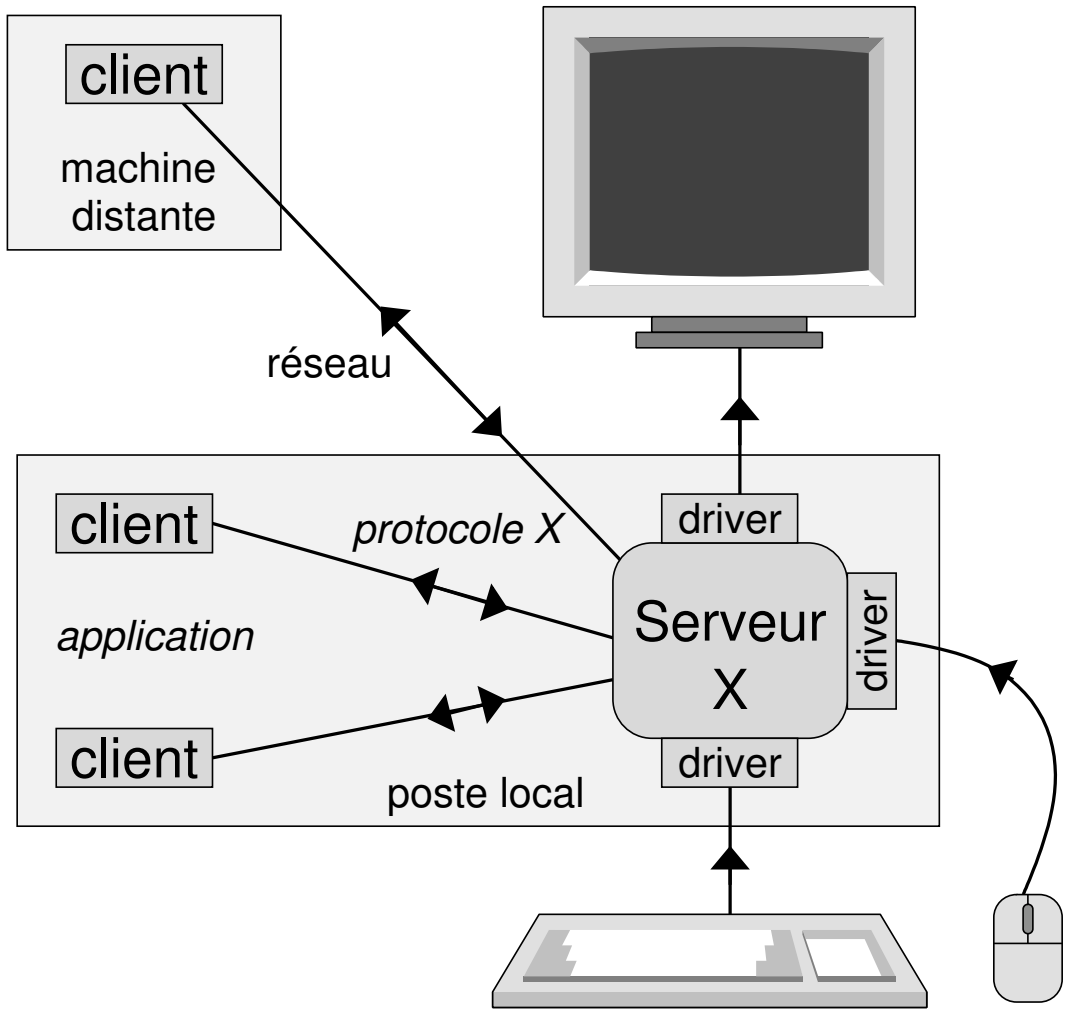
Signal	Numéro	Fonction
HUP	1	Signal envoyé au processus en premier plan associé à un terminal lorsque celui-ci est fermé
INT	2	Envoyé depuis le clavier avec la combinaison de touches <CTRL-C> (par défaut)
QUIT	3	Envoyé depuis le clavier avec la combinaison de touches <CTRL-|> (par défaut)
KILL	9	Ne peut 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

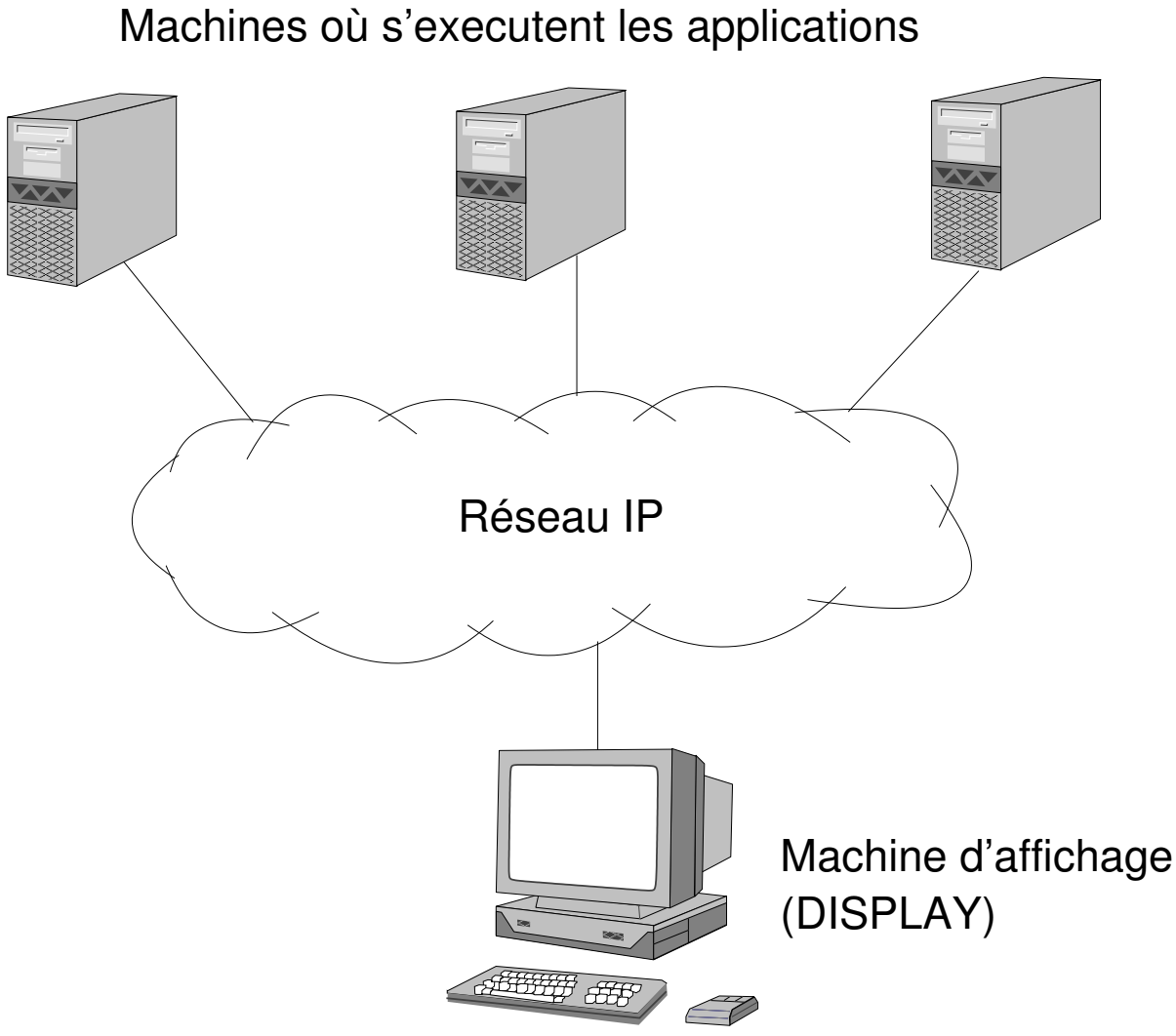
La commande `kill`

- `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

Les processus zombies

- Lorsqu'un processus fils se termine, son père doit acquitter la terminaison. C'est un problème de programmeur, pas d'utilisateur. Si l'acquittement 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 acquitté 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


client
machine
distante
réseau
client
protocole X
application
client
poste local
driver
Serveur
X
driver
driver


Machines où s'executent les applications
Réseau IP
Machine d'affichage
(DISPLAY)

- 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`
  - ⇒ autentifie un utilisateur (qui doit posséder le bon *cookie* de 128 bits)

X-Window - Le serveur et les applications

- 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 librairies applicatives pour dessiner boutons, menu, assenceurs, 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

X-Window - Une application très particulière

- 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 !)

Sommaire

Les scripts shell

- Deux familles : Bourne Shell et C-Shell
- Standard (Bourne) : `/bin/sh` (voir `man sh (1)`)
- Interpréteur de commandes (terminal ou fichier)
- Possède
  - des commandes internes
  - des structures de contrôle
- Manipule des variables

Attente d'une commande, lecture, analyse de la commande

Si la commande est une ***commande interne***

    Alors exécution la commande

Sinon Si ***fonction***

    Alors exécution la suite de commandes de la **fonction** (dans le shell courant)

Sinon

    Création d'un processus fils

    Lancement de la commande par le processus fils

    Attendre la fin

Finsi

Commandes internes (ex. en Bash)

- `cd` : changer de répertoire
- `pwd` : afficher le répertoire courant
- `echo` : afficher à l'écran
- `read` : lecture
- `export` : exportation de variables
- `eval` : évaluation de commande
- `exec` : exécution de commande
- `wait` *n* : attente de la fin du processus n
- `exit` *n* : sortie en renvoyant le code n
- `trap` *commande n* : détection et exécution au signal n
- `break` : sortie de boucle
- `set` : positionnement de variable
- `shift` : décalage des arguments
- `unset` : annulation de variable
- `help` *commande* ...

Fonctions et alias du shell

- Exécutent des suites de commandes dans le **shell courant** (pas de création de processus fils)
- Syntaxe des fonctions :
  - `function () { liste de commandes ... }`
- Utiles pour
  - configurer l'environnement interactifs
  - l'écriture de script
- Les alias
  - Permettent de renommer des commandes en les paramétrant
  - Création de nouvelles commandes
  - Exemple : `alias rm='rm -i'`

Les variables du shell

- Syntaxe : 1er caractère : une lettre ou un souligné, puis lettre(s), chiffre(s) ou souligné(s)
- Affectation
  - `VAR=‘pwd‘`
  - `VAR=`*`chaîne`*
  - `VAR="`*`chaîne de caracteres`*`"`
  - `VAR=`*`12`*
- Les guillemets (*double-quote*) masquent l'interprétation des caractères blanc (espace) et Tab
- Le caractère «\» masque l'interprétation du caractère qui le suit
- Les accents aigus (*single quote*) masquent l'interprétation de tous les caractères

Les variables du shell

- Accès : `$VAR`
- Variables d'environnement
  - `PATH` : liste des chemins accessibles
  - `HOME` : répertoire par défaut (avant-dernier champ de `/etc/passwd`
  - `PS1` : 1er prompt (généralement le caractère `$`)
  - `PS2` : 2ème prompt (généralement le caractère >)
  - `IFS` : séparateur de champ (blanc par défaut)

Variables systèmes

- ? : code de retour de la dernière commande
- # : nombre de paramètres passées à la commande en cours
- $: numéro du PID du shell courant
- ! : numéro du PID du dernier shell lancé en background

Transmission de variables

- Affectation à l'appel de la procédure
  - exemple : `prompt> TERM=vt100` *`nom_de_la_procédure`*
- Paramètres positionnels (arguments)
  - `prompt>` *`nom_de_la_procédure`* `ARG1 ARG2 ARG3...`
    accessibles depuis l'intérieur du script par les variables définies par leur position $1, $2, $3, ...
    $0 est le *nom_de_la_procédure*

Visibilité des variables

- Les variables sont locales au script sauf si elles ont été explicitement exportées
- `export VAR`
- N.B. `VAR` et non `$VAR` : c'est la variable qui est exportée et non son contenu

La boucle for

- Syntaxe :
  for VAR **in** w1 w2 w3 ... w*n*
  do
  liste de commandes
  done
- autre écriture :
  for i do ... done
  `$i` prendra les valeurs des paramètres positionnels

La structure case

- Syntaxe :

```
case $VAR in
 cas1 ) liste de commandes ;;
 cas2 ) liste de commandes ;;
 ...
 casn ) liste de commandes ;;
esac
```

- Intégrations des cas :
  - **[c1-c2]** branchement si la variable est composée d'un seul caractère compris entre c1 et c2
  - **[c1c2c3]** branchement si la variable testée est composée d'un seul caractère égal à c1 c2 ou c3
  - **[xx|yy|zz]** branchement si la variable testée est composée de 2 caractères xx, yy ou zz

Structure `if-then-else`

- Syntaxe :
  if *liste de commandes*
  then
  liste de commandes
  else
  liste de commandes
  fi
- le test du if porte **sur le code retour de la dernière commande**

La commande `test`

- Test numérique :
  - `test X` *`option`* `Y`
  - *option* : `-eq`, `-ne`, `-gt`, `-ge`, `-le`
- Test sur les fichiers et répertoires
  - `test` *`option`* `fichier`
  - *option* : `-s` fichier existe et n'est pas vide, `-d` fichier est un répertoire, `-f` fichier est ordinaire, `-w` fichier a le droit en écriture, `-r` fichier a le droit en lecture
- Test sur les chaînes de caractères
  - `test S1` *`option`* `S2`
  - *option* : = ou `!=`
  - `test` *`option`* `S`
  - `-z` : longueur 0 ; `-n` vérifie que la chaîne a une longueur non nulle

La boucle `while`

- Syntaxe :
  while *liste de commandes* **do**
  liste de commandes
  done
- NB : La boucle **while** est exécutée tant que le code retour de la dernière commande de la condition est vrai

Les substitutions

- **$*** ou **$@** tous les paramètres positionnels
- **${var}** identique à `$var` (sert à la concaténation)
- **${var-chose}** si var est définie rend `var` sinon `chose`
- **${var=chose}** si var est définie rend `var` sinon affectation de `chose` à `var`
- **${var ?message}** si `var` est définie rend `var` sinon apparition du message
- **‘accent grave‘** : `var = ‘date‘`

Méta caractères du shell

- ***** : remplacé par un nb quelconque de caractères pris dans les fichiers du répertoire
- **?** : remplacé par 1 seul caractère
- **[...]** : tout caractère compris entre les crochets sera pris en compte s'il existe dans les noms des fichiers du répertoire de travail
- **"** : le guillemet permet de passer plusieurs arguments mais n'empêche pas l'interprétation des caractères spéciaux
- **'** : la quote interdit au shell d'interpréter les caractères spéciaux
- \ : interdit l'interprétation du caractère suivant
- **()** : permet de grouper des commandes (nouveau shell)

Les redirections

- $>$: Redirection de la sortie standard
- $>>$: Redirection de la sortie standard en mode append
- $<$: Substitution de l'entrée standard
- $<<$: Substitution temporaire de l'entrée standard (délimitée par 2 chaînes de caractères)
- $|$: *Pipe*, connecte une sortie sur le fichier standard d'entrée d'une commande

Comprendre un script shell


```
Last login: Thu Sep  7 14:08:01 2006 from plohr.enst-bretagne.fr
Kickstart-installed fedora core mer aoû 23 09:11:12 CEST 2006
pc-info-104.enst-bretagne.fr[1]>SETUP

          SETUP possible pour les logiciels suivants :
# ECLIPSE5      Eclipse 5.6_44
# JAVA15        Java 2 SDK Standard Edition (J2SE) v1.5.0_08
# JGRASP        JGrasp 1.8.4
# LATEX2E       Ajout styles non compris dans la distribution te
# MATLABR14SP2  Matlab R14 SP2
(../..)

Ce shell est deja configure pour CSHRC
pc-info-104.enst-bretagne.fr[2]>
```

Comprendre un script shell


```
Last login: Thu Sep  7 14:08:01 2006 from plohr.enst-bretagne.fr
Kickstart-installed fedora core mer aoû 23 09:11:12 CEST 2006
pc-info-104.enst-bretagne.fr[1]>SETUP

          SETUP possible pour les logiciels suivants :
# ECLIPSE5      Eclipse 5.6_44
# JAVA15        Java 2 SDK Standard Edition (J2SE) v1.5.0_08
# JGRASP        JGrasp 1.8.4
# LATEX2E       Ajout styles non compris dans la distribution te
# MATLABR14SP2  Matlab R14 SP2
(../..)

Ce shell est deja configure pour CSHRC
pc-info-104.enst-bretagne.fr[2]>
```

Comprendre un script shell


```
Last login: Thu Sep  7 14:08:01 2006 from plohr.enst-bretagne.fr
Kickstart-installed fedora core mer aoû 23 09:11:12 CEST 2006
pc-info-104.enst-bretagne.fr[1]>SETUP

          SETUP possible pour les logiciels suivants :
# ECLIPSE5      Eclipse 5.6_44
# JAVA15        Java 2 SDK Standard Edition (J2SE) v1.5.0_08
# JGRASP        JGrasp 1.8.4
# LATEX2E       Ajout styles non compris dans la distribution te
# MATLABR14SP2  Matlab R14 SP2
(../..)

Ce shell est deja configure pour CSHRC
pc-info-104.enst-bretagne.fr[2]>
```

Comprendre un script shell

```
pc-info-104.enst-bretagne.fr[2]>SETUP -v LATEX
=====
# LATEX         Ajout styles non compris dans la distribution te

setenv TEXMF "{/usr/share/texmf,/usr/local/lib/texmf}"

rehash
=====
pc-info-104.enst-bretagne.fr[3]>which SETUP
SETUP:   a comme alias eval  `/usr/home/enstb1/etc/Setup !*`
pc-info-104.enst-bretagne.fr[4]>cat /usr/home/enstb1/etc/Setup
```

Comprendre un script shell


```
pc-info-104.enst-bretagne.fr[2]>SETUP -v LATEX
=====
# LATEX         Ajout styles non compris dans la distribution te

setenv TEXMF "{/usr/share/texmf,/usr/local/lib/texmf}"

rehash
=====
pc-info-104.enst-bretagne.fr[3]>which SETUP
SETUP:   a comme alias eval  `/usr/home/enstb1/etc/Setup !*`
pc-info-104.enst-bretagne.fr[4]>cat /usr/home/enstb1/etc/Setup
```

Comprendre un script shell

```
pc-info-104.enst-bretagne.fr[2]>SETUP -v LATEX
=====
# LATEX          Ajout styles non compris dans la distribution te

setenv TEXMF "{/usr/share/texmf,/usr/local/lib/texmf}"

rehash
=====
pc-info-104.enst-bretagne.fr[3]>which SETUP
SETUP:   a comme alias eval  `/usr/home/enstb1/etc/Setup !*`
pc-info-104.enst-bretagne.fr[4]>cat /usr/home/enstb1/etc/Setup
```

Comprendre un script shell

```
pc-info-104.enst-bretagne.fr[2]>SETUP -v LATEX
=====
# LATEX          Ajout styles non compris dans la distribution te

setenv TEXMF "{/usr/share/texmf,/usr/local/lib/texmf}"

rehash
=====
pc-info-104.enst-bretagne.fr[3]>which SETUP
SETUP:   a comme alias eval  `/usr/home/enstb1/etc/Setup !*`
pc-info-104.enst-bretagne.fr[4]>cat /usr/home/enstb1/etc/Setup
```

```
pc-info-104.enst-bretagne.fr[2]>SETUP -v LATEX
=====
# LATEX         Ajout styles non compris dans la distribution te

setenv TEXMF "{/usr/share/texmf,/usr/local/lib/texmf}"

rehash
=====
pc-info-104.enst-bretagne.fr[3]>which SETUP
SETUP:   a comme alias eval  `/usr/home/enstb1/etc/Setup !*`
pc-info-104.enst-bretagne.fr[4]>cat /usr/home/enstb1/etc/Setup
```

```
pc-info-104.enst-bretagne.fr[2]>SETUP -v LATEX
=====
# LATEX          Ajout styles non compris dans la distribution te

setenv TEXMF "{/usr/share/texmf,/usr/local/lib/texmf}"

rehash
=====
pc-info-104.enst-bretagne.fr[3]>which SETUP
SETUP:   a comme alias eval  `/usr/home/enstb1/etc/Setup !*`
pc-info-104.enst-bretagne.fr[4]>cat /usr/home/enstb1/etc/Setup
```

Le script SETUP (1/6)


```
#!/bin/ksh
#########################################
# Setup standard (CLB/AB) 19/12/97      #
#########################################
dirsetup=/usr/home/enstb1/etc/setup
##############
dirssetup=$dirsetup:$SETUPPATH:-""
#############
PATH=/usr/5bin:/bin:/usr/bin:/usr/ucb #pour SOLARIS(ne gene pas S
departement=enstb
ok=1
```

(../..)

Le script SETUP (2/6)

(../..)

```
case "$1" in
        "")
                reponse=readme
                ok=0;;
        -v | -V)
                reponse=voir
                shift
                echo "setenv VOIRSETUP \"$*\" ; \c"
                ok=0;;
        *)
                reponse=$*;;
esac
```

(../..)

Le script SETUP (3/6)

(../..)

```
if [ $ok -eq 1 ]
then
        quel=$QUELSETUP
        IFS=":"
        for clb in $quel
        do
                if [ $clb = $reponse ]; then
                        echo "echo Environnement $reponse deja in
                        exit
                fi
        done
fi
```

(../..)

Le script SETUP (4/6)

(../..)

```
IFS=":"
for dir in $dirssetup
do
        if [ -f "$dir/$departement.$TYPESTATION.$reponse" ]
        then
                setupdir=$dir
                break
        fi
done
```

(../..)

Le script SETUP (5/6)

(../..)

```
if [ -f "$setupdir/$departement.$TYPESTATION.$reponse" ]
then
        echo "source $setupdir/$departement.$TYPESTATION.$reponse
        case $reponse in
                CSHRC)          quel="";;
                LOGIN)          ok=0;;
                LOGIN+)         ok=0;;
                readme)         ok=0;;
                voir)           ok=0;;
        esac
else
        echo "echo Pas de SETUP pour $reponse sur $TYPESTATION ;
fi
```

(../..)

Le script SETUP (6/6)

(../..)

```
if [ $ok -eq 1 ]; then
        if [ "$quel" = "" ]
        then
                echo "setenv QUELSETUP $reponse ; \c "
        else
                echo "setenv QUELSETUP $reponse$quel ; \c "
        fi
fi
echo ""
```

Logiciels sources au format général GNU

- Téléchargeables sous forme de fichier de type archives `tar` compressées avec `gzip` (`.tgz`, `.tar.gz`) ou `bzip2` (`.bz2`)
  - `tar xvf` *`paquetage`*
- Contiennent un script de configuration et de création des `Makefiles` adaptés à l'architecture et à la version du système : `configure`
- Configuration, compilation, installation
  - `[bash]$ ./configure [--options]`
  - `[bash]$ make`
  - `[bash]$ make install`

- Trois niveaux d'utilitaires : `aptitude`, `apt`, `dpkg`
  - `dselect`/`aptitude`/`synaptic` offrent une interface texte ou graphique et permet de configurer les moyens de recherche des paquetages, de faire des suggestions, de les installer, les mettre à jour et les désintaller
  - lorsque l'on connait très exactement ce que l'on veut installe/désintaller il est plus rapide d'utiliser les commandes apt : `apt-get`, `apt-cache`, ...
  - `dpkg` pour manipuler un fichier de paquetage déjà sur le disque, ex. : lister le contenu d'un paquetage : `dpkg -L` *`nomDuPackage`*

- Exemples :

```
linux# apt-cache search linuxconf
linuxconf - a powerful Linux administration kit
linuxconf-x - X11 GUI for Linuxconf
linuxconf-dev - Development files for Linuxconf
linuxconf-i18n - international language files for Linuxcon
linux#
```

Packages Debian - Installation d'un logiciel


```
linux# apt-get install linuxconf-x
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
libwxxt1
The following NEW packages will be installed:
libwxxt1 linuxconf-x
0 packages upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 532kB of archives. After unpacking 1438kB will be used.
Do you want to continue? [Y/n] Y
Get:1 ftp://172.16.19.2 stable/main libwxxt1 1.67c-6 [486kB]
Get:2 ftp://172.16.19.2 stable/main linuxconf-x 1.17r5-2 [45.8kB]
Fetched 532kB in 1s (499kB/s)
Selecting previously deselected package libwxxt1.
(Reading database ... 30948 files and directories currently installed.)
Unpacking libwxxt1 (from .../libwxxt1_1.67c-6_i386.deb) ...
Selecting previously deselected package linuxconf-x.
Unpacking linuxconf-x (from .../linuxconf-x_1.17r5-2_i386.deb) ...
Setting up libwxxt1 (1.67c-6) ...
Setting up linuxconf-x (1.17r5-2) ...
linux#
```

Redhat Pakages Manager

- La commande `rpm`
- Permet d'installer (`-i`) ou de supprimer (`-e`) des logiciels :
  - `rpm -ivh nom_du_package`
  - Le nom du package peut être une URL
- Gère les dépendances entre logiciels (entre bibliothèques) : refuse d'installer si une dépendances n'existe pas (forçage possible mais dangereux)
- Permet de s'informer sur un *package*, savoir ce qu'il contient, de retrouver à quel *package* appartient tel fichier, de connaître les *packages* installés

...

- Permet de créer un *package* à partit d'une arborescence source compilée
- gestion de la base installée : `/var/lib/{rpm | rpm.rpmsave}`
- Commande `yum` recherche, télécharge et installe un paquetage

Où trouver les paquetages RedHat

- Sur les CD-ROM d'installation
  - si monté à l'endroit standard :
    - `/mnt/cdrom/Redhat/RPMS`
- Sur le web
  - `http://www.rpmfind.com`
- Outils systèmes
  - `gnorpm`
  - `yum`
  - ...