Introduction à Linux

version 1.0 (2005)

par Marc Gaëtano

Ecole polytechnique de l'université de Nice-Sophia Antipolis

Plan du cours (1)

I. Introduction

- Historique et caractéristiques
- L'environnement de travail
- L'interpréteur de commandes
- Aide et documentation

Plan du cours (2)

II. Principes et commandes de base

- Système de fichiers
- Commandes de base
- Liens symboliques et physiques
- Droits d'accès

Plan du cours (3)

III. Redirections et processus

- Entrées/sorties et redirections
- Les tuyaux
- Contrôle de tâche et de processus

Plan du cours (4)

IV. Compléments sur le shell

- Variables
- Substitution de commandes
- Alias
- La commande for

Plan du cours (5)

V. Le monde Linux

- Le projet GNU et la licence GPL
- Les distributions Linux
- Linux et ses concurrents

Plan du cours (6)

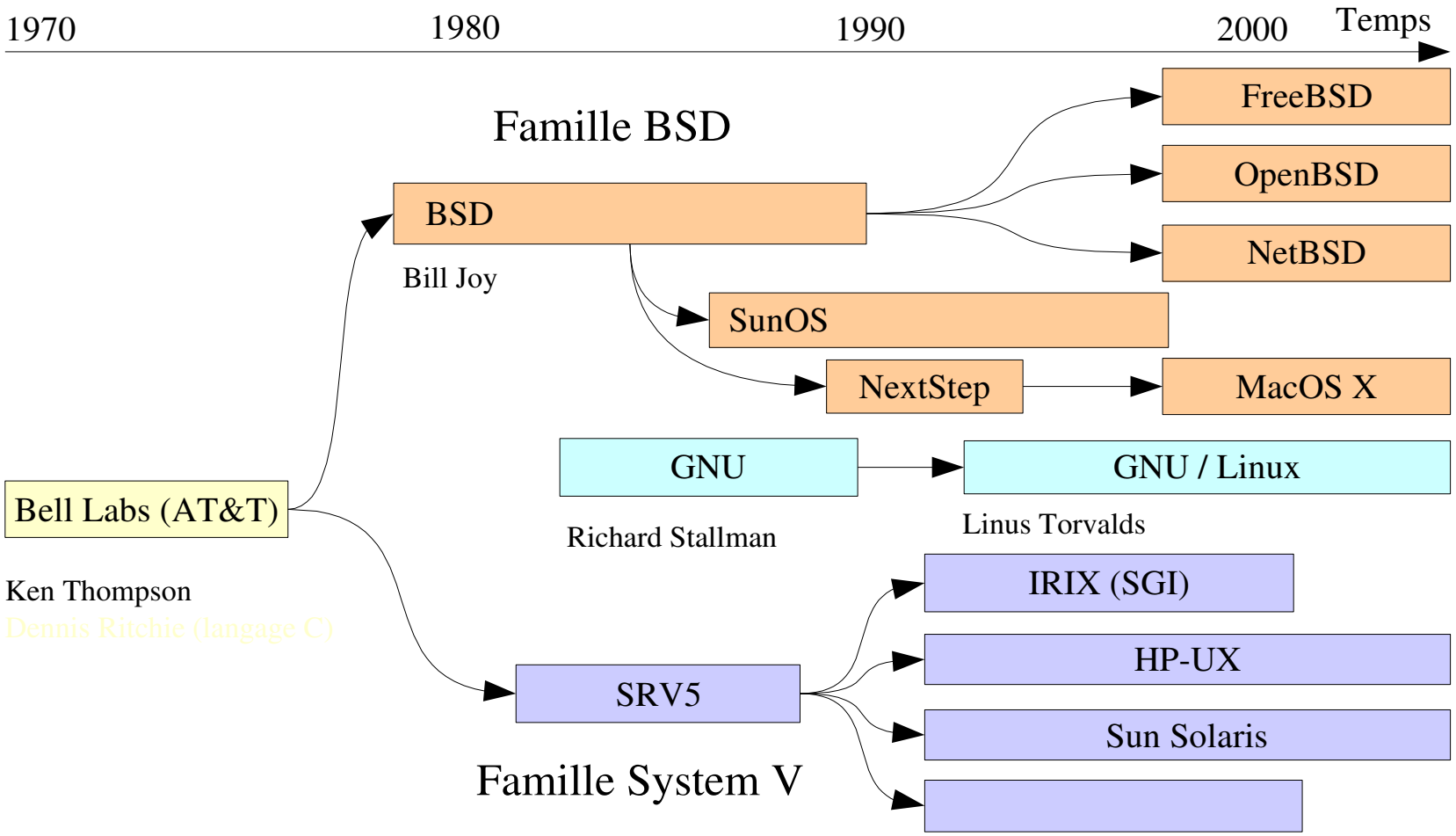
VI. Epilogue

- Ressources internet sur Linux
- Licence et historique du cours

I. Introduction

- Historique et caractéristiques
- L'environnement de travail
- L'interpréteur de commandes
- Aide et documentation

Historique d'Unix



La philosophie d'Unix

- “Small is beautiful”
- Faire que chaque programme fasse une seule chose et bien
- Conjuguer portabilité et efficacité
- Éviter les interfaces captives
- Plusieurs niveaux d'abstraction dans le système
  - Noyau: niveau matériel
  - Interpréteurs de commande: niveau texte
  - X Windows et Window Manager: niveau graphique

Principales caractéristiques d'Unix

Depuis le début dans les années 1970

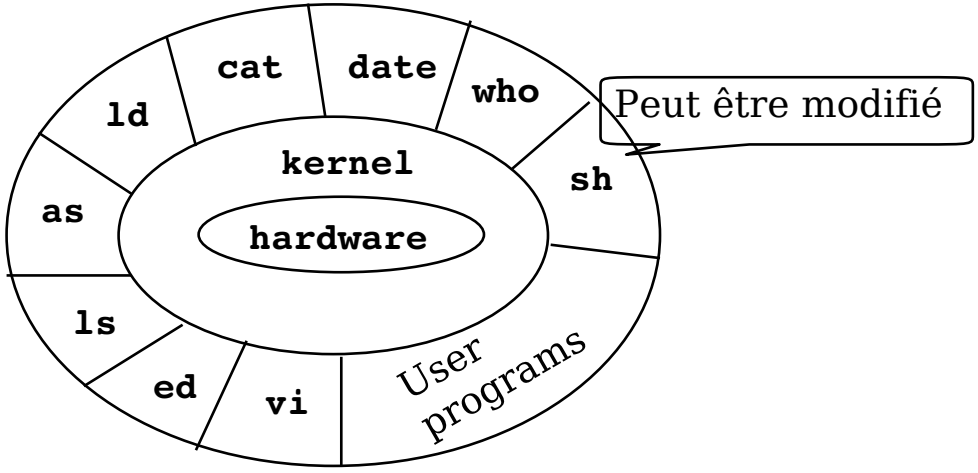
- Multi-utilisateur et sécurisé
  Par défaut, les utilisateurs ordinaires ne peuvent pas toucher aux fichiers d'autres utilisateurs.
  En particulier, ils ne peuvent ni modifier les paramètres du système, ni supprimer des programmes, etc.
  Utilisateur “root” : utilisateur administrateur, ayant tous les droits.
- Multi-tâche, prise en charge de multiples processeurs
- Extrêmement flexible
- Prise en charge du réseau

Couches dans un système Unix

- Noyau
- Bibliothèque C
- Bibliothèques système
- Bibliothèques d'applications
- Programmes utilisateurs :

  au même niveau que les

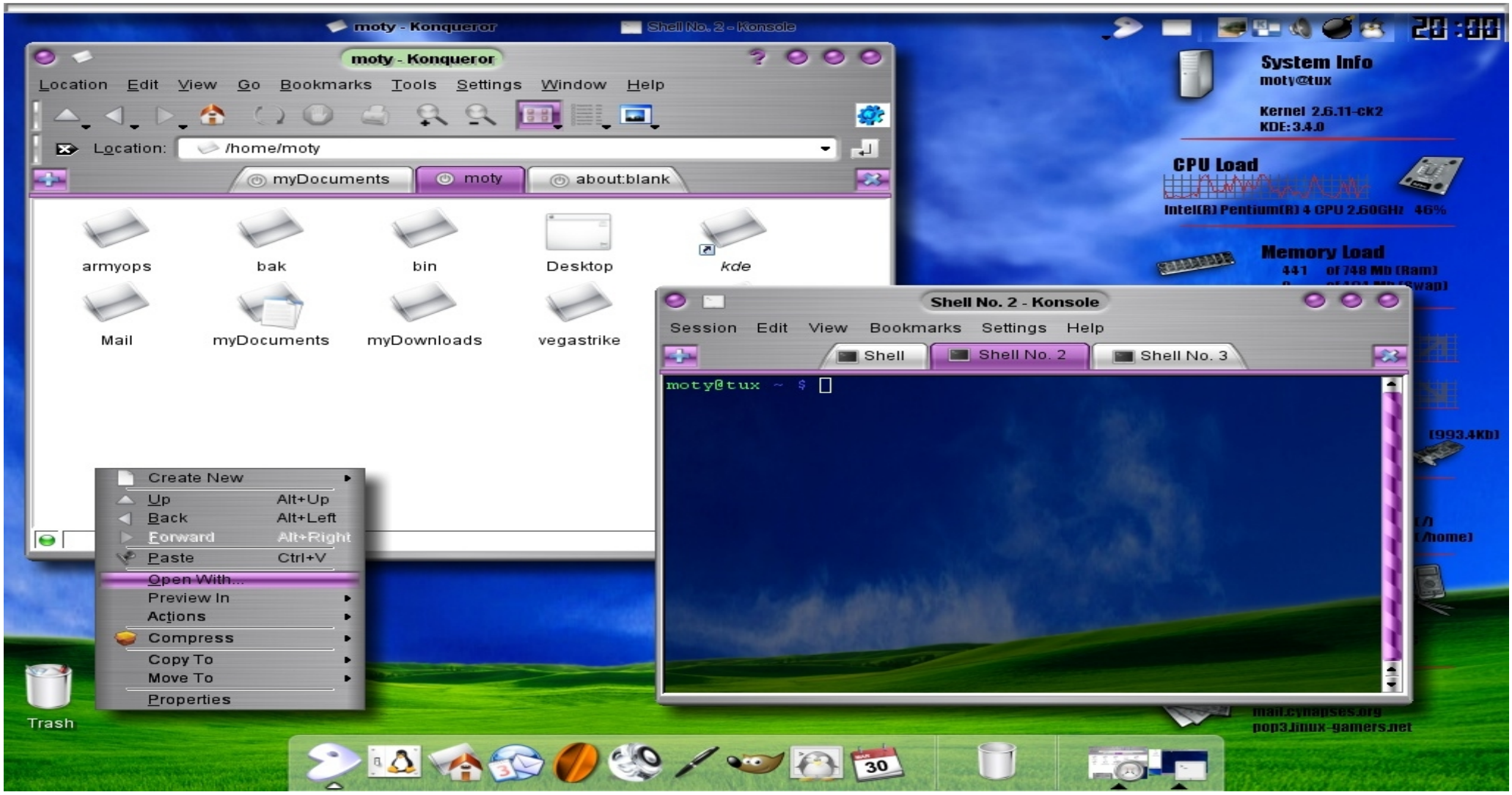
  commandes systèmes



Environnement de travail (1)

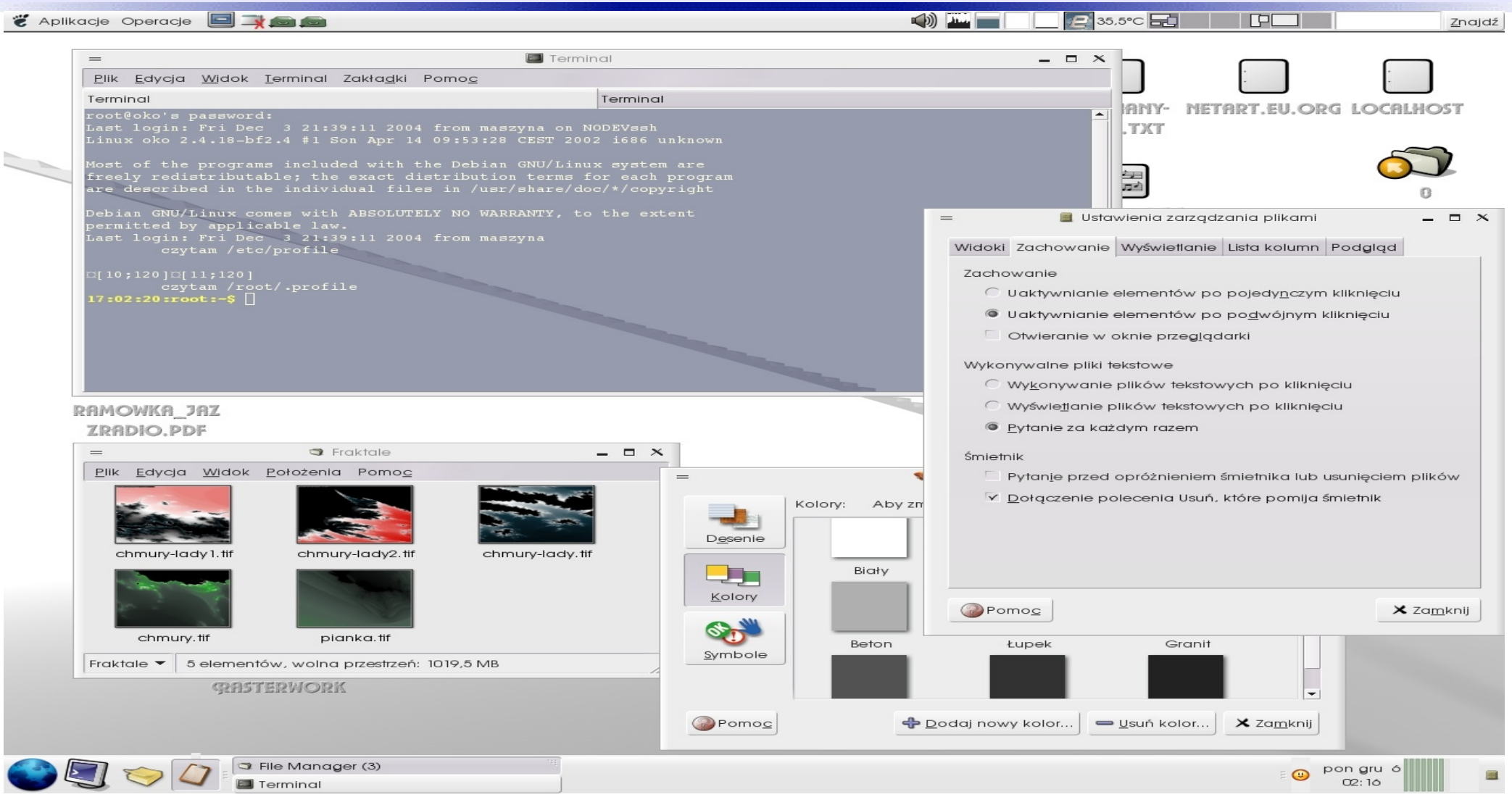
- L'accès aux fonctionnalités et aux données dans votre environnement de travail Linux s'effectue de deux façons complémentaires :
  - via une interface graphique : assurée par un Window Manager (comme GNOME ou KDE)
  - via un (ou plusieurs) terminal (ou terminaux) en mode texte constitué par une fenêtre dans laquelle on interagit avec le système
- Les fonctionnalités graphiques changent avec le Window Manager : elles constituent une interface graphique aux commandes de bases du système
- Les commandes de base ne changent pas : elles sont les mêmes (à quelques exceptions près) pour tous les systèmes Unix et sont standardisées
- Tout est paramétrable et programmable : l'environnement graphique est entièrement paramétrable et le système de base sous-jacent est entièrement programmable

Environnement de travail (2)



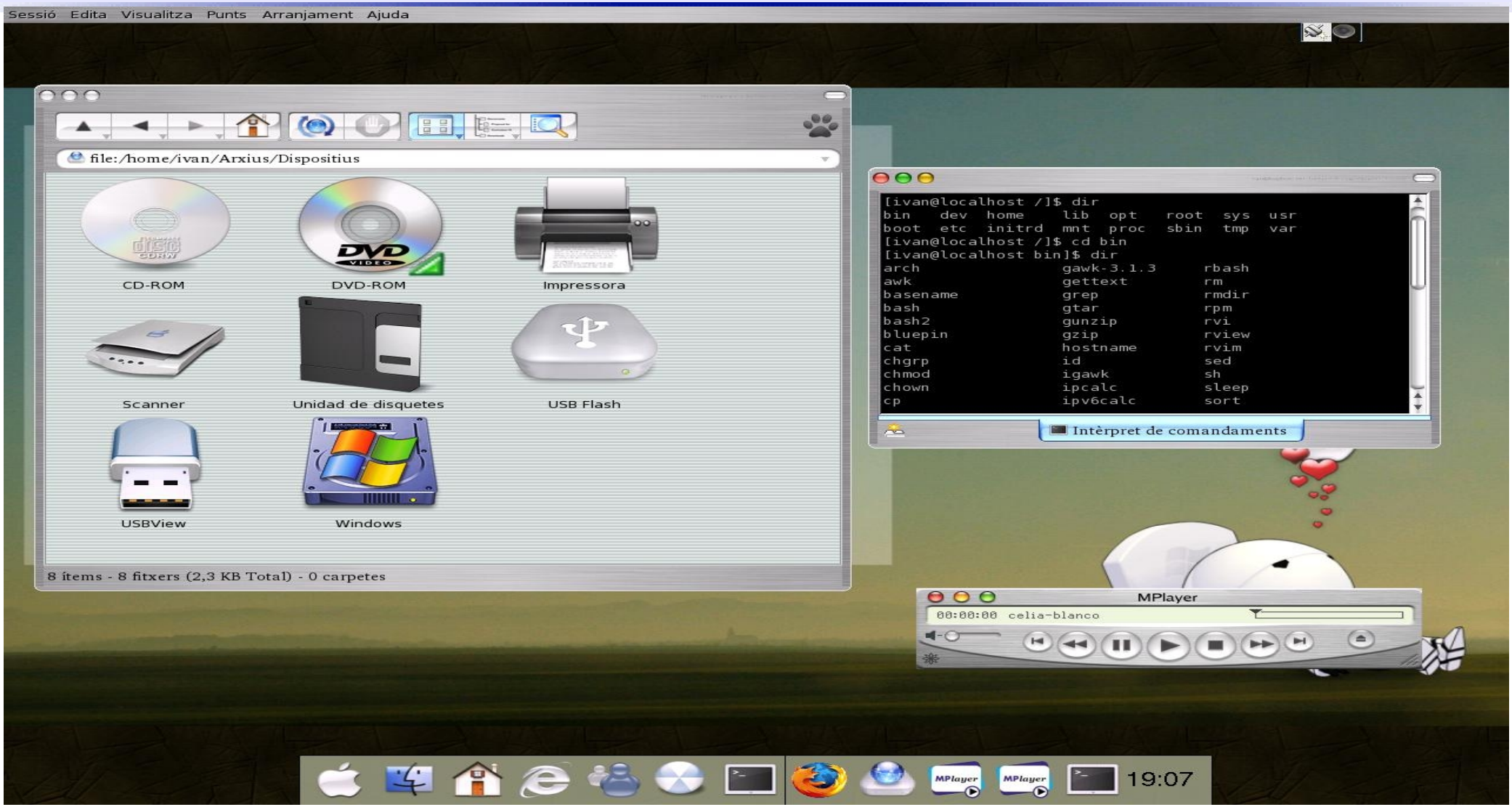
Le thème Metal4KDE pour KDE 3.2+ d'après Moty Rahamim

Environnement de travail (4)



Le thème Ana pour GNOME (GTK 2.x) d'après Tarchalski

Environnement de travail (5)



Le thème Mac OSX pour KDE 3.2+ d'après Ivanciko

Quelques applications standards

- **XEmacs**, **Kate** : éditeurs de texte
- **OpenOffice, KOffice** : suites bureautique complètes
- **Mozilla**, **Firefox**, **Konqueror**, **Nautilus** : navigateurs Internet
- **Evolution**, **Thunderbird**, **Kmail** : clients de messagerie
- **Kopete**, **Gaim** : clients de messagerie instantanée
- **GIMP** : éditeur graphique très puissant
- **Gwenview** : afficheur de galeries de photos
- **Amarok**, **XMMS**, **Noatun**, **Kaffeine** : lecteurs audio et multimédia
- **K3b**, **GnomeBaker** : graveurs de CD et DVD
- **aMule**, **KMLDonkey**, **KTorrent** : clients P2P

Éditeurs de texte

Un éditeur de texte est une application qui sert à fabriquer (et modifier) des fichiers de texte (ASCII ou autres)

Éditeurs de texte graphiques

- **Emacs**, **Xemacs** **(très vivement conseillé !)**
- **Kate** (uniquement sous KDE)

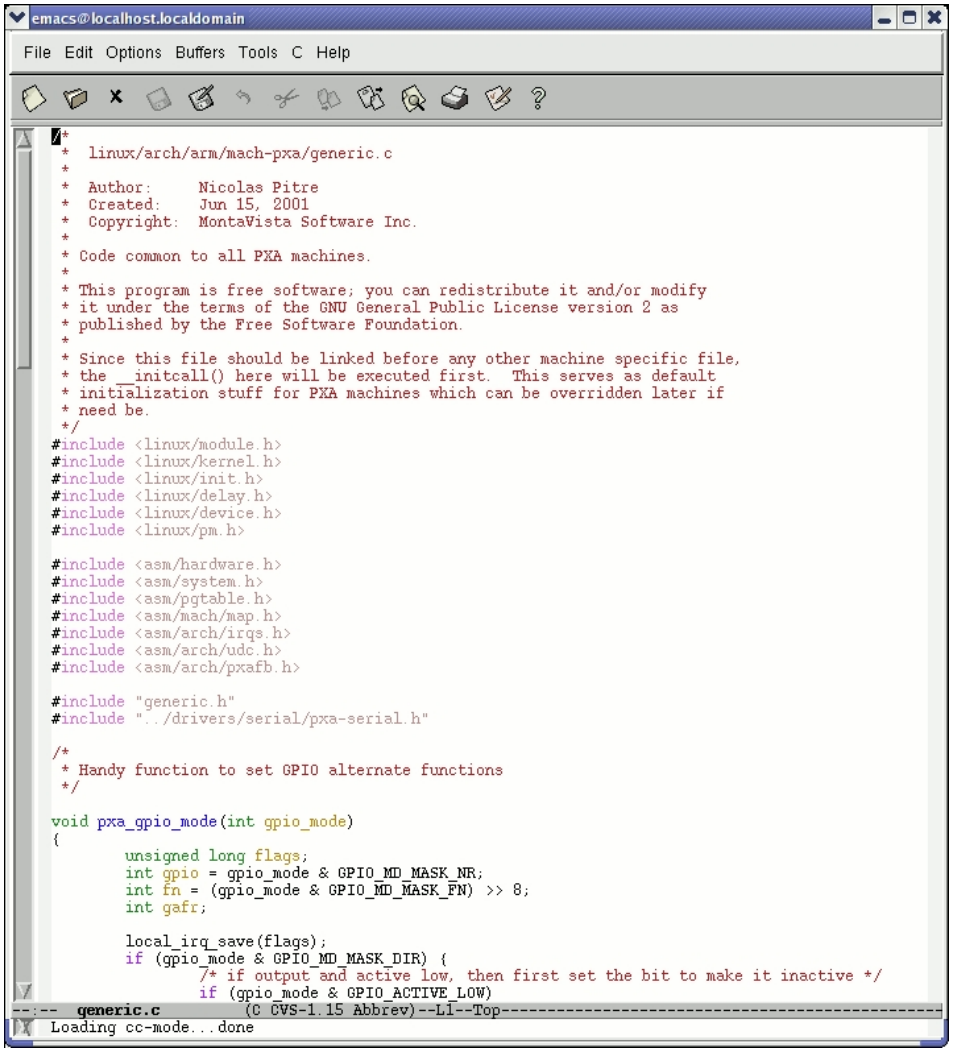
Éditeurs en mode texte uniquement

- **vi** (semi-graphique et très populaire)
- **nano** (très léger en mémoire et facile à apprendre)

Éditeurs spécialisés

- **Quanta+** (pour le web)

Emacs / XEmacs



- Emacs et Xemacs sont très semblables (choisissez selon votre goût)
- Fonctionnalités d'éditeur de texte extrêmement puissantes
- Editeur multi-modes programmable
- Parfait pour les utilisateurs avancés
- Bien plus d'un simple éditeur de texte (courrier, shell, navigateur)
- Raccourcis clavier non standards mais très complets et très puissants
- Nécessite un apprentissage assez long

Interpréteurs de commandes (1)

- Interpréteurs de commandes : outils pour exécuter des commandes tapées par un utilisateur dans un terminal
- Appelés “shells” (coquilles) parce qu’ils masquent sous leur surface les détails du système d’exploitation sous-jacent
- Les commandes sont tapées dans un terminal en mode texte, constitué par une fenêtre dans un environnement graphique, ou plus rarement par une console sur un écran en texte seul
- Les résultats sont aussi affichés sur le terminal. Une sortie graphique n’est pas obligatoire (pour la plupart des commandes la sortie est du texte)
- Les interpréteurs de commandes peuvent être programmables : ils fournissent toutes les ressources nécessaire pour l’écriture de programmes complexes (variables, conditions, boucles...)

Interpréteurs de commandes (2)

Une session konsole sous KDE 3.4

Interpréteurs les plus connus

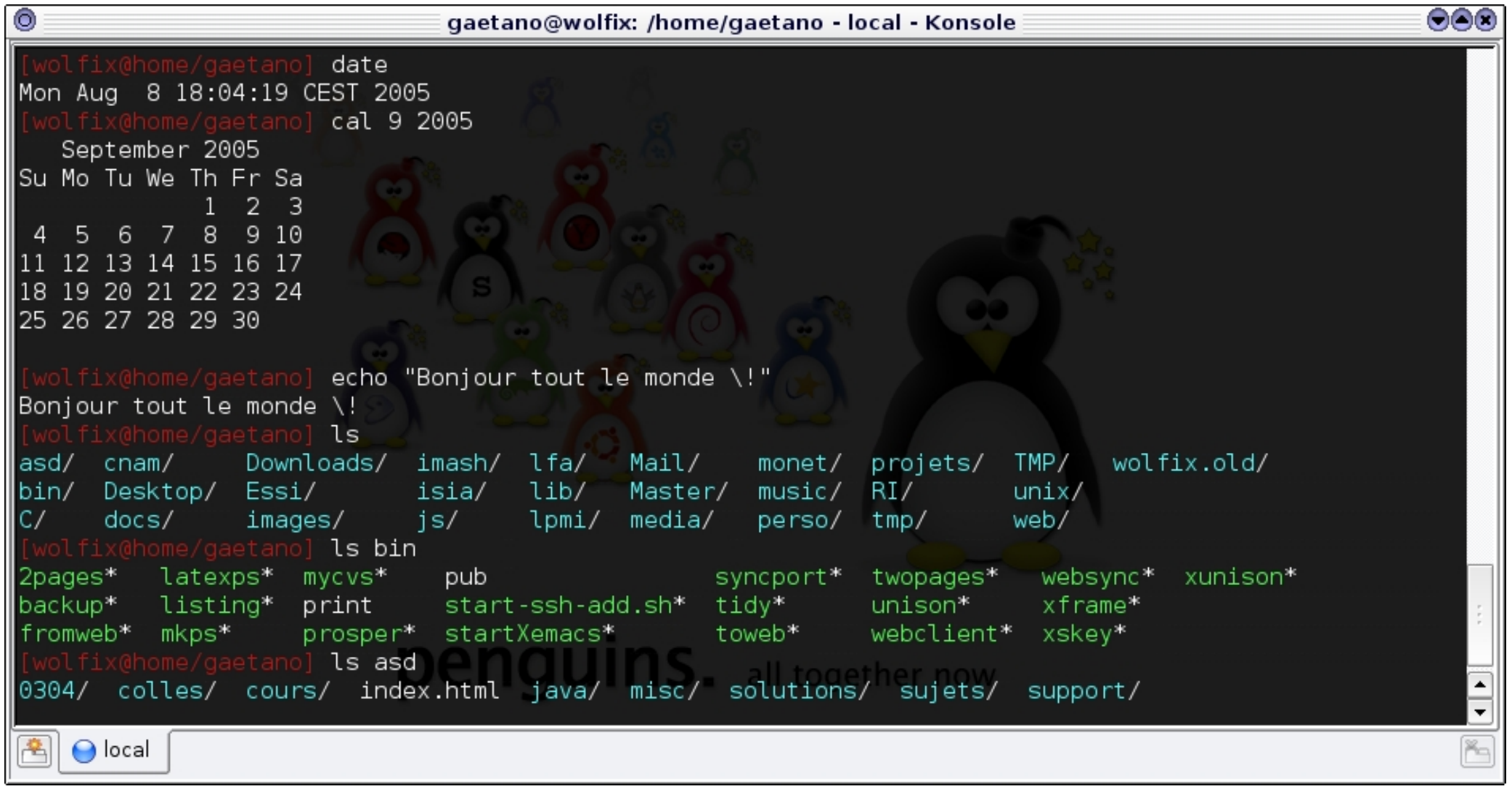
- **sh** : le Bourne shell (par Steve Bourne, obsolète)
  Le shell de base qu'on trouve tojours dans tous les systèmes Unix
- **csh** : le C shell (obsolète)
  Shell avec une syntaxe à la C, qui a connu son heure de gloire
- **tcsh** : le TC shell (toujours très populaire)
  Une implémentation compatible avec le C shell, avec des fonctionnalités avancées (complète les noms de commandes, rappel de commandes antérieures et bien d'autres)
- **bash** : le Bourne Again shell (par Steve Bourne, le plus populaire)
  Une version améliorée de sh avec de nombreuses fonctions nouvelles
- **ksh** : le Korn Shell (par David Korn, développé chez AT&T)
  Compatible avec le Bourne shell et inclut de nombreuses fonctions du C shell
- **zsh** : un Bourne shell étendu avec beaucoup d'améliorations. Il reprend la plupart des fonctions les plus pratiques de bash, ksh et tcsh

Syntaxe des commandes (1)

Une commande Unix peut avoir 0, un nombre fixe ou un nombre variable d'arguments, plus un certain nombre d'options. Une option commence en général par un tiret (`-`) et deux tirets à la suite (`--`) indiquent la fin des options sur une ligne de commande

- `[hal@home/bob] date`
  Une commande sans paramètre qui affiche la date et l'heure courante
- `[hal@home/bob] cal 2005`
  Une commande avec un paramètre (une année) qui affiche le calendrier complet de l'année en question
- `[hal@home/bob] cal 9 2005`
  La même commande avec deux paramètres, le mois et l'année
- `[hal@home/bob] echo`
  Sans argument cette commande affiche une ligne vide
- `[hal@home/bob] echo Bonjour tout le monde`
  Affiche les quatres arguments `Bonjour`, `tout`, `le` et `monde`, et va à la ligne
- `[hal@home/bob] echo -n Bonjour tout le monde`
  L'option -n empêche le passage à la ligne

Syntaxe des commandes (2)



L'interpréteur **bash** en action

Aide sur les commandes

La plupart des commandes de Linux proposent au moins un paramètre d'aide

- `-h`
  (`-` est surtout utilisé pour introduire des options en 1 caractère)
- `--help`
  (`--` est toujours utilisé pour introduire l'option "longue" correspondante, qui rend les scripts plus faciles à comprendre)

Les commandes affichent souvent un court résumé des options disponibles quand vous utilisez un argument invalide.

Pages de manuel

`man <mot_clé>`

Affiche une ou plusieurs pages de manuel pour `<mot_clé>`

- `man man`

La plupart des pages de manuel disponibles concernent des commandes Unix, mais aussi des fonctions, entêtes ou structures de données de bibliothèques C, ou même des fichiers de configuration du système

- `man stdio.h`
- `man fstab` (pour `/etc/fstab`)

Les pages de manuel sont recherchées dans les répertoires spécifiées par la variable d'environnement `MANPATH`

Pages info

- Sous GNU, les pages de manuel sont en voie de remplacement par les pages info. Certaines pages de manuel indiquent même de consulter plutôt les pages info

```
info <commande>
```

- Fonctionnalités d'`info`:
  - Documentation structurée en sections ("noeuds") et sous-sections ("sous-noeuds")
  - Possibilité de parcourir cette structure: sommet, suivant, précédent, haut
  - Pages info générées à partir des mêmes sources texinfo que la documentation en HTML

Recherche de ressources sur Internet (1)

Résolution de problèmes

- La plupart des archives des forums et des listes de discussion sont publiques, et indexées très régulièrement par Google
- Si vous recherchez la cause d'un message d'erreur, copiez-le mot à mot dans le formulaire de recherche, entouré par des guillemets. Il est très probable que quelqu'un d'autre aura déjà rencontré le même problème
- N'oubliez pas d'utiliser Google Groups: http://groups.google.com/ Ce site indexe plus de 20 années de messages de groupes de discussion

Recherche de ressources sur Internet (2)

Recherche de documentation

- Recherchez `<outil> OR <outil> page` pour trouver la page d'accueil de l'outil ou du projet et ensuite trouver les plus récentes ressources de documentation
- Recherchez `<outil> documentation or <outil> manual` (en anglais) dans votre moteur de recherche préféré

Recherche de documentation générique

- WikiPedia: http://fr.wikipedia.org
  De nombreuses et utiles définitions en informatique. Une vraie encyclopédie. Ouverte aux contributions de chacun

II. Principes et commandes de base

- Système de fichiers
- Commandes de base
- Liens symboliques et physiques
- Droits d'accès

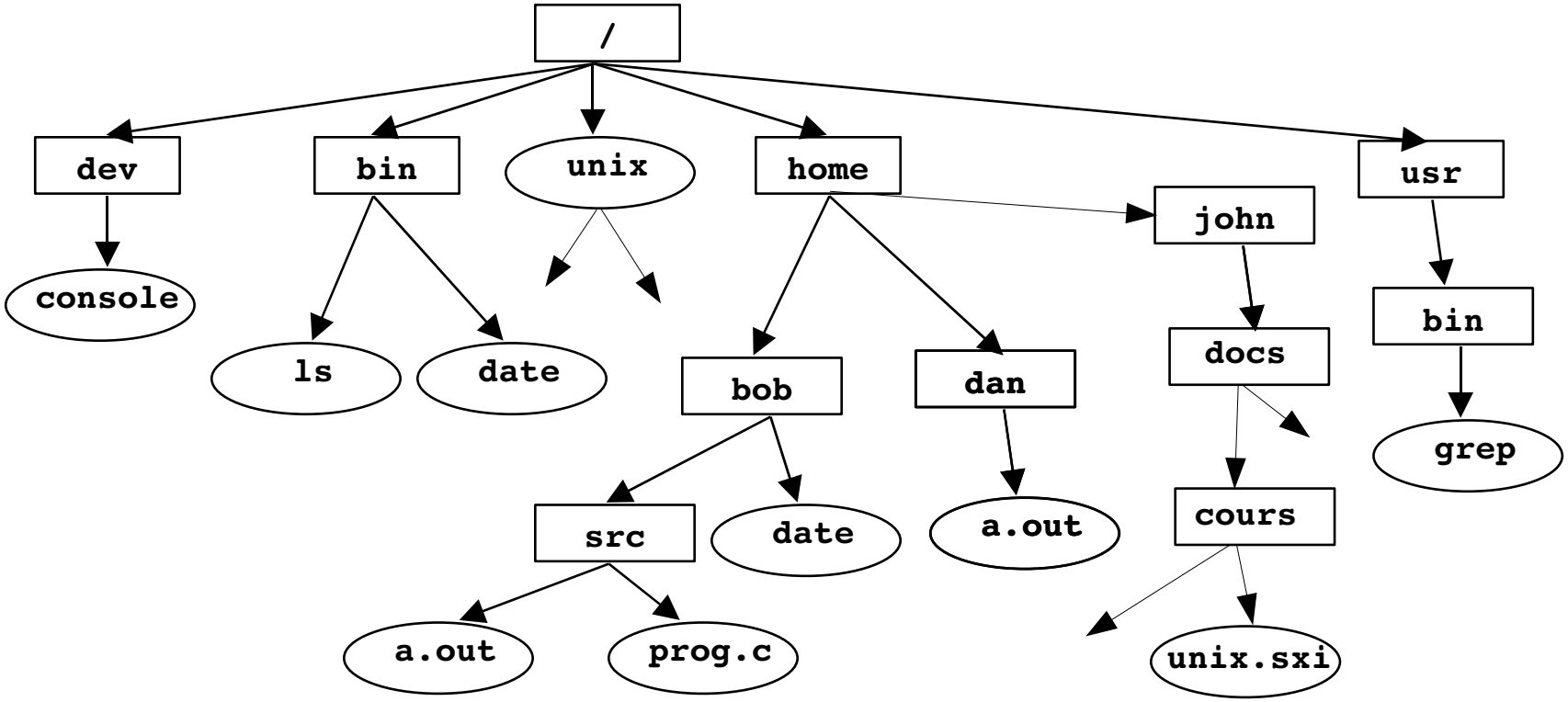
Fichiers et répertoires

Presque tout dans Unix / Linux est un fichier

- Même les répertoires sont des fichiers, contenant une liste de fichiers et de répertoires.
- Depuis le début d'Unix, aucune limitation majeure quant à la longueur d'un nom de fichier. Tout caractère (les espaces en particulier) peut être utilisé dans le nom, et les extensions sont facultatives. Les différences de majuscules ou de minuscules constituent des fichiers distincts.
- Un *chemin* («path») est une séquence de répertoires imbriqués, avec un fichier ou un répertoire à la fin, séparés par le caractère `/`
  - Chemin relatif: `docs/cours/unix.html`
    Relatif au répertoire courant
  - Chemin absolu: `/home/john/docs/cours/unix.html`
    Chemin depuis le répertoire racine du système (`/`)

Structure de fichiers dans Linux (1)

Structure arborescente



Structure de fichiers dans Linux (2)

Rien d'imposé par le système. Peut varier d'un système à l'autre, même entre deux installations de Linux!

`/`	Répertoire racine
`/bin/`	Commandes de base du système
`/boot/`	Images, initrd et fichiers de configuration du noyau
`/dev/`	Fichiers représentant des périphériques `/dev/hda`: premier disque dur IDE
`/etc/`	Fichiers de configuration du système
`/home/`	Répertoires utilisateurs
`/lib/`	Bibliothèques de base du système (partagées)

Structure de fichiers dans Linux (3)

`/lost+found`	Fichiers détériorés que le système a essayé de récupérer
`/mnt/`	Systèmes de fichiers montés `/mnt/usbdisk/`, `/mnt/windows/` ...
`/opt/`	Outils spécifiques installés par l'administrateur. Souvent remplacé par `/usr/local/`
`/proc/`	Accès aux informations du système `/proc/cpuinfo`, `/proc/version` ...
`/root/`	répertoire utilisateur de l'administrateur
`/sbin/`	Commandes réservées à l'administrateur
`/sys/`	Contrôle du système et des périphériques (fréquence du processeur, gestion de l'alimentation des périphériques, etc.)

Structure de fichiers dans Linux (4)

`/tmp/`	Fichiers temporaires
`/usr/`	Programmes utilisateurs ordinaires, non essentiels au système. `/usr/bin/`, `/usr/lib/`, `/usr/sbin`...
`/usr/local/`	Outils spécifiques installés par l'administrateur. (souvent préféré à `/opt/`)
`/var/`	Données utilisées par le système ou ses serveurs `/var/log/`, `/var/spool/mail` (courrier entrant), `/var/spool/lpd` (travaux d'impression)...

Répertoires spéciaux (1)

Le répertoire relatif `./`

- Le répertoire courant. Utile pour les commandes qui ont un répertoire comme argument. Également utile parfois pour lancer des commandes dans le répertoire courant (voir plus tard)
- Ainsi `./lisezmoi.txt` et `lisezmoi.txt` sont équivalents

Le répertoire relatif `../`

- Le répertoire parent (englobant). Fait partie toujours partie du répertoire `.`(voir `ls -a`). Unique référence au répertoire parent.
- Utilisation la plus courante:
  `cd ..`

Répertoires spéciaux (2)

Le répertoire absolu `~/`

- Pas vraiment un répertoire spécial. Les interpréteurs de commande le remplacent juste par le répertoire utilisateur de l'utilisateur courant
- Ne peut pas être utilisé dans la plupart des programmes, car il n'est pas un vrai répertoire (voir la variable `HOME`)

Le répertoire absolu `~bob/`

- De façon analogue, remplacé par les shells par le répertoire utilisateur de l'utilisateur `bob`

Chemins absolus et relatifs

Un fichier peut être désigné de façon absolu ou relative

- Un chemin absolu identifie de manière unique un fichier ou un répertoire dans l'arborescence :

```
/home/bob/src/a.out
/usr/bin/grep
```

- Un chemin relatif identifie un fichier ou un répertoire en fonction du répertoire courant (on est sous `~bob`) :

```
src/a.out
../dan/a.out
```

Les commandes basename et dirname

- `basename fichier [suffix]`
  Retourne le nom de base d'un fichier :
  - `basename xxx/yyy/zzz`
    Retourne `zzz`
  - `basename xxx/yyy/zzz.c`
    Retourne `zzz.c`
  - `basename xxx/yyy/zzz.c '.c'`
    Retourne `zzz`
- `dirname fichier`
  Retourne le *préfixe* répertoire d'un fichier :
  - `dirname xxx/yyy/zzz`
    Retourne `xxx/yyy`

La chaîne de caractères `fichier` n'a pas besoin d'être le chemin d'un vrai fichier !

La commande ls

Affiche la liste des fichiers dans le répertoire courant, en ordre alphanumérique, sauf ceux qui commencent par le caractère "`.`"

- `ls -a` («all»: tous)
  Affiche tous les fichiers (y compris les fichiers `.*`)
- `ls -l` (long)
  Affichage en format long (type, date, taille, propriétaire, permissions)
- `ls -t` (temps)
  Affiche les fichiers les plus récents en premier
- `ls -S` ("size": taille)
  Affiche les fichiers les gros en premier
- `ls -r` («reverse»: inversé)
  Affiche en ordre inverse
- `ls -ltr` (les options peuvent être combinées)
  Format long, les fichiers les plus récents à la fin

La commande `ls` possède une cinquantaine d'options !

Les commandes cd et pwd

- `cd rep` (*change directory*)
  Fait de `rep` le nouveau répertoire courant
- `cd`
  Sans argument, fait du répertoire utilisateur le nouveau répertoire courant
- `cd -` (le caractère tiret : '-')
  Fait du répertoire précédent le répertoire courant (le dernier répertoire depuis lequel on a fait cd) le nouveau répertoire courant
- `pwd` (*print working directory*)
  Affiche le chemin absolu du répertoire courant

Les commandes mv et cp

- `mv ancien_nom nouveau_nom` (*move*)
  Change le nom du fichier ou du répertoire donné
- `mv -i` (*interactive*)
  Si le fichier existe déjà, demander confirmation à l'utilisateur
- `cp fichier_orig fichier_dest`
  Crée une copie d'un fichier d'origine
- `cp fich1 fich2 fich3 ... rep`
  Copie tous les fichiers vers le répertoire de destination (dernier argument)
- `cp -i` (*interactive*)
  Si le fichier de destination existe déjà, demander confirmation à l'utilisateur
- `cp -r rep_orig rep_dest` (*recursive*)
  Copie du répertoire tout entier

Les commandes rm et rmdir

- `rm fich1 fich2 fich3...` (*remove*)
  Supprime les fichiers donnés
- `rm -i fich1 fich2 fich3...` (*interactive*)
  Demande toujours à l'utilisateur de confirmer les suppressions
- `rm -r rep1 rep2 rep3...` (*recursive*)
  Suppression des répertoires donnés et de tout leur contenu
- `rm -f fich1 fich2 fich3...` (*force*)
  Suppression des fichiers donnés sans message ni question
- `rmdir rep1 rep2 rep3...` (*remove directory*)
  Suppression des répertoires donnés seulement s'ils sont vides

La commande mkdir

- `mkdir rep` (*make directory*)
  Fabrique `rep`, un nouveau répertoire, dans le courant. Si `rep` existe dèjà, provoque une erreur
- `mkdir rep1 rep2 ... repN`
  Fabrique séquentiellement plusieurs répertoires
- `mkdir rep1/rep2`
  Il faut que le répertoire `rep1` existe déjà, sinon provoque une erreur
- `mkdir -p rep1/rep2/.../repN` (*parent*)
  Fabrique les répertoires `rep1 rep2 ... repN-1` si besoin, et enfin le répertoire `repN`

Substitutions sur les noms de fichiers

- `ls *txt`
  L'interpréteur remplace d'abord `*txt` par tous les noms de fichiers et de répertoires finissant par `txt` (y compris `.txt`), sauf ceux commençant par `.`, et enfin exécute la ligne de commande `ls`
- `ls -d .*`
  Affiche tous les fichiers et les répertoires commençant par `.`
  `-d` indique à `ls` de ne pas afficher le contenu des dossiers `.*`
- `ls ?.log`
  Affiche tous les fichiers dont le nom commence par 1 caractère et finit par `.log`
- `rm -rf *`
  Efface TOUS les fichiers et sous-répertoires du répertoire courant.
  A utiliser avec précaution !

Caractères génériques (wildcards)

Plus généralement :

- `*` dénote 0, 1 ou plusieurs caractères

```
*.c        doc*unix.html
```

- `?` dénote un (et exactement un) caractère quelconque

```
?nix       *.s??
```

- `[...]` dénote un caractère appartenant à un ensemble

```
[Uu]nix    *.sx[ci]
```

- `[^...]` dénote un caractère n'appartenant pas à un ensemble

```
*.[^co]
```

Afficher le contenu de fichiers

Plusieurs façons d'afficher le contenu de fichiers

- `cat fich1 fich2 fich3 ...` (concaténer)
  Met bout à bout et affiche le contenu des fichiers donnés
- `more fich1 fich2 ...` (plus de détails)
  A chaque page, demande à l'utilisateur d'appuyer sur une touche pour continuer. Peut aussi aller directement à la première apparition d'un mot clé (commande "`/`")
- `less fich1 fich2 fich3 ...` (moins)
  Un `more` plus puissant !
  Ne lit pas le fichier entier avant de commencer à afficher
  Permet de remonter en arrière dans le fichier (commande "`?`")

Les commandes head, tail et cat

- `head [-n] fichier` (*tête*)
  Affiche les n premières lignes (ou 10 par défaut) du fichier donné.
  N'a pas besoin d'ouvrir le fichier en entier pour le faire
- `tail [-n] fichier` (*queue*)
  Affiche les n dernières lignes (ou 10 par défaut) du fichier donné
  Ne charge pas tout le fichier en mémoire. Très utile pour les gros fichiers.
- `tail -f fichier` (*follow*)
  Affiche les 10 dernières lignes du fichier donné et continue à afficher les nouvelles lignes au fur et à mesure qu'elles sont rajoutées en fin de fichier. Très pratique pour suivre les rajouts à un fichier de journal ("log")
- `cat [-n] fichier`
  Affiche le contenu du fichier donné. Avec l'option `-n` numérote chaque ligne affichée de 1 à n (le nombre de lignes)

La commande wc

- `wc fichier` (*word count*)
  Affiche le nombre de lignes, de mots et de caractères du fichier `fichier`
- `wc -c fichier`
  Affiche le nombre d'octets contenus dans le fichier `fichier`
- `wc -m fichier`
  Affiche le nombre de caractères contenus dans le fichier `fichier`
- `wc -l fichier`
  Affiche le nombre de lignes contenues dans le fichier `fichier`
- `wc -w fichier`
  Affiche le nombre de mots contenus dans le fichier `fichier`

La commande sort

- `sort fichier`
  Trie les lignes du fichier selon l'ordre des caractères et les affiche.
- `sort -r fichier` (*reverse*)
  Idem, mais en ordre inverse
- `sort -ru fichier` (*unique*)
  Idem, mais ne produit qu'une seule fois les lignes identiques.
- Plus de possibilités seront abordées plus tard

La commande grep

- `grep motif fichiers`
  Parcourt les fichiers donnés et affiche les lignes qui correspondent au motif spécifié.
- `grep erreur *.log`
  Affiche toutes les lignes contenant `erreur` dans les fichiers `*.log`
- `grep -i erreur *.log`
  Idem, mais indifférent aux majuscules / minuscules
- `grep -ri erreur .`
  Idem, mais récursivement dans `.` et ses sous-répertoires
- `grep -v info *.log`
  Affiche toutes les lignes des fichiers, sauf celles qui contiennent `info`

La commande find

Plus facile à expliquer par quelques exemples

- `find . -name “*.pdf” [-print]`
  Recherche tous les fichiers `*.pdf` dans le répertoire courant (`.`) et ses sous-répertoires. Vous devez utiliser les guillemets pour empêcher le shell de substituer le caractère `*`
- `find docs -name "*.pdf" -exec xpdf {} ';'`
  Recherche tous les fichiers `*.pdf` dans le répertoire `docs` et les affiche l’un après l’autre
- De nombreuses possibilités existent ! Cependant, les 2 exemples ci-dessus couvrent la plupart des besoins

La commande locate

Outil de recherche à base d'expressions régulières, essentiellement pour trouver l'emplacement de fichiers ou de répertoires

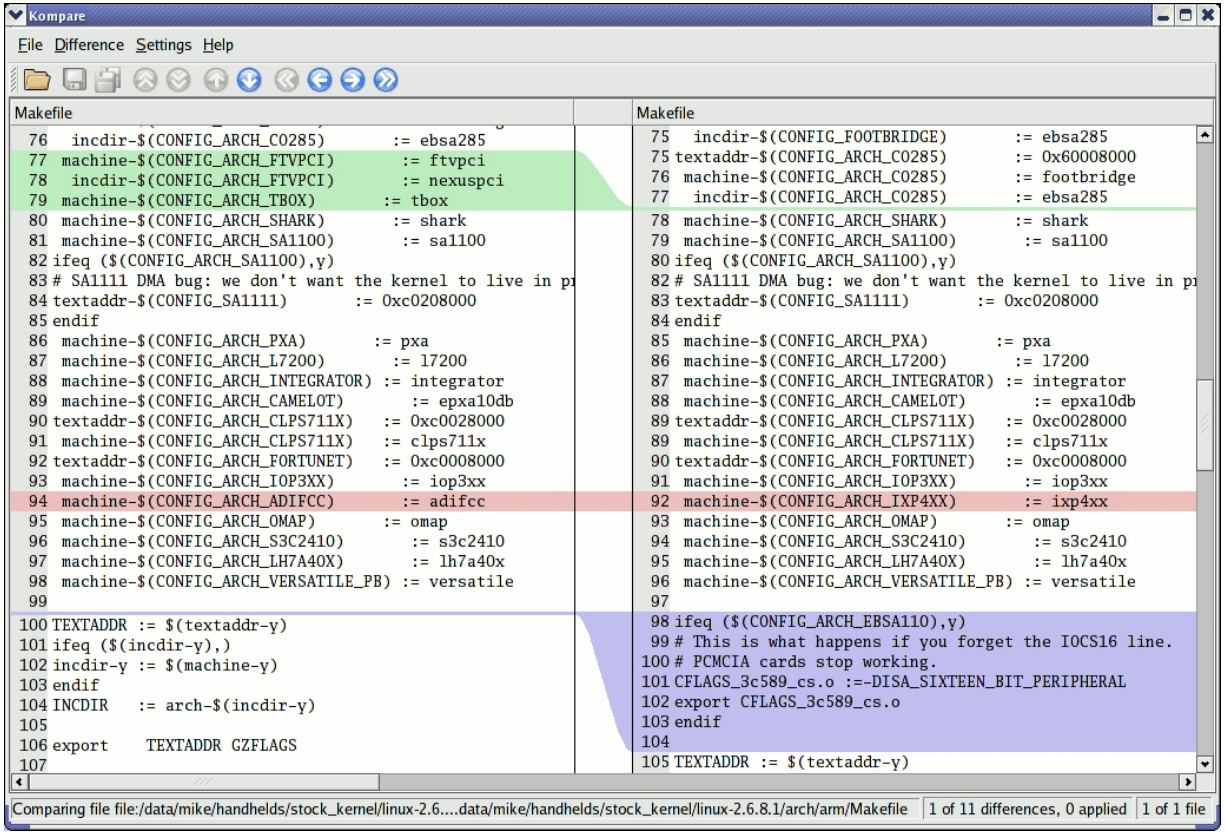
- `locate clé`
  Affiche tous les fichiers sur votre système contenant `clé` dans leur nom
- `locate "*.pdf"`
  Affiche tous les fichiers `*.pdf` existant sur votre système.
- `locate "/home/frigo/*mousse*"`
  Affiche tous les fichiers `*mousse*` dans le répertoire indiqué (chemin absolu)
- `locate` est très rapide grâce à l'indexation de tous les fichiers dans une base de données dédiée, qui est mise à jour régulièrement
- `locate` est beaucoup moins précis que `find` qui est beaucoup plus adapté aux recherches associées à des actions sur les fichiers trouvés

Comparaison de fichiers et répertoires

- `cmp fichier1 fichier2`
  Compare deux fichiers quelconques.
- `diff fichier1 fichier2`
  Affiche les différences entre deux fichiers, ou rien si les fichiers sont identiques.
- `diff -r rep1/ rep2/`
  Affiche les différences entre fichiers de même nom dans les deux répertoires.
- Pour examiner en détail les différences, mieux vaut utiliser des outils graphiques

kompare (KDE)

Un autre outil pratique pour comparer des fichiers et fusionner leurs différences



Mesure de la taille de fichiers

- `du -h fichier` (disk usage)
  `-h`: affiche la taille du fichier donné, sous forme lisible par un humain: K (kilo-octets), M (mega-octets) or G (giga-octets). Sinon `du` rend le nombre brut de blocs occupés par le fichier sur le disque (difficile à lire).

  Remarque: l'option `-h` n'existe que dans GNU `du` (pas disponible sur le `du` de Sun Solaris, par exemple)

- `du -sh rep`
  `-s`: rend la somme des tailles de tous les fichiers dans le répertoire donné

Mesure de l'espace disque

- `df -h rep`
  Affiche des informations sur l'espace disque utilisé et disponible dans le système de fichiers qui contient le répertoire donné.
  De même, l'option `-h` n'existe que dans GNU `df`.

- Exemple :

```
> df -h .
Filesystem            Size  Used Avail Use% Mounted on
/dev/hda5             9.2G  7.1G  1.8G  81% /
```

- `df -h`
  Affiche les informations d'espace disque pour tous les systèmes de fichiers disponibles sur le système. Quand des erreurs surviennent, utile pour vérifier si des systèmes de fichiers pleins.

Compression

Très utile pour compacter de gros fichiers et économiser de la place

- `[un]compress fichier`
  Utilitaire de compression traditionnel d'Unix. Crée des fichiers `.Z`.
  Seulement gardé pour raisons de compatibilité. Performance moyenne.
- `g[un]zip fichier`
  Utilitaire de compression GNU zip. Crée des fichiers `.gz`.
  Assez bonne performance (un peu meilleur que Zip)
- `b[un]zip2 <fichier>`
  Le plus récent et le plus performant des utilitaires de compression. Crée des fichiers `.bz2`. La plupart du temps 20-25% meilleur que `gzip`.
  Utilisez celui-ci ! Maintenant disponible sur tous les systèmes Unix.

Archivage (1)

Utile pour sauvegarder ou publier un ensemble de fichiers

- `tar`: à l'origine "tape archive" ("archive sur bande")
- Création d'une archive:
  `tar cvf archive fichiers-ou-répertoires`
  `c` (créer) : pour fabriquer l'archive
  `v` (verbeux) : utile pour suivre la progression de l'archivage
  `f` (fichier) : archive créée dans un fichier (et non sur une bande)
- Exemple :
  ```
  tar cvf /backup/home.tar /home
  bzip2 /backup/home.tar
  ```

Archivage (2)

- Afficher le contenu d'une archive ou vérifier son intégrité:
  `tar tvf archive`
- Extraire tous les fichiers d'une archive:
  `tar xvf archive`
- Extraire seulement quelques fichiers d'une archive:
  `tar xvf archive fichiers-ou-répertoires`
  Les fichiers ou répertoires sont donnés avec un chemin relatif au répertoire racine de l'archive.

Options supplémentaires dans GNU tar

GNU `tar` sous GNU / Linux

Permet de compresser et décompresser des archives au vol. Utile pour éviter de créer d'énormes fichiers intermédiaires. Plus facile à faire qu'en combinant `tar` et `gip/bzip2`

- `j`: [dé]compresse au vol avec `bzip2`
- `z`: [dé]compresse au vol avec `gzip`
- Exemples
  - `tar jcvf projet.tar.bz2 projet/`
  - `tar ztf projet.tar.gz`

Impression sous Unix

- Multi-utilisateurs, multi-travaux, multi-clients, multi-imprimantes :
  Sous Unix / Linux, les commandes d'impression n'impriment pas vraiment. Elles envoient des tâches à des queues d'impression, soit sur la machine locale, soit sur des serveurs d'impression ou sur des imprimantes réseau
- Système indépendant de toute imprimante :
  Les serveurs d'impression n'acceptent que des travaux en PostScript ou en texte. Les pilotes d'imprimante sur le serveur se chargent de la conversion vers le format propre à chaque imprimante

- Système robuste :
  Redémarrez un système, il continuera à imprimer les travaux en attente
- L'impression est soit pillotée par l'application (comme OpenOffice) soit lancée par l'utilisateur (impression de fichier PostScript ou texte)

Impression de fichiers PostScript

- Variable d'environnement utile: `PRINTER`
  Définit l'imprimante par défaut sur le système. Exemple:
  `export PRINTER=niv2a`
- `lpr [-Pimp] fichiers`
  Envoie les fichiers à la file d'attente de `imp`, l'imprimante spécifiée. Les fichiers doivent être en format texte ou PostScript, sinon, vous n'imprimerez que des déchets
- `a2ps [-Pimp] fichiers`
  Convertit de nombreux formats vers PostScript et l'envoie le résultat vers la queue spécifiée. Fonctionnalités utiles: plusieurs pages / feuille, numérotation des pages, cadre d'informations

Contrôle de travaux d'impression

- `lpq [-Pqueue]`
  Affiche tous les travaux d'impression de la queue par défaut ou de la queue donnée

```
lp is not ready
Rank    Owner   Job     File(s)                         Total Size
1st     asloane 84      troyens_windows_nsa.ps          60416 bytes
2nd     amoore  85      gw_bush_erreurs_irak.ps         65024000 bytes
```

- `cancel numéro_tâche [queue]`
  Retire la tâche spécifiée de la queue d'impression
- Le contrôle des travaux d'impression est également possible depuis le Window Manager : `KjobViewer` sous KDE

Utilisation de fichiers PostScript et PDF

Manipulation d'un fichier PostScript

- Visualisation avec `gv`, `Kghostview`
- Conversion possible en PDF avec `ps2pdf` :
  `ps2pdf document.ps [document.pdf]`
- Impression directe avec `lpr`

Manipulation d'un fichier PDF

- Visualisation avec `acroread`, `xpdf`, `Kpdf`
- Conversion possible en PostScript avec `pdf2ps`:
  `pdf2ps document.pdf [document.ps]`
- Impression après conversion en PostScript

Liens symboliques

Un lien symbolique est un fichier spécial qui est juste une référence au nom d'un autre (fichier ou répertoire)

- Utile pour simplifier et réduire l'utilisation du disque quand deux fichiers ont le même contenu.
- Exemple:
  `/usr/local/bin/java -> /usr/jdk1.5.0_03/bin/java`
- Comment distinguer les liens symboliques:
  - `ls -l` affiche `->` et le fichier référencé par le lien
  - GNU `ls` affiche les liens avec une couleur différente

Création de liens symboliques

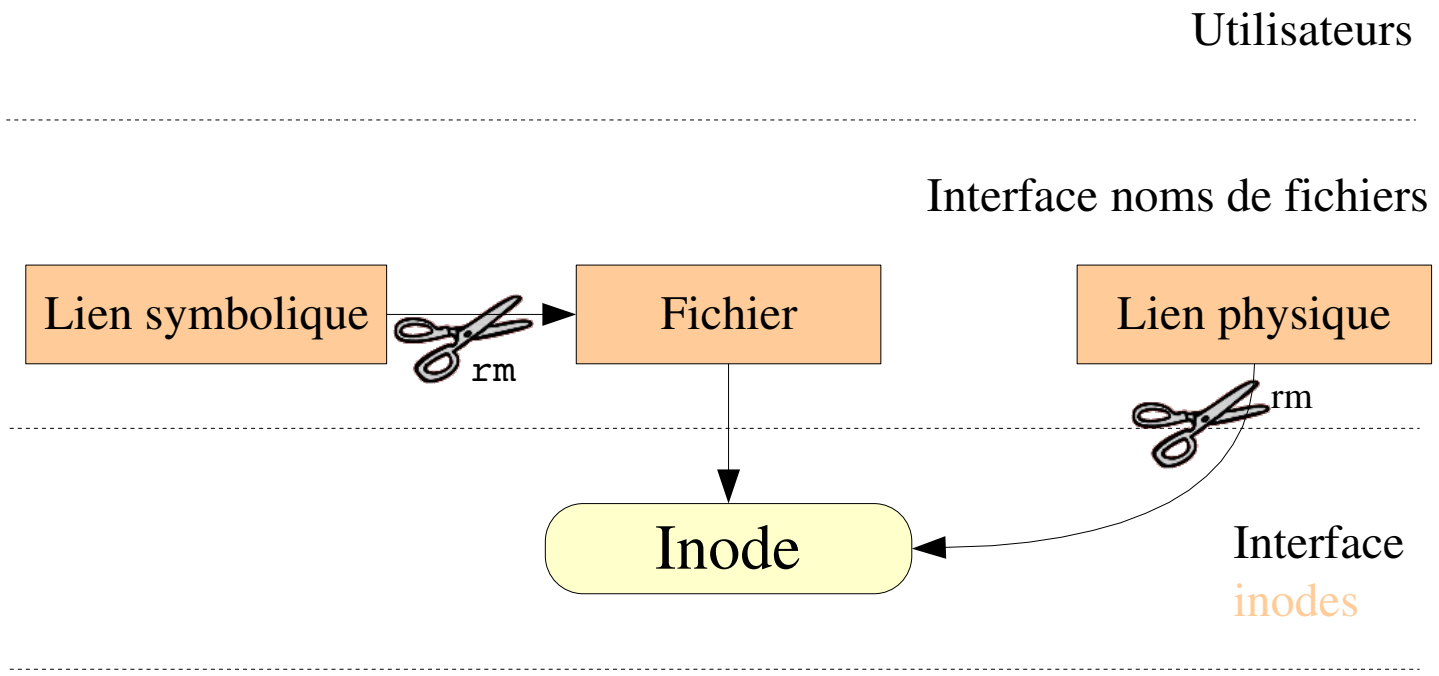
- Pour créer un lien symbolique (même ordre que dans `cp`) :
  `ln -s nom_fichier nom_lien`
- Pour créer un lien vers un fichier dans un autre répertoire, avec le même nom :
  `ln -s ../LISEZ_MOI.txt`
- Pour créer plusieurs liens d'un coup dans un dossier donné :
  `ln -s fich1 fich2 fich3 ... rep`
- Pour supprimer un lien :
  `rm nom_lien`
  Bien-sûr, cela ne supprime pas le fichier référencé par le lien

Liens physiques

- Par défaut, `ln` crée des *liens physiques*
- Un *lien physique* vers un fichier est un fichier ordinaire, avec exactement le même contenu physique
- Bien qu'ils économisent toujours de la place, les liens physiques sont indiscernables des fichiers d'origine.
- Si vous supprimez le fichier d'origine, cela n'affecte pas le contenu du lien physique.
- Le contenu est supprimé quand il n'y a plus aucun fichier (lien physique) qui y fait référence.

Noms de fichiers et inodes

Pour mieux comprendre les liens symboliques et physiques



Droits d'accès aux fichiers

Utiliser `ls -l` pour consulter les droits d'accès

3 types de droits d'accès :

- Accès en lecture (`r`)
- Accès en écriture (`w`)
- Droits d'exécution (`x`)

3 types de niveaux d'accès:

- Utilisateur (`u`)
  pour le propriétaire du fichier
- Groupe (`g`)
  tout fichier a un attribut "groupe", qui correspond à une liste d'utilisateurs
- Autres (`o`)
  pour tous les autres (propriétaire et groupe exclus)

Contraintes de droits d'accès

- `x` sans `r` est autorisé mais sans valeur.
  Vous devez pouvoir lire un fichier pour l'exécuter
- Les répertoires requièrent à la fois les droits `r` et `x` : `x` pour entrer, `r` pour accéder au contenu
- Vous ne pouvez pas renommer, supprimer ou copier des fichiers dans un répertoire si vous n'avez pas accès en écriture à ce répertoire
- Si vous avez accès en écriture à un répertoire, vous POUVEZ supprimer un fichier même si vous ne disposez pas de droits d'écriture pour ce fichier (souvenez-vous qu'un répertoire est juste un fichier décrivant une liste de fichiers). Cela permet même de modifier un fichier (le supprimer et le recréer) même protégé en écriture

Exemples de droits d'accès

- `-rw-r--r--`
  Lisible et modifiable pour le propriétaire, seulement lisible pour les autres.
- `-rw-r-----`
  Lisible et modifiable pour le propriétaire, seulement lisible pour les utilisateurs appartenant au groupe du fichier.
- `drwx------`
  Répertoire seulement accessible par son propriétaire
- `-------r-x`
  Fichier exécutable seulement par les autres, mais ni par votre groupe ni par vous-même. Droits d'accès typique d'un piège !

chmod: modifier les permissions

- `chmod permissions fichiers`
  2 formats possibles pour `permissions` : format en base 8 ou format symbolique
- Format en base 8 : `chmod abc fichiers`
  avec `a`, `b` et `c` valant `r*4+w*2+x` (`r`, `w`, `x` égaux à 0 ou 1)
  Exemple : `chmod 644 fichier` (`rw` pour `u`, `r` pour `g` et `o`)
- Format symbolique : plus facile à comprendre sur des exemples

`chmod go+r`	ajouter le droit en lecture au groupe et aux autres
`chmod u-w`	supprimer le droit d'écriture pour le propriétaire
`chmod a-x`	(*all*) enlever les droits d'exécution à tous les utilisateurs

Autres options de chmod (1)

```
chmod -R a+rX linux/
```

Rend `linux` et tout ce qu'il contient accessible à tout le monde!

- `R` : applique les changements récursivement
- `X` : `x`, mais seulement pour répertoires et fichiers déjà exécutables. Très pratique pour ouvrir récursivement l'accès à des répertoires, sans donner le droit d'exécution à tous les fichiers

Autres options de chmod (2)

```
chmod a+t /tmp
```

- `t` : (“sticky”: collant). Permission spéciale pour les répertoires, autorisant uniquement l’effacement d’un fichier par son propriétaire ou par celui du répertoire
- Utile pour les répertoires accessibles en écriture par plusieurs utilisateurs, comme `/tmp`

III. Redirections et processus

- Entrées/sorties et redirections
- Les tuyaux
- Contrôle de tâche et de processus

Descripteur de fichier

- Les entrées/sorties des commandes sont repérées par des descripteurs de fichier ("*file descriptor*")
- Un descripteur de fichier est un *entier* associé à un fichier spécial
- On peut modifier (en les *redirigeant*) les descripteurs de fichier associés aux entrées/sorties de n'importe quelle commande
- Les entrées/sorties *standards* ne sont que des descripteurs de fichier prédéfinis associés par défaut aux entrées/sorties des commandes
- Pour chaque commande, on distingue :
  - l'entrée standard : c'est le clavier
  - la sortie standard : c'est l'écran
  - la sortie d'erreur standard : c'est l'écran

Entrée standard (1)

De nombreuses commandes, invoquées sans arguments en entrée, lisent leurs données sur l'*entrée standard*

- Le *file descriptor* décrivant l'entrée standard est `0`
- L'entrée standard est aussi le fichier `/dev/stdin`
- Par défaut, l'entrée standard est le clavier
- L'entrée standard peut être écrite (redirigée) dans un fichier en utilisant le symbole `<`

Entrée standard (2)

- ```
  [hal@/home/bob] sort
  windows
  linux
  <Ctrl D>
  linux
  windows
  ```

`sort` prend l'entrée standard comme entrée, c'est à dire tout ce qui est tapé sur le clavier ! (jusqu'au premier `<Ctrl D>`)

- `sort < participants.txt`
  L'entrée standard de `sort` est prise dans le fichier indiqué. Note : ici, on écrirait plutôt `sort participants.txt`
- `mail -s "Hello" dan < my_mail.txt`
  L'entrée standard de la commande `mail` est redirigée vers le fichier `my_mail.txt` qui contient le corps du message


  ```

Sortie standard (1)

Toutes les commandes qui produisent du texte sur le terminal le font en écrivant sur leur *sortie standard*

- Le *file descriptor* décrivant la sortie standard est `1`
- La sortie standard est aussi le fichier `/dev/stdout`
- Par défaut, la sortie standard est l'écran
- La sortie standard peut être écrite (redirigée) dans un fichier en utilisant le symbole `1>` ou simplement `>`
- La sortie standard peut être rajoutée à la fin d'un fichier existant par le symbole `1>>` ou simplement `>>`

Sortie standard (2)

- `ls ~ > tous_mes_fichiers.txt`

  Copie le résultat de la commande `ls` dans le fichier `tous_mes_fichiers.txt`. Le fichier est écrasé s'il existait avant ou bien créé s'il n'existait pas.

- `ls ~/bin >> tous_mes_fichiers.txt`

  Ajoute la sortie de la nouvelle commande `ls` à la suite du fichier `tous_mes_fichiers.txt`

- `cat .zlogin .zshrc .zlogout > zsh_all.txt`

  Concatène le contenu des trois fichiers `.zlogin`, `.zshrc` et `.zlogout` dans le fichier `zsh_all.txt`

- `find / -name '*.gif' -print > images.txt`

  Copie la sortie de la commande `find` (tous les fichiers du système d'extension `gif`) dans le fichier `images.txt`

L’erreur standard (1)

Les messages d’erreur d'une commande sont envoyés vers l’*erreur standard* (et non pas la sortie standard)

- Le *file descriptor* décrivant l'erreur standard est `2`
- L'erreur standard est aussi le fichier `/dev/stderr`
- Par défaut, l'erreur standard est l'écran
- L'erreur standard peut être écrite (redirigée) dans un fichier en utilisant le symbole 2>
- L'erreur standard peut être rajoutée à la fin d’un fichier existant par le symbole 2>>

L'erreur standard (2)

- Redirection de l'erreur standard vers un fichier :
  ```
  gcc myfile.c 2> erreur.log
  ```
- Redirection de la sortie et de l'erreur standard vers des fichiers distincts :
  ```
  gcc myfile.c > comp.log 2> erreur.log
  ```
- Redirection de la sortie et de l'erreur standard vers le même fichier :
  ```
  gcc myfile.c > comp.log 2>&1
  ```
  Syntaxe alternative :
  ```
  gcc myfile.c &> comp.log
  ```
- On peut ignorer les erreurs en redirigeant l'erreur standard vers le pseudo-fichier `/dev/null` :
  ```
  find / -name '*.gif' > images.txt 2> /dev/null
  ```

Les tuyaux (1)

Pour combiner et enchaîner les commandes entre elles

- Les tuyaux Unix servent à rediriger la sortie d'une commande vers l'entrée d'une autre commande :
  - `find . -name '*.gif' -print | more`
  - `ls | wc -l`
  - `cat *.log | grep -i erreur | sort`
  - `cat /home/*/devoirs.txt | grep note | more`
- On peut combiner tuyaux et redirections :
  - `cat devoirs/*/note.txt | grep note > notes.txt`
  - `find / -name '*.cpp' 2> /dev/null | more`

Les tuyaux (2)

Les tuyaux s'utilisent principalement avec des filtres destinés à modifier la sortie d'une commande par application successive d'autres commandes

- Recherche de ligne contenant un motif parmi les lignes du flux d'entrée avec les commandes `grep`, `egrep` et `fgrep`
- Sélection de colonnes ou de champs sur les lignes du flux d'entrée avec la commande `cut`
- Tri des lignes du flux d'entrée avec la commande `sort`
- Remplacement de caractères sur les lignes du flux d'entrée avec la commande `tr`
- Duplication du canal de sortie standard avec la commande `tee`

Les commandes tr et cut

- `tr chaine1 chaine2`
  Remplace les caractères de `chaine1` par les caractères de `chaine2`
  - `echo Bonjour Tout le Monde | tr 'Bol' 'b0L'`
    Retourne `b0nj0ur t0ut Le m0nde`
  - `echo Bonjour Tout le Monde | tr 'A-Z' 'a-z'`
    Retourne `bonjour tout le monde`
  - `echo 'a  demain !' | tr -s ' '`
    Retourne `a demain !`
- `cut [option]... [fichier]...`
  Retourne des *parties* des lignes du fichier `fichier` (ou de l'entrée standard)
  - `echo '/usr/bin/X11:/usr/bin:/bin' | cut -d':' -f2`
    Retourne `/usr/bin`
  - `ls -l | tr -s ' ' | cut -d' ' -f3,6,8`
    Affiche les fichiers du répertoire courant précédés par le propriéraire et la date de dernière modification, à raison d'un fichier par ligne

La commande tee

`tee [-a] fichier`

- La commande `tee` peut être utilisée pour envoyer en même temps la sortie standard vers l'écran et vers un fichier.
- `make | tee compil.log`
  Lance la commande `make` et stocke sa sortie dans le fichier `compil.log`
- `make install | tee -a compil.log`
  Lance la commande `make install` et rajoute sa sortie à la fin du fichier `compil.log`

La notion de processus (1)

- Depuis sa création, Unix prend en charge le vrai multi-tâche préemptif
- Faculté de lancer de nombreuses tâches en parallèle, et de les interrompre même si elles ont corrompu leur propre état ou leur propres données
- Faculté de choisir quels programmes précis vous lancez
- Faculté de choisir les entrées utilisées par vos programmes, et de choisir où vont leurs sorties

La notion de processus (2)

Sous Unix, tout ce qui s'exécute est un processus

- Un processus :
  - une commande, un groupe de commande ou un script en cours d'exécution
  - son contexte d'exécution : entrées/sorties, variables d'environnement
- Chaque processus :
  - s'exécute de façon indépendante et en temps partagé
  - possède ses propres entrées/sorties
  - peut être individuellement suspendu, réactivé ou tué

Processus en tâche de fond

Même mode d'utilisation dans tous les shells

- Très utile :
  - Pour les tâches en ligne de commande dont les résultats peuvent être examinés plus tard, en particulier celles qui prennent beaucoup de temps.
  - Pour lancer des applications graphiques depuis la ligne de commande et les utiliser ensuite à la souris.
- Démarrer une tâche et ajouter `&` à la fin

```
find / -name '*.gif' > ~/images.txt &
```

Contrôle des tâches de fond

- `jobs`
  Fournit la liste des tâches de fond issues du même shell

```
[1]-  Running find / -name '*.gif' > ~/images.txt &
[2]+  Running make install > /dev/null &
```

- `fg`
  `fg %n`
  Faire de la dernière (ou `n`-ième tâche) de fond la tâche courante

- Mettre la tâche courante en arrière plan :
  `<Ctrl Z>`
  `bg`

- `kill %n`
  Interrompt la nième tâche

Exemples de contrôle de tâches

```
[hal@home/bob]  jobs
[1]-  Running find / -name '*.gif' > ~/images.txt &
[2]+  Running make install > /dev/null &

[hal@home/bob] fg
make install

[hal@home/bob] <Ctrl Z>
[2]+  Stopped make install

[hal@home/bob] bg
[2]+ make install &

[hal@home/bob] kill %1
[1]+  Terminated find / -name '*.gif' > ~/images.txt &
```

Liste de tous les processus

- `ps -ux`
  Affiche tous les processus appartenant à l'utilisateur courant.
- `ps -aux` (remarque: `ps -edf` sur systèmes System V)
  Affiche tous les processus existant sur le système
-
```
ps -aux | grep bart | grep bash
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
bob       3039  0.0  0.2   5916 1380 pts/2    S    14:35   0:00 /bin/bash
bob       3134  0.0  0.2   5388 1380 pts/3    S    14:36   0:00 /bin/bash
bob       3190  0.0  0.2   6368 1360 pts/4    S    14:37   0:00 /bin/bash
bob       3416  0.0  0.0      0    0 pts/2    RW   15:07   0:00 [bash]
```

- PID: (Process ID) Identifiant du processus
  VSZ: (Virtual SiZe)Taille virtuelle du processus (code + données + pile)
  RSS: (ReSident Size) Nombre de Ko occupés en mémoire
  TTY: (TeleTYpe) Terminal
  STAT: Statut: R (Runnable: exécutable), S (Sleep: endormi), W (paging: en cours de pagination), Z (Zombie)...

Arrêt de processus

- `kill pid ...`
  Envoie un signal d'arrêt aux processus spécifiés. Cela permet aux processus de sauvegarder leurs données et s'arrêter eux-mêmes. A utiliser en premier recours.
  Exemple:
  `kill 3039 3134 3190 3416`
- `kill -9 pid ...`
  Envoie un signal d'arrêt immédiat. Le système lui-même se charge d'arrêter les processus. Utile quand une tâche est vraiment bloquée (ne répond pas à un `kill` simple)
- `killall [-signal] commande`
  Arrête toutes les tâches exécutant `commande`. Exemple:
  `killall bash`
- `kill -9 -1`
  Arrête tous les processus de l'utilisateur courant (`-1` : tous les processus)

Activité en temps réel des processus

`top` – Affiche les processus les plus actifs, triés par utilisation du proc.

```
top - 15:44:33 up  1:11,  5 users,  load average: 0.98, 0.61, 0.59
Tasks:  81 total,   5 running,  76 sleeping,   0 stopped,   0 zombie
Cpu(s): 92.7% us,  5.3% sy,  0.0% ni,  0.0% id,  1.7% wa,  0.3% hi,  0.0% si
Mem:    515344k total,   512384k used,     2960k free,    20464k buffers
Swap:  1044184k total,        0k used,  1044184k free,   277660k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
3809 jdoe      25   0  6256 3932 1312 R 93.8  0.8   0:21.49 bunzip2
2769 root      16   0  157m  80m  90m R  2.7 16.0   5:21.01 X
3006 jdoe      15   0 30928  15m  27m S  0.3  3.0   0:22.40 kdeinit
3008 jdoe      16   0  5624  892 4468 S  0.3  0.2   0:06.59 autorun
3034 jdoe      15   0 26764  12m  24m S  0.3  2.5   0:12.68 kscd
3810 jdoe      16   0  2892  916 1620 R  0.3  0.2   0:00.06 top
```

- L'ordre de tri peut être changé en tapant
  `M`: utilisation Mémoire, `P`: %CPU, `T`: Temps d'exécution.
- On peut arrêter une tâche en tapant `k` (kill) et son numéro

Commandes time, times et uptime

- `time commande`
  Exécute la commande commande et affiche le temps consommé par cette exécution.
  Exemple :
```
$ time ls -R / &> /dev/null
real    0m7.917s
user    0m1.698s
sys     0m2.294s
```

- `times`
  Affiche les temps cumulés pour tous les processus lancés depuis le début de la session

- `uptime`
  Affiche le temps depuis lequel le système est actif (donc le temps depuis le dernier *boot*) ainsi que les valeurs moyennes de charge du système

- Il existe également de nombreuses applications graphiques pour contrôler les paramètres systèmes (charge, processus) comme par exemple **KSysGuard** sous KDE

Groupage de commandes

- Séquence de commandes avec le symbole `;`
  `echo “mes fichiers”; ls;  echo ”--------”`
- Groupage logique avec les opérateurs `||` (ou) et `&&` (et)
  - `cat ~/infos 2>/dev/null || echo "erreur"`
    N’exécute `echo` que si la première commande échoue
  - `ls ~/infos 2>/dev/null && cat ~/infos`
    N’affiche le contenu d'`infos` que si la commande `ls` réussit
- Groupage en sous-shell avec les parenthèses "`(`" et "`)`"
  `pwd; ( cd ..; pwd ); pwd`
  Affiche successivement les chemins du répertoire courant, du répertoire père, puis à nouveau du répertoire courant

Exécutions différées

- `sleep 60`
  Attend 60 secondes (ne consomme pas de ressources système)
- `at 6:00pm today chmod o-rx /net/devoirs`
  Enlève aujourd'hui à 18h les droits en lecture et en accès au répertoire /`net/devoirs` pour *"others"*
- `nohup /home/bob/bin/compile_project`
  Lance la commande `compile_project` en batch et place le résultat dans le fichier `nohup.out`
- `crontab`
  Gère la `crontab` de l'utilisateur pour lancer des commandes en temps différé

IV. Compléments sur le shell

- Variables
- Alias
- Substitution de commandes
- Fichiers de configuration zsh
- La commande for

Variables d'environnement

- Les shells permettent à l'utilisateur de définir des *variables*.
  Celles-ci peuvent être réutilisées dans les commandes shell.
  Convention : les noms sont en minuscules
- Vous pouvez aussi définir des *variables d'environnement*: des variables qui sont sont aussi visibles depuis les scripts ou les exécutables appelés depuis le shell.
  Convention : les noms sont en majuscules
- La commande `env`
  Affiche toutes les variables d'environnement existantes ainsi que leur valeur

Variables de shell

- Un seul type possible : la chaîne de caractères
- Pas besoin d'être déclarée pour recevoir une valeur par affectation

```
une_variable=100
```

  - **Attention** : il ne faut **PAS** mettre d'espace entre la variable, le signe = et la valeur

- Utilisation d'une syntaxe particulière pour accéder à la valeur

```
echo $une_variable
```

- Pour qu'une variable garde sa valeur initiale dans un sous-shell

```
export une_variable
```

Exemples de variables de shell

Variables de shell

- ```
  projdir=/home/bob/unix/projet
  ls -la $projdir; cd $projdir
  ```

Variables d'environnement

- `cd $HOME`
- `echo $PWD`
- `echo $PRINTER`
- `export JAVA_SOURCE=$HOME/java/src`


  ```

Variables d'environnement standards

Utilisées par de nombreuses applications

- `LD_LIBRARY_PATH`
  Chemin de recherche de bibliothèques partagées
- `DISPLAY`
  Écran sur lequel afficher les applications X (graphiques)
- `EDITOR`
  Éditeur par défaut (vi, emacs...)
- `HOME`
  Répertoire de l'utilisateur courant.
- `HOST`
  Nom de la machine locale
- `MANPATH`
  Chemin de recherche des pages de manuel.
- `PATH`
  Chemin de recherche des commandes
- `PRINTER`
  Nom de l'imprimante par défaut
- `SHELL`
  Nom du shell courant
- `TERM`
  Nom du terminal / mode courant
- `USER`
  Nom de l'utilisateur courant

Variables d'environnement PATH

- `PATH`
  Spécifie l'ordre de recherche des commandes pour le shell

  ```
  ~/bin:/usr/local/bin:/usr/bin:/bin:/sbin
  ```

- `LD_LIBRARY_PATH`
  Spécifie l'ordre de recherche pour les bibliothèques partagées (codes binaires partagés par les applications, comme la bibliothèque C) pour `ld`

  ```
  /usr/local/lib:/usr/lib:/lib:/usr/X11R6/lib
  ```

- `MANPATH`
  Spécifie l'ordre de recherche pour les pages de manuel

  ```
  /usr/local/man:/usr/share/man
  ```

Variables et substitutions

- Substitution de la valeur d'une variable
  `echo $HOST`
- Substitution protégée de la valeur d'une variable
  `TMP=${USER}_$$` (et non `$USER_$$`)
- Substitution du résultat d'une commande
  `nb_files=$(ls | wc -l)` (ou `` `ls | wc -l` ``)
- Autres substitutions
  Dans les substitutions suivantes, si la variable `var` a une valeur, alors
  `${var-VAL}` substitue `var`, sinon substitue VAL
  `${var=VAL}` substitue `var`, sinon donne à `var` la valeur `VAL` et substitue `var`
  `${var?VAL}` substitue `var`, sinon affiche `VAL` et quitte le shell
  `${var+VAL}` substitue `VAL`, sinon ne substitue rien

Alias

Les shells vous permettent de définir des *alias* (des raccourcis pour des commandes que vous utilisez très souvent)

Exemples

- `alias ls='ls -F'`
  Utile pour toujours lancer des commandes avec certains paramètres
- `alias rm='rm -i'`
  Utile pour faire que `rm` demande toujours une confirmation
- `alias cherche='find $HOME -name'`
  Utile pour remplacer des commandes utilisées régulièrement
- `alias cia='. /home/sydney/env/cia.sh'`
  Utile pour initialiser rapidement un environnement
  (`.` est une commande shell pour exécuter le contenu d'un script shell)
- En **zsh** les alias sont définis dans le fichier `.zshrc`

Les fichiers de zsh (1)

Des scripts shell sont lus automatiquement à chaque fois qu'un shell `zsh` est lancé ou terminé

- Il existe toujours deux scripts de nom `zxxx`
  - Le script `/etc/zxxx` :
    non modifiable par l'utilisateur et lancé en premier
  - Le script `~/.zxxx` :
    modifiable par l'utilisateur et lancé à la suite
- Permet de personnaliser son environnement
- Les scripts `~/.zxxx` ne sont pas obligatoires

Les fichiers de zsh (2)

- `/etc/zshenv` et `~/.zshenv`
  Scripts shell lus à chaque fois qu'un shell `zsh` est lancé
- `/etc/zprofile` et `~/.zprofile`
  Scripts shell lus si le shell lancé est un shell de login
- `/etc/zshrc` et `~/.zshrc`
  Scripts shell lus si le shell lancé est interactif
- `/etc/zlogin` et `~/.zlogin`
  Scripts shell lus si le shell lancé est un shell de login
- `/etc/zlogout` et `~/.zlogout`
  Scripts shell lus à la sortie d'un shell de login

Les fichiers de zsh (3)

- `.zshenv`
  Contient les initialisations des variables d'environnement
- `.zprofile`
  Ne doit pas être utilisé avec le `.zlogin` !
- `.zshrc`
  Contient les *alias*, les *setopt* et les *fonctions* `zsh`
- `.zlogin`
  Contient le lancement de divers programmes
- `.zlogout`
  Contient des commandes spécifiques à la sortie d'un shell login (nettoyage)

La commande for (1)

- Exécute les commandes `commandes` en donnant successivement à la variable `var` les valeurs `val_1`,..., `val_2`, `val_n`

```
for var in val_1 val_2 ... val_n
  do
     commandes
  done
```

- `val_i` peut être une chaîne de caractères quelconque incluant des caractères génériques (wildcards)

```
for file in *; do ... done
```

- En l'absence de `val_i` , la variable `var` prend comme valeurs successives les valeurs des **paramètres** du script :

```
for arg
  do echo $arg; done
```

La commande for (2)

- Pour afficher les fichiers exécutables du répertoire courant :

```
for file in *
  do [ -f $file -a -x $file ] && echo $file; done
```

- Pour compiler des fichiers sources C du répertoire courant :

```
for file in prog1.c prog2.c prog3.c
  do gcc -c $file; done
```

- Pour imprimer tous les fichiers sources C du répertoire courant :

```
for file in *.c
  do lpr $file; done
```

- Pour tuer tous les processus de l'utilisateur de nom "`emacs`" :

```
for proc in $(ps x | grep emacs | cut -d' ' -f2)
  do kill $proc; done
```

La commande seq

- Affiche une séquence de nombres

  `seq i j`
  Si `i` est plus petit ou égal à `j`, affiche les entiers `i`, `i+1`,...,`j`, sinon ne fait rien

  `seq -s char i j`
  Remplace le séparateur (un '`\n`' par défaut) par le caractère `char`

  `seq i k j`
  Si `i` est plus petit ou égal à `j`, affiche les entiers `i`, `i+k`,...,`j`, pour tout les `k` tels que `i+k` ne dépasse pas `j`, sinon ne fait rien

- Utilisé la plupart du temps avec l'itération `for`

```
for i in $(seq 1 $max); do
  mkdir "tmp$i"
done
```

V. Le monde Linux

- Le projet GNU et la licence GPL
- Les distributions Linux
- Linux et ses concurrents

Le projet GNU

GNU = GNU is Not Unix

- Projet de réaliser un système à la Unix entièrement libre
- Lancé en 1984 par Richard Stallman, un chercheur du MIT, à une époque où les sources d'Unix n'étaient plus libres d'accès
- Composants initiaux: compilateur C (gcc), make (GNU make), Emacs, bibliothèque C (glibc), outils de base (ls, cp ...)
- Cependant, en 1991, le projet GNU n'avait toujours pas de noyau et tournait sur des Unix propriétaires

Les Logiciels Libres

- GNU, Linux et de nombreux autres programmes sont des *Logiciels Libres*
- Le *Logiciel Libre* fournit à son utilisateur les 4 libertés suivantes :
  - La liberté d'exécuter le programme, pour quelque but que ce soit
  - La liberté d'étudier son fonctionnement, et de l'adapter à ses besoins
  - La liberté de redistribuer des copies pour aider autrui
  - La liberté d'améliorer le programme, et de partager ses améliorations avec autrui
- Voir http://www.gnu.org/philosophy/free-sw.fr.html

La licence GPL

- Les licences *Copyleft* utilisent les lois sur le copyright pour permettre aux auteurs d'exiger que toute version modifiée reste aussi un logiciel libre
- La licence GNU GPL (GNU General Public License) exige que toutes modifications et travaux dérivés soient aussi publiés sous licence GPL
  - Ne s'applique qu'aux logiciels publiés
  - Tout programme utilisant du code sous la GPL (statiquement ou même dynamiquement) est considéré comme une extension ce code
- Pour plus de détails :
  - Copyleft: http://www.gnu.org/copyleft/copyleft.html
  - Questions / réponses sur la GPL: http://www.gnu.org/licenses/gpl-faq.html

GNU Linux

- Noyau libre semblable à un noyau Unix, conçu par Linus Torvalds en 1991
- Le système complet se repose sur les outils GNU : bibliothèque C, gcc, binutils, fileutils, make, emacs...
- Le système complet est donc appelé “GNU / Linux”
- Très tôt partagé comme Logiciel Libre (Licence GPL), ce qui attira des contributeurs et des utilisateurs de plus en plus nombreux
- Depuis 1991, connaît une croissance supérieure à tout autre système d'exploitation (Unix et autres)

Distributions GNU Linux

- Se chargent de publier un ensemble cohérent de versions compatibles du noyau, de la bibliothèque C, des compilateurs, des outils
- Les outils sont disponibles sous forme de *paquetages* qui peuvent facilement être installés, supprimés ou mis à jour. Les dépendances entre outils sont gérées plus ou moins automatiquement
- Distributions commerciales: incluent de l'assistance technique. Le code source est libre, mais les binaires ne sont pas libres d'accès
- Distributions communautaires: sources et binaires sont librement disponibles. Ne comprennent pas d'assistance technique

Distributions Linux

- Vous permettent d'installer Linux dans un emplacement libre sur votre disque dur, tout en gardant Windows ("double démarrage")
- Ont une interface très conviviale qui peut détecter automatiquement détecter la plupart des matériels (pas de pilote à installer)
- Vous permettent de choisir les types d'applications à installer
- Fournissent une interface de configuration conviviale
- Distributions recommandées pour les débutants :
  Fedora Core, Mandriva ou (K)Ubuntu

Distributions commerciales

- Red Hat: http://www.redhat.com/
  La plus populaire. Fiable, sûre, conviviale et facile à installer, prise en charge par tous les fournisseurs de logiciel et de matériel.

- Suse (Novell): http://www.suse.com/
  L'alternative principale. Face à installer et conviviale. Pas encore d'infos sur sa stabilité. Pas encore prise en charge par tous les fournisseurs.

- Mandriva: http://www.mandriva.com/
  Conviviale, facile à installer, plus innovante, mais moins stable (peut-être davantage depuis l'introduction des pré-versions communautaires). Cible principalement les utilisateurs individuels. Peu prise en charge par les fournisseurs.

Distributions communautaires

- Debian: http://debian.org/
  Très stable et sûre, mais plus difficile à configurer et à installer. Peu conviviale pour les utilisateurs. Version stables peu fréquentes (tous les 2 ou 3 ans). Recommandée pour les serveurs

- Fedora Core: http://fedora.redhat.com/
  Stable, sûre, conviviale, facile à installer. Sortie fréquente de nouvelles versions complètes

- Mandriva Community: http://www.mandriva.com/
  Facile à installer, sûre, conviviale, sortie fréquente de versions complètes, mais moins stable (pas assez de tests et de prise en compte des retours des utilisateurs et des testeurs)

- (K)Ubuntu: http://www.ubuntulinux.org/ et http://www.kubuntu.org/
  Nouvelle distribution très facile à installer et facile à configurer.
  Une nouvelle version (promise) tous les six mois. Une distribution très prometteuse !

Autres systèmes Unix libres

GNU / Hurd: http://www.gnu.org/software/hurd/hurd.html

- Outils GNU avec le Hurd, le micro-noyau de GNU
- De plus en plus mûr, mais pas encore assez pour être utilisé par tous. Jusqu'en 2004 surtout utilisé par ses développeurs eux-mêmes.

Famille BSD

- FreeBSD: http://www.freebsd.org/
  Système BSD puissant, multi-plateforme, sûr et populaire.
- OpenBSD: http://openbsd.org/
  Système BSD puissant, multi-plateforme, sûr et populaire.
  Construit pour une fiabilité et une sécurité extrêmes. Populaire pour serveurs sur Internet.

- NetBSD: http://netbsd.org/
  Distribution BSD dont le but est d'être extrêmement portable
  (par exemple disponible sur ARM)

Linux et ses concurrents (1)

Linux est une alternative sérieuse aux systèmes commerciaux

Sécurité

- Sans virus
  La plupart des virus sont conçus pour tirer parti des failles de sécurité de Windows et n'ont aucun effet sur GNU / Linux.
- Décourage les pirates
  Même si vous êtes connecté en permanence à Internet, votre système attire moins les pirates.
- A l'épreuve des virus
  Même si vous exécutiez un virus compatible avec Linux, il n'aurait pas la permission de modifier le système.
- A l'épreuve des erreurs
  Les utilisateurs ne peuvent ni toucher au système ni aux fichiers d'un autre utilisateur. Ils ne peuvent endommager que leurs propres fichiers.

Linux et ses concurrents (2)

Respect de la vie privée

- Votre système ne va pas discrètement recueillir des informations sur les films ou les sites internet que vous préférez

Convivialité

- Vos programmes sont conçus pour des utilisateurs par des utilisateurs. Ils sont mieux susceptibles de satisfaire vos besoins

Liberté

- Les données que vous créez vous appartiennent pour toujours. Elles ne sont pas prisonnières d'une application propriétaire à travers un format propriétaire (parfois breveté)
- On peut facilement contacter les développeurs pour leur suggérer de nouvelles fonctionnalités
- Vous êtes libres d'aider votre entourage en partageant vos programmes avec lui
- Vous êtes libres d'améliorer ces programmes vous mêmes

Linux et ses concurrents (3)

Vous pouvez utiliser Linux pour :

- La bureautique : traitement de texte, tableur, présentations
- Internet : navigation et courrier électronique
- Le multimédia : vidéo, son et graphiques (y compris appareils photo numériques)
- Votre activité professionnelle (développement et programmation)

Vous pouvez utiliser un autre système d'exploitation pour :

- Les jeux : la plupart des jeux grand public ne sont encore conçus que pour Windows ou Mac
- Utiliser des logiciels propriétaires spécifiques ou des cdroms éducatifs
- Utiliser du matériel non encore pris en charge sous Linux

Essayer Linux

Kaella est un cdrom Linux sans installation
http://kaella.linux-azur.org/

- Entièrement en français
- Charge Linux en mémoire, rien n'est installé sur votre disque dur
- D'étonnantes capacités de reconnaissance du matériel
- Plus de 2 Go d'applications disponibles
- Vous pouvez accéder à vos fichiers Windows, les ouvrir et mêmes les modifier avec des applications sous Linux
- Une excellente façon d'essayer et de montrer GNU / Linux
- Permet d'effectuer une installation permanente sur votre disque dur

VI. Epilogue

- **Ressources internet sur Linux**
- **Licence et historique du cours**

Ressources Linux sur internet

Une petite sélection de ressources internet diverses sur Linux

- Le site officiel de Linux
  http://www.linux.org/
- Le site officiel GNU
  http://www.gnu.org/
- Le site de Linux-France
  http://www.linux-france.org/
- La page web du Linux Journal
  http://www.linuxjournal.com/
- Toutes les nouveautés sur Linux
  http://www.linuxcentral.com/
- Un site sur les distributions Linux
  http://distrowatch.com/
- Un site d'informations sur Linux
  http://www.toolinux.com/
- Un site de forums sur Linux
  http://www.linuxquestions.org/
- Un site dédié aux débutants Linux
  http://www.delafond.org/survielinux/
- Un site généraliste sur Linux
  http://linuxfr.org/pub/
- Le site de Linux-Azur
  http://www.linux-azur.org/
- Le site de la conférence Solutions Linux
  http://www.solutionslinux.fr/

Licence

Historique du cours

Les contributions de la version initiale proviennent de M. Opdenacker.
Détails disponibles sur http://free-electrons.com/doc/ChangeLog

- 28 sep. 2004, première publication.
- 20-24 sep. 2004, première session pour Atmel, Rousset (France)

Les contributions de la version actuelle proviennent de M. Gaëtano.
Des ressources diverses associées à ce cours ont disponibles sur
http://www-local.essi.fr/~gaetano/master

- 5 sep. 2005, première session aux MASTERs de Polytech'Nice-Sophia