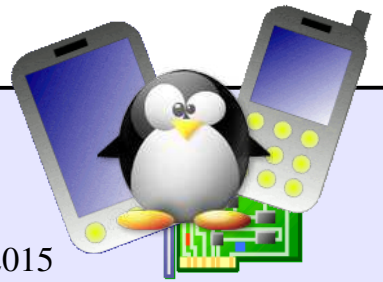
Introduction à Unix et GNU / Linux

Michael Opdenacker
Free Electrons
http://free-electrons.com

Traduction française par Julien Boibessot
Mise à jour Fabien Deleu
(Département GTR de l'IUT de Béthune)

Créer avec OpenOffice.org 2.x



Comment prononcer “Linux?”

Bien sur, chaque pays et/ou langage peuvent avoir leur propre prononciation.

En anglais, il est difficile de deviner!
En fait, voici comment Linus Torvalds le prononce:
http://free-electrons.com/pub/audio/torvalds-says-linux.ogg

Droit de copie

creative commons
COMMONS DEED

Attribution – ShareAlike 2.0

Vous êtes libres :

- de reproduire, distribuer et communiquer cette création au public
- de modifier cette création
- d'utiliser cette création à des fins commerciales

Selon les conditions suivantes :

Paternité. Vous devez citer le nom de l'auteur original.

Partage des Conditions Initiales à l'Identique. Si vous modifiez, transformez ou adaptez cette création, vous n'avez le droit de distribuer la création qui en résulte que sous un contrat identique à celui-ci.

- A chaque réutilisation ou distribution, vous devez faire apparaître clairement aux autres les conditions contractuelles de mise à disposition de cette création.
- Chacune de ces conditions peut être levée si vous obtenez l'autorisation du titulaire des droits.

Ce qui précède n'affecte en rien vos droits en tant qu'utilisateur (exceptions au droit d'auteur: copies réservées à l'usage privé du copiste, courtes citations, parodie...)

Licence : http://creativecommons.org/licenses/by-sa/2.0/legalcode

© Copyright 2006-2004
Michael Opdenacker
michael@free-electrons.com

Sources du document, mises a jour et traductions :
http://free-electrons.com/docs/command-line/

Corrections, suggestions, contributions et traductions sont les bienvenues!

Plus facile à lire avec...

Ce document est le plus facile à lire avec un lecteur PDF récent ou avec OpenOffice.org lui-même! Vous pouvez:

- Utilise les hyperliens internes ou externes.
  Donc n'hésitez pas à cliquer sur ces liens!
- Trouver facilement des pages grâce à la recherche automatique.
- Utiliser les miniatures de pages pour naviguer rapidement dans le document.

Si vous lisez une copie papier ou HTML, vous feriez mieux de récupérer une copie au format PDF ou OpenOffice.org sur http://free-electrons.com/training/intro_unix_linux!

Feuille mémoire des commandes

C'est un compagnon très utile pour cette présentation.

Les exemples des commandes les plus utilisés sont donnés sur juste une feuille.

Suggestions d'utilisation

Coller cette feuille sur votre mur, utilisez-la comme papier peint de bureau, faites-lui un tapis de souris, imprimez-la sur vos vêtements, décomposez-la en signets...

Attention

A tenir éloigné des souris!

Récuperez-la sur
http://free-electrons.com/training/intro_unix_linux

Sommaire (1)

Sommaire (2)

Shells, interpréteur de commandes et interpréteur de fichiers

Sommaire (3)

Entrée/Sortie standard, redirections, pipes

- Entrée et sortie standard, redirections
- Pipes : redirection de la sortie standard à une autre command e
- Erreur standard

Sommaire (4)

Contrôle des tâches

- Parfaite maîtrise des tâches
- programmes en taches de fond, suspendre, reprendre et annul er
- Liste de tous les processus
- Arrêter les processus
- Variables d'environnement
- Variables d'environnement PATH
- alias, fichier .bashrc

Sommaire (5)

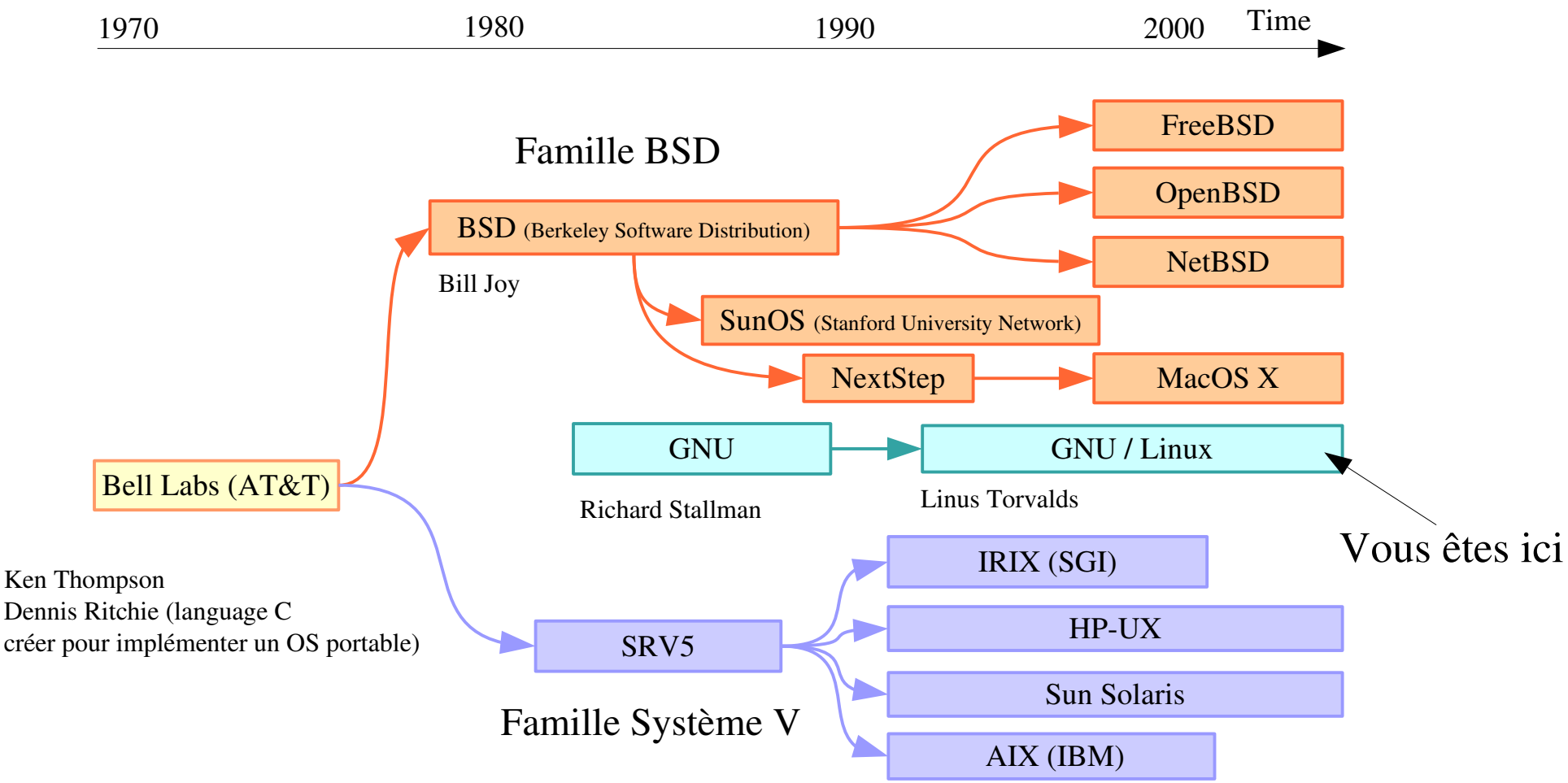
Divers

- Éditeurs de texte
- Compression et archivage
- Impression
- Comparer des fichiers et des répertoires
- Recherche de fichiers
- Récupérer des informations sur les utilisateurs

Sommaire (6)

Introduction

Arbre généalogique d'Unix



La philosophie d'Unix

De nos jours, les systèmes les plus puissants sont basés sur un design vieux de plus de 35 ans!

- Ce qui est simple est beau
- Faire que chaque programme fasse une seule chose et bien
- Préférer la portabilité à l'efficacité
- Éviter les interfaces captives

Abstraction du système

- Noyau: niveau matériel
- Shell: niveau texte
- X Window: niveau graphique

Principale caractéristiques d'Unix

Au départ, Unix à été créé pour les ordinateurs multi-utilisateurs

- 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.
- `root`: utilisateur administrateur avec tous les privilèges
- Multi-tâches
- Supporte plusieurs processeurs
- Extrêmement flexible
- Prise en charge du réseau
- Portable
- Scalable

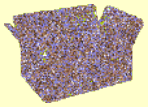
Architecture système d'Unix

applications graphiques des utilisateurs

Navigateur web, office, multimedia...


Applications en ligne de commande

ls, mkdir, wget, ssh, gcc, busybox...




Librairies partagées

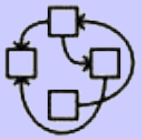
libjpeg, libstdc++, libxml...

Librairie C

GNU C library, uClibc...

noyau système

Linux, Hurd...

Matériel et périphérique



Espace utilisateur

Espace noyau

Matériel

Le projet GNU

GNU = GNU is Not Unix (« GNU N'est pas Unix »)
(un acronyme récursif!)

- Projet de réaliser un système à la Unix entièrement l
- 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

Les logiciels libres garantissent les 4 libertés suivantes aux utilisateurs:

- La liberté d'exécuter le programme, qu'elle que soit le bu
- 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.html

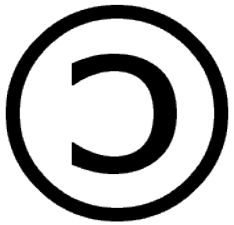
Les logiciels libres sous licence BSD

- Bien sur, elle garantit les 4 libertés aux utilisateurs
- Cependant, elle permet de rendre propriétaire le programme pour soi
- Exemple de licences: BSD, Apache

La licence GNU General Public License (GPL)

La contribution majeure du projet GNU!

- Les licences *Copyleft* utilisent la loi sur le copyright pour permettre aux auteurs d'exiger que toute modification d'un logiciel libre reste un logiciel libre. Voir http://www.gnu.org/copyleft/copyleft.html



- La licence GNU GPL exige que toutes modifications et travaux dérivés soient aussi publiés sous licence GPL:
  - Ne s'appliquent qu'aux logiciels publiés
  - Tous les programmes incluant le code GPL (que ce soit par lien statique ou dynamique) sont considérés comme une extension de ce code.

FAQ GPL: http://www.gnu.org/licenses/gpl-faq.html

GNU Lesser General Public License

http://www.gnu.org/copyleft/lesser.html

- Licence Copyleft similaire à GNU GPL:
  Les modifications doivent être échangés selon les mêmes conditions
- Cependant, permet l'utilisation au sein de programmes propriétaires.
- Utiliser par plusieurs librairies de logiciels libres.
  Exemples:
  glibc, GTK, Wine, SDL

Logiciel libre et open source

Le mouvement des logiciels libres

- Approche fondée sur des principes
- Basé sur la liberté individuelle et l'utilité sociale de la coopération.
- Voir http://www.gnu.org/philosophy/free-software-for-freedom.html

Le mouvement open source

- Approche pragmatique
- Invoque principalement les avantages de partager les sources et fait ses choix selon la supériorité technique.

Bien que les motivations de départ sont différentes, les deux mouvements travaillent très bien ensemble!

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 (pas seulement Unix).

Distributions GNU / Linux

- Se chargent de publier un ensemble cohérent de versions compatibles du noyau, de la bibliothèque C, des compilateurs, des outils... Cela représente un travail très conséquent!
- Les outils sont disponibles sous forme de *paquetages* qui peuvent facilement être installés, supprimés ou mis à jour. Les dépendances entre outils sont gérées 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. Fourni sans assistance technique obligatoire.
- Ne confondez pas la version de distribution avec celle du noyau!

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, conviviale et stable. Obtiens le support des fournisseurs de logiciel et de matériel..

- Mandriva (anciennement Mandrake): http://mandrivalinux.com/
  Conviviale, facile à installer, plus innovante, mais moins stable. Cible principalement les utilisateurs individuels. Peu pris en charge par les fournisseurs de logiciel et de matériel.

Distributions communautaires

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

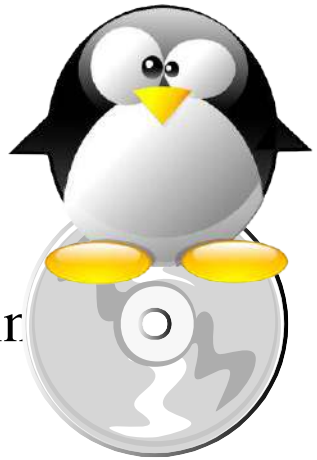
- Ubuntu Linux: http://ubuntu-linux.org/
  La distribution communautaire qui progresse le plus.
  Basé sur Debian mais avec une version stable tout les 6 mois.
  Conviviale pour les utilisateurs. Bonne pour les débutants.

- Debian: http://debian.org/
  Très stable et sûre, mais plus difficile à configurer et à installer. Conviviale pour les développeurs mais pas encore pour les utilisateurs. Version stables pas assez fréquentes (tous les 2 ou 3 ans). La meilleure pour les serveurs, mais pas pour les débutants.

- Mandriva Community: http://mandrivalinux.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 testeurs).

Distributions live (1)

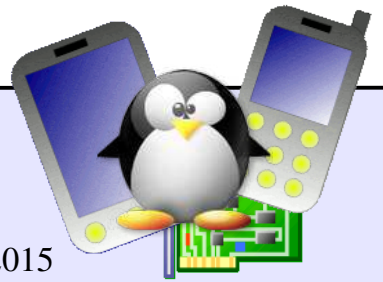
- Linux s'amorce à partir d'un périphérique de stockage (cd-rom, dvd-rom ou usb) et démarre tout à partir de ce périphérique.
- Idéal pour essayer GNU / Linux et les applications des logiciels libres sans avoir à installer quoi que ce soit sur le disque dur!
- Le système est prêt et démarre en 2-3 minutes.
  Plus rapide qu'installer et configurer GNU / Linux!
- Aussi efficace pour récupérer des données lorsque le système d'origin ne démarre plus.
- Utilise un système de compression pour mettre en mémoire 3 à 4 fois la capacité de stockage !



Liste des distributions live : http://frozentech.com/content/livecd.php

Distributions live (2)

- Knoppix: http://knoppix.net/
  La plus populaire. Disponible en CD et DVD.
  Idéale pour l'auto configuration de votre matériel!
- Ubuntu: http://ubuntu-linux.org/
  Distribue un CD live avec chaque version (tous les 6 mois).



Autres systèmes Unix libres (1)

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'à présent (2005), 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.
  Disponible sur ARM et autres

Autres systèmes Unix libres (2)

Famille Système V

- OpenSolaris: http://opensolaris.org/
  Le noyau open source de Sun Solaris.
  A débuté en juin (2005). Pas encore de version stable.

Autres

- eCos: http://ecos.sourceware.org/
  Système embarqué à temps réel très léger fourni par Red Hat / Cygnus.
  API compatible avec POSIX.

Système de fichiers Unix

Tout est fichier

Presque tous dans Unix est un fichier!

- Fichiers ordinaires
- Répertoires
  Les répertoires ne sont juste que des fichiers listant plusieurs fichiers
- Liens symboliques
  Fichiers faisant référence au nom d'un autre fichier
- Périphériques et dispositifs
  La lecture et l'écriture à partir d'un dispositif se fait comme un fichier
- Pipes
  Utiliser pour mettre en cascade plusieurs programmes
  ```
  cat *.log | grep error
  ```
- Sockets
  Communication inter processus

Noms de fichiers

Depuis le début d'Unix, les noms de fichiers ont les caractéristiques suivantes:

- Sensibles aux majuscules / minuscules
- Pas de longueur limite évidente
- Peuvent contenir tous caractères (incluant l'espace, à l'exception de `/`).
  Les types de fichiers sont stockés dans un fichier (“nombre magique”).
  Les extensions d'un nom de fichier n'ont pas besoin et ne sont pas interprétés. Ils sont justes utilisés pour les utilisateurs .
- Exemples de noms de fichiers:

```
README          .bashrc         Windows Buglist
index.htm       index.html      index.html.old
```

Chemins de fichiers («path»)

Un *chemin* est une séquence de répertoires emboîtés avec un fichier ou un répertoire à la fin, séparés par le caractère `/`

- Chemin relatif: `documents/fun/microsoft_jokes.html`
  Relatif au répertoire courant
- Chemin absolu:
  `/home/bill/bugs/crash9402031614568`
- `/` : *répertoire racine («root»).*
  Le début des chemins absolus pour tous les fichiers du système de fichiers (même pour les fichiers sur des périphériques externes ou de partage réseau).

Structure de fichiers dans GNU/Linux (1)

Rien d'imposé par le système. Peut varier d'un système à l'autre, même entre deux installations de GNU / 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 utilisateur
`/lib/`	Bibliothèques de base du système (partagées)

Structure de fichiers dans GNU/Linux (2)

`/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 GNU/Linux (3)

`/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)...

Interpréteur de commandes et interpréteur de fichiers

Interpréteurs de commandes

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

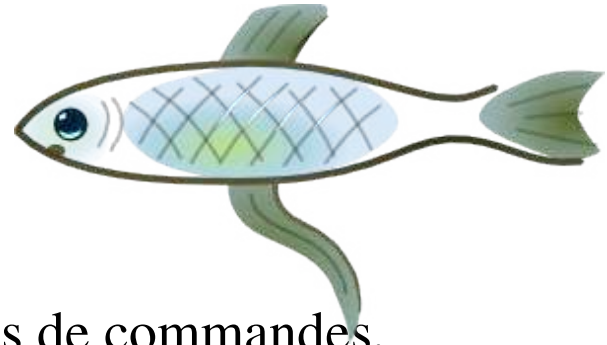
Interpréteurs les plus connus

Interpréteurs de commandes les plus connus et les plus populaires

- sh: Le Bourne shell (obsolète)
  Le shell de base qu'on trouve traditionnellement dans les systèmes Unix, par Steve Bourne.
- 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 (le plus populaire)
  Une version améliorée de sh avec de nombreuses fonctions nouvelles.

fish: un bon nouvel interpréteur de commandes

Le Friendly Interactive SHell
http://roo.no-ip.org/fish/



- Caractéristiques standards: historique,
  complète les noms de commandes et de fichiers...
- Apporte de nouvelles fonctionnalités: complète les options de commandes,
  description des commandes, syntaxe mise en valeur..
- Facilite l'ouverture de tous les fichiers: fournit une commande `open`.
- Syntaxe plus facile et consistante (pas conforme à POSIX)
  Rend plus facile la création de script shell.

Les débutants en ligne de commande peuvent apprendre plus facilement! Même les utilisateurs expérimentés devraient trouvés ce shell très pratique.

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

Substitutions sur noms de fichiers

Plus facile à présenter par des exemples!

- `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 `.*`
- `cat ?.log`
  Affiche le contenu de tous les fichiers dont le nom commence par 1 caractère et finit par `.log`

Répertoires spéciaux (1)

- 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 loin)
- Ainsi `./lisezmoi.txt` et `lisezmoi.txt` sont équivalents

`../`

- 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)

`~/`

- 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.

`~sydney/`

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

Les commandes CD et PWD

- `cd <dir>`
  Change le répertoir courant en `<dir>`
- `pwd`
  Affiche le répertoire courant ("répertoire de travail")

La commande cp

- `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` (interactif)
  Demande confirmation à l'utilisateur dans le cas où le fichier de destination existe déjà
- `cp -r <rep_orig> <rep_dest>` (récursive)
  Copie du répertoire tout entier

Copie intelligente avec rsync

rsync («remote sync»: sync. à distance) a été conçu pour synchroniser des répertoires sur 2 machines reliées par un lien à faible débit.

- Ne copie que les fichiers qui ont changé. Les fichiers de taille identique sont comparés au moyen de sommes de contrôle.
- Ne transfère que les blocs qui diffèrent au sein d'un fichier!
- Peut compresser les blocs transférés
- Conserve les liens symboliques et les permissions sur les fichiers: également très pratique pour les copies sur la même machine.
- Peut fonctionner à travers ssh (shell sécurisé). Très pratique pour mettre à jour le contenu d'un site Internet, par exemple.

Exemples rsync (1)

- `rsync -a /home/arvin/agents_sd6/ /home/sydney/vrac/`

  `-a`: mode archive. Équivalent à `-rlptgoD`... Un moyen facile de dire que vous voulez de la récursion et souhaitez préserver presque tout.

- `rsync -Pav --delete /home/steve/idées/ /home/bill/mes_idées/`

  `-P`: `--partial` (garder les fichiers partiellement transférés) et `--progress` (afficher la progression du transfert)

  `--delete`: effacer les fichiers à l'arrivée qui n'existent plus à la source.

  Attention: les noms de répertoires doivent finir par `/` . Sinon, vous obtenez un répertoire `mes_idées/idées/` à la destination.

Exemples rsync (2)

- Copie vers une machine distante

```
rsync -Pav /home/bill/legal/arguments/ \
bill@www.sco.com:/home/legal/arguments/
```

Un mot de passe sera demandé à l'utilisateur `bill`.

- Copie depuis une machine distante à travers ssh

```
rsync -Pav -e ssh
homer@cuve.duff.com/prod/bière/ \
frigo/homer/bière/
```

On demandera à l'utilisateur `homer` le mot de passe de sa clé ssh.

Les commandes mv et rm

- `mv <ancien_nom> <nouveau_nom>` (“move”: déplacer)
  Change le nom du fichier ou du répertoire donné
- `mv -i` (interactif)
  Si le fichier existe déjà, demander confirmation à l’utilisateur
- `rm fich1 fich2 fich3 ...` (“remove”: supprimer)
  Supprime les fichiers donnés
- `rm -i` (interactif)
  Demande toujours à l’utilisateur de confirmer les suppressions
- `rm -r rep1 rep2 rep3` (récursif)
  Suppression des répertoires donnés et de tout leur contenu

Création et suppression de répertoires

- `mkdir rep1 rep2 rep3 ...` (“make dir”: créer rép.)
  Crée des répertoires avec les noms spécifiés
- `rmdir rep1 rep2 rep3 ...` (“remove dir”: suppr. rép.)
  Supprime les répertoires spécifiés
  Sécurité: ne fonctionne que quand les répertoires sont vides
  Alternative: `rm -r`

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)
  Fait plus que `more` avec moins!
  Ne lit pas le fichier entier avant de commencer à afficher
  Permet de remonter en arrière dans le fichier (commande "`?`")

Les commandes head et tail

- `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: suivre)
  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")
- Exemples

```
head bogues_windows.txt
tail -f vulnérabilités_outlook.txt
```


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 sort

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

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:
  `biographie_anakin_skywalker -> biographie_darth_vador`
- 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

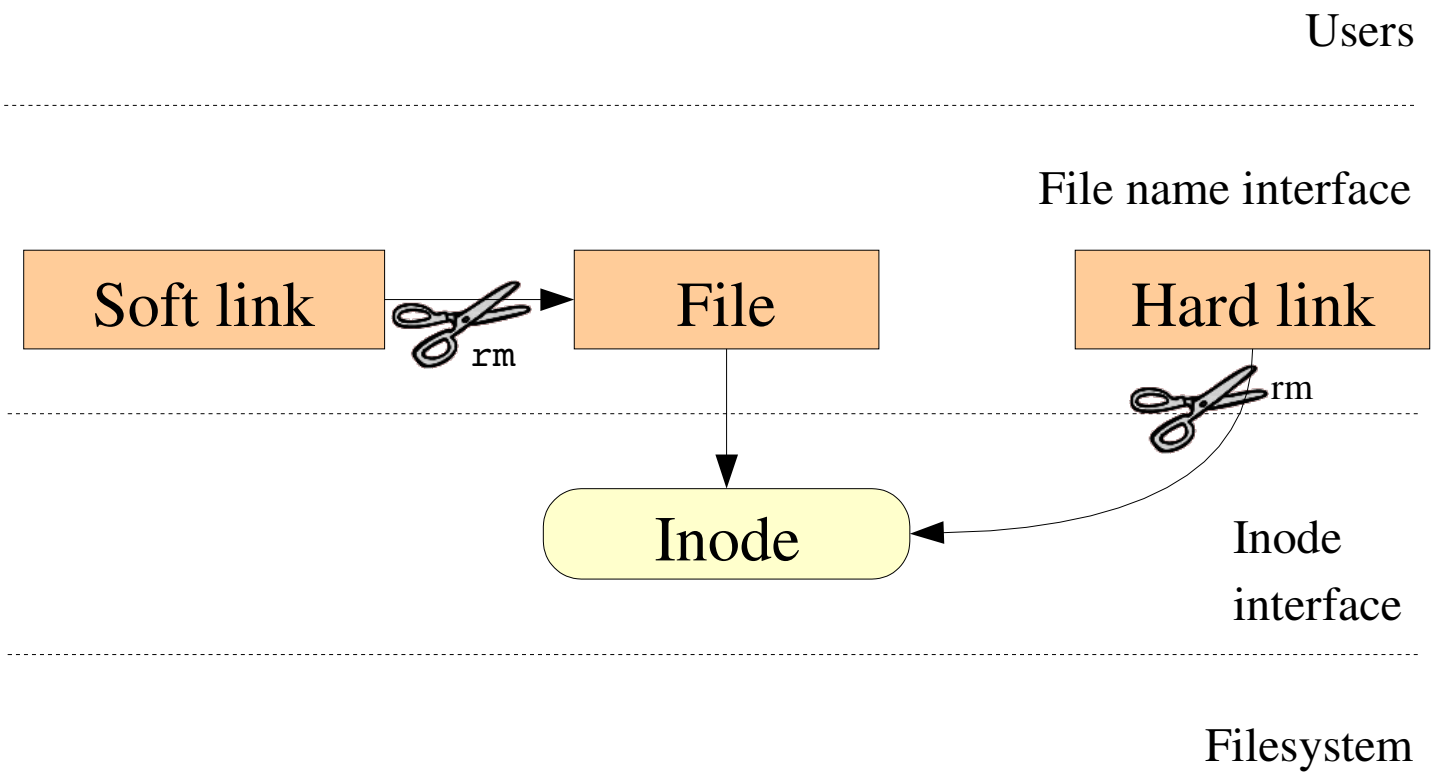
- Pour créer un lien symbolique (même ordre que dans `cp`):
  `ln -s nom_fichier nom_lien`
- Pour créer un lien vers un fichier dans un autre répertoire, avec le même nom:
  `ln -s ../LISEZ_MOI.txt`
- Pour créer plusieurs liens d'un coup dans un dossier donné:
  `ln -s fich1 fich2 fich3 ... rep`
- Pour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

Permet de 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`: read)
- Accès en écriture (`w`: write)
- 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`: others): 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 vos amis ni par vous-même. Droits d'accès parfaits pour un piège...

chmod: modifier les permissions

- `chmod <permissions> <fichiers>`
  2 formats pour les droits d'accès:
- Format en base 8 (abc):
  `a,b,c = r*4+w*2+x` (`r`, `w`, `x`: booléens)
  Exemple: `chmod 644 <fichier>`
  (`rw` pour `u`, `r` pour `g` et `o`)
- Format symbolique. Facile à comprendre par des exemples:
  `chmod go+r`: ajouter droit en lecture au groupe et aux autres
  `chmod u-w`: supprimer droit d'écriture pour le propriétaire
  `chmod a-x`: (a: all = tous). 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 ajouter des droits 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`.
- Afficher par `ls -l` avec un caractère `t`

Entrée et sortie standard, redirections, pipes

Sortie standard

Plus de détails sur les sorties des commandes

- Toutes les commandes qui sortent du texte sur votre terminal le font en écrivant sur leur *sortie standard*.
- La sortie standard peut être écrite (redirigée) dans un fichier en utilisant le symbole >
- La sortie standard peut être rajoutée à la fin d'un fichier existant par le symbole >>

Exemples de redirection de sortie

- `ls ~saddam/* > ~gwb/weapons_mass_destruction.txt`
- `cat obiwan_kenobi.txt > starwars_biographies.txt`
  `cat han_solo.txt >> starwars_biographies.txt`
- `echo “README: No such file or directory” > README`
  Moyen facile de créer un fichier sans éditeur de texte.
  Également une blague Unix sympathique dans ce cas.

Entrée standard

Plus de détails sur ce que les commandes prennent en entrée

- De nombreuses commandes, quand on ne leur donne pas d'arguments en entrée, peuvent chercher leurs entrées sur l'*entrée standard*.
- `sort`
  `windows`
  `linux`
  `[Ctrl][D]`
  `linux`
  `windows`

  `sort` prend l'entrée standard comme entrée: dans ce cas, ce que vous tapez dans le terminal (terminé par `[Ctrl][D]`)
- `sort < participants.txt`
  L'entrée standard de `sort` est prise dans le fichier indiqué.

Les pipes

- Les pipes Unix sont très utiles pour rediriger la sortie standard d'une commande vers l'entrée standard d'une autre commande.
- Exemples
  - `cat *.log | grep -i error | sort`
  - `grep -ri error . | grep -v "ignored" | sort -u \`
    `> serious_errors.log`
  - `cat /home/*/homework.txt | grep mark | more`
- Il s'agit d'une des fonctionnalités les plus puissantes des shells Unix!

La commande tee

`tee [-a] file`

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

L'erreur standard

- Les messages d'erreur sont normalement envoyés (si le programme est bien écrit) vers l'*erreur standard* au lieu de la sortie standard.
- L'erreur standard peut être redirigée par `2>` ou `2>>`
- Exemple:

```
cat f1 f2 faux_fich > nouv_fich 2> fich_erreur
```

- Note: `1` est le descripteur de la sortie standard, donc `1>` est équivalent à `>`
- On peut rediriger à la fois la sortie et l'erreur standard vers le même fichier en utilisant `&>`

```
cat f1 f2 faux_fich &> fich_entier
```


La commande yes

Utile pour remplir l'entrée standard en utilisant toujours la même chaîne de caractères.

- `yes <string> | <command>`
  Remplit l'entrée standard de `<command>` avec `<string>`
  (`y` par défaut)
- Exemples
  ```
  yes | rm -r dir/
  bank> yes no | credit_applicant
  yes "" | make oldconfig
  ```
  (équivaut à appuyer sur `Entrer` pour accepter les paramètres par défaut)

Périphériques spéciaux

Ils ressemblent à des fichiers, mais

- `/dev/null`
  Le destructeur de données! Fait disparaître toutes données écrites dans ce fichier. Utile pour se débarrasser d'une sortie indésirable (telles que des «logs»):
  ```
  mplayer black_adder_4th.avi &> /dev/null
  ```

- `/dev/zero`
  Les lectures à partir de ce fichiers renverront toujours des caractères `\0`
  Utile pour créer un fichier rempli de zéros:
  ```
  dd if=/dev/zero of=disk.img bs=1k count=2048
  ```

Contrôle de tâches

Parfaite maîtrise des tâches

- Depuis le début, 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.

Processus

“Tout dans Unix est fichier
Tout dans Unix qui n'est pas un fichier est un processus”

Processus

- Instance d'un programme en cours d'exécution
- Plusieurs instances d'un même programme peuvent s'exécuter en même temps
- Données associées aux processus:
  ouvrir un fichier, mémoire allouée, pile, id processus, parent, priorité, état...

Programmes en tâche de fond

Même mode d'utilisation dans tous les shells

- 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: ajouter `&` au bout de votre ligne:

```
trouver_prince_charmant --beau --intelligent --riche &
```

Contrôle des tâches de fond

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

```
[1]-  Running ~/bin/trouver_sens_vie --sans-dieu &
[2]+  Running make mistakes &
```

- `fg`
  `fg %<n>`
  Faire de la dernière / nième 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

```
> jobs
[1]-  Running ~/bin/trouver_sans_vie --sans-dieu &
[2]+  Running make mistakes &

> fg
make mistakes

> [Ctrl] Z
[2]+  Stopped make mistakes

> bg
[2]+ make mistakes &

> kill %1
[1]+  Terminated ~/bin/trouver_sens_vie --sans-dieu
```


Liste de tous les processus

... quel que soit le shell, le script ou le processus qui les ait lancés

- `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
  bart      3039  0.0  0.2   5916  1380 pts/2    S    14:35   0:00 /bin/bash
  bart      3134  0.0  0.2   5388  1380 pts/3    S    14:36   0:00 /bin/bash
  bart      3190  0.0  0.2   6368  1360 pts/4    S    14:37   0:00 /bin/bash
  bart      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)...


  ```

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

Arrêt de processus (1)

- `kill <pids>`
  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 <pids>`
  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 `kill -1`).
- `kill -9 -1`
  Arrête tous les processus de l'utilisateur courant. `-1`: tous les processus.

Arrêt de processus (2)

- `killall [-<signal>] <command>`
  Arrête toutes les tâches exécutant `<commande>`. Exemple: `killall bash`
- `xkill`
  Vous laisse arrêter une application graphique en cliquant dessus! Très rapide! Utile quand vous ne connaissez pas le nom de commande de l'application.

Restauration d'une application graphique plantée

- Si votre application graphique est plantée et que vous ne pouvez plus accéder à votre terminal, ne rebootez pas!
- Il est probable que votre système soit encore intact. Essayer d'accéder à une console texte en appuyant sur les touches `[Ctrl][Alt][F1]` (ou `[F2],[F3]` pour davantage de consoles texte)
- Dans la console texte, vous pouvez arrêter l'application corrompue.
- Une fois fait, vous pouvez retourner à la session graphique en appuyant sur `[Ctrl][Alt][F5]` ou `[Ctrl][Alt][F7]` (suivant de votre distribution)
- Si vous ne pouvez pas identifier le programme corrompu, vous pouvez arrêter tous les processus: `kill -9 -1`
  Vous êtes ensuite redirigé vers l'écran de connexion.

Séquence de commandes

- Possibilité de taper la prochaine commande dans votre terminal même si la commande courante n'est pas terminée.
- Possibilité de séparer plusieurs commandes par le symbole `;` :

```
echo "Vous êtes le plus beau"; sleep 10;
echo "des menteurs"
```

- Conditions: utiliser `||` (ou) ou `&&` (et):

```
more Dieu || echo "Désolé, Dieu n'existe pas"
```

N'exécute `echo` que si la première commande échoue.

```
ls ~sd6 && cat ~sd6/* > ~sydney/recettes.txt
```

N'affiche le contenu des fichiers du répertoire que si la commande `ls` réussit (indique un accès en lecture).

Quotes (1)

Les guillemets («double quotes») peuvent être utilisés pour empêcher le shell d'interpréter l'espace comme un argument de séparation, comme pour empêcher l'expansion de motifs de noms de fichiers.

```
> echo "Hello World"
  Hello World

> echo "You are logged as $USER"
  You are logged as bgates

> echo *.log
  find_prince_charming.log cosmetic_buys.log

> echo "*.log"
  *.log
```

Quotes (2)

Les simples quotes fournissent une fonctionalité similaire, mais ce qui est entre les quotes n'est jamais remplacé

```
> echo 'You are logged as $USER'
  You are logged as $USER
```

Les quotes inversés (`) peuvent être utilisés pour appeler une commande à travers une autre.

```
> cd /lib/modules/`uname -r`; pwd
  /lib/modules/2.6.9-1.6_FC2
```

Elles peuvent aussi être utilisés dans une doubles quotes.

```
> echo "You are using Linux `uname -r`"
  You are using Linux 2.6.9-1.6_FC2
```

Mesure du temps écoulé

- `time find_expensive_housing --near <...command output...>`

  `real` `0m2.304s` (temps écoulé réel)

  `user` `0m0.449s` (temps CPU à executer le programme)

  `sys` `0m0.106s` (temps CPU passé en appels système)

  real = user + sys + *waiting*

  waiting = temps attente E/S + temps d'inactivité (exécution d'autres tâches)

Variables d'environnement

- Les shells permettent à l'utilisateur de définir des *variables*. Celles-ci peuvent être réutilisées dans la commandes shell.
  Convention: noms 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: noms en majuscules
- `env`
  Affiche toutes les variables d'environnement existantes ainsi que leur valeur.

Exemples de variables de shell

Variables de shell (bash)

- ```
  projdir=/home/marshall/gadgets
  ls -la $projdir; cd $projdir
  ```

Variables d'environnement (bash)

- ```
  cd $HOME
  ```
- ```
  export DEBUG=1
  ./chercher_vie_extraterrestre
  ```
  (affiche des informations de mise au point si `DEBUG` est défini)


  ```

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.
- `HOSTNAME`
  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`
  Type du terminal courant
- `USER`
  Nom de l'utilisateur courant

Variables d'environnement PATH

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

```
/home/acox/bin:/usr/local/bin:/usr/kerberos/bin:/usr/bin:/bin:/usr/X11R6/bin:/bin:/usr/bin
```

- `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
```

Mise en garde sur PATH

Il est fortement recommandé de ne pas avoir le répertoire “`.`” dans votre variable d’environnement `PATH`, en particulier pas au début.

- Un intrus pourrait placer un fichier `ls` malveillant dans vos répertoires. Il serait exécuté à chaque appel de `ls` depuis ces répertoires et pourrait s’attaquer à vos données personnelles.
- Si vous avez un fichier exécutable de nom `test` dans un répertoire, il sera utilisé à la place du programme `test` par défaut et certains scripts ne fonctionneront plus correctement.
- Chaque fois que vous entrez dans un nouveau répertoire, le shell perdra du temps à mettre à jour sa liste de commandes disponibles.

Lancez vos propres commandes ainsi: `./test`

Alias

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

Exemples

- `alias ls='ls -la'`
  Utile pour toujours lancer des commandes avec certains paramètres
- `alias rm='rm -i'`
  Utile pour faire que `rm` demande toujours une confirmation
- `alias tor='trouver_objet_rambaldi --vite --risque'`
  Utile pour remplacer de longues 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)

La commande which

Avant de lancer une commande, `which` vous dit où elle est trouvée:

- ```
  bash> which ls
  alias ls='ls --color=tty'
          /bin/ls
  ```
- ```
  tcsh> which ls
  ls:       aliased to ls --color=tty
  ```
- ```
  bash> which alias
  /usr/bin/which: no alias in
  (/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin)
  ```
- ```
  tcsh> which alias
  alias: shell built-in command.
  ```

Fichier .bashrc

- `.bashrc`
  Script shell lu à chaque fois qu'un shell `bash` est lancé.
- Vous pouvez utiliser ce fichier pour définir
  - Vos variables d'environnement par défaut (`PATH`, `EDITOR`...)
  - Vos alias
  - Votre invite de shell ("prompt": voir le manuel de `bash` pour plus de détails)
  - Un message de bienvenue

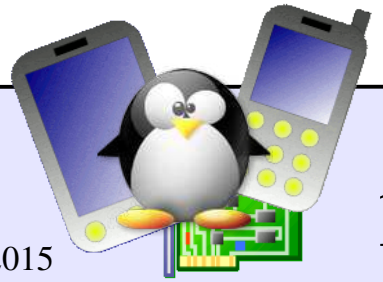
Utilitaires divers

Editeur de commande

- Vous pouvez utiliser les flèches gauche et droite pour bouger le curseur dans la ligne de commande.
- Vous pouvez utiliser `[Ctrl][a]` pour aller au début de la ligne, et `[Ctrl][e]` pour aller à la fin.
- Vous pouvez utiliser les touches haut et bas pour sélectionner les commandes précédentes

Historique de commande (1)

- `history`
  Affiche les dernières commandes que vous avez lancer et leur numéros. Vous pouvez alors utiliser copier et coller.
- Vous pouvez rappeler les dernières commandes:
  `!!`
- Vous pouvez rappeler une commande par son nom
  `!1003`
- Vous pouvez rappeler la dernière commande avec le début d'une chaîne de caractères:
  `!cat`



Historique de commande (2)

- Vous pouvez faire une substitution sur la dernière commande:
  `^more^less`
- Vous pouvez lancer une autre commande avec les mêmes arguments:
  `more !*`

Éditeurs de texte

Éditeurs de texte graphiques

Conviennent pour la plupart des besoins

- nedit
- Emacs, Xemacs

Éditeurs en mode texte uniquement

Souvent indispensables aux administrateurs système et parfaits pour les utilisateurs expérimentés

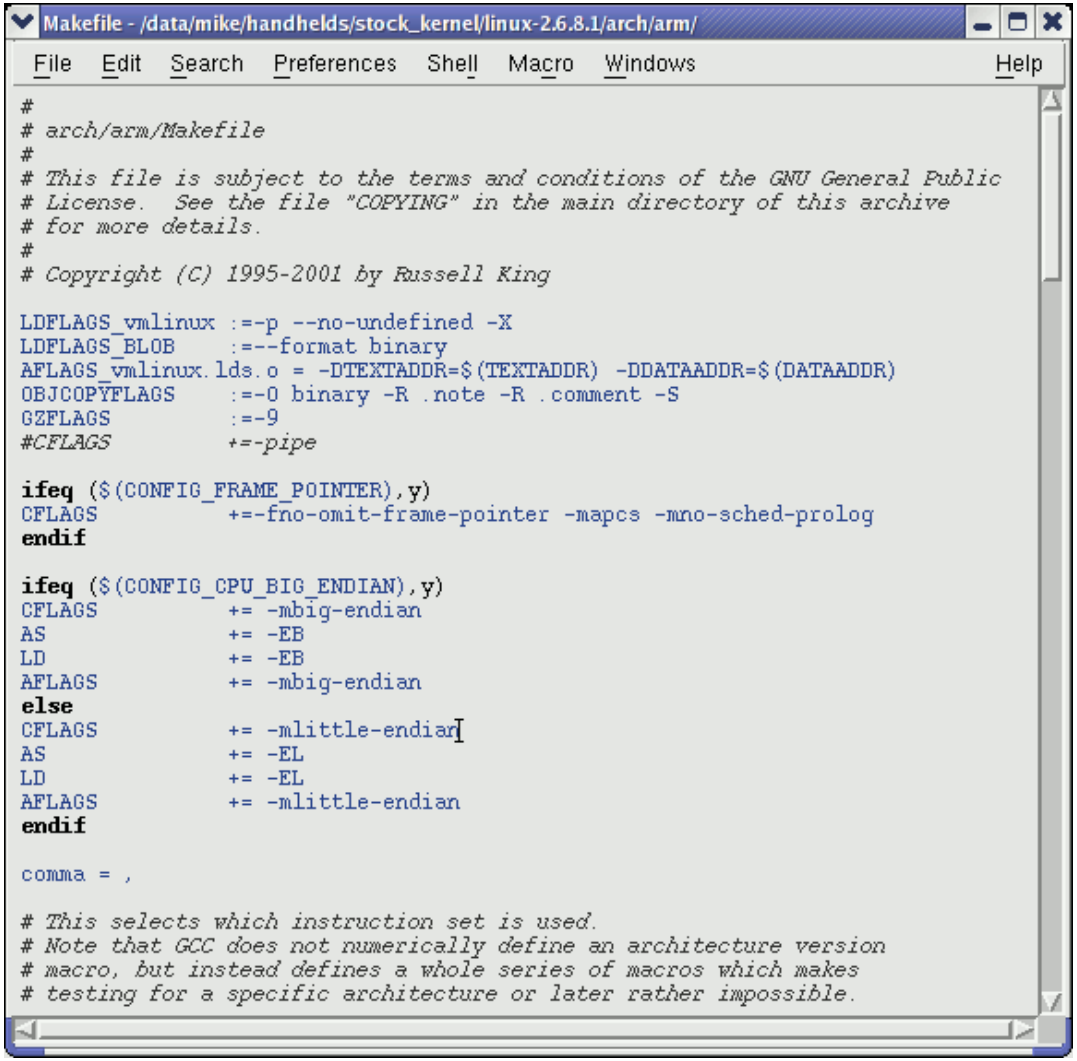
- vi
- nano

L'éditeur de texte nedit (1)

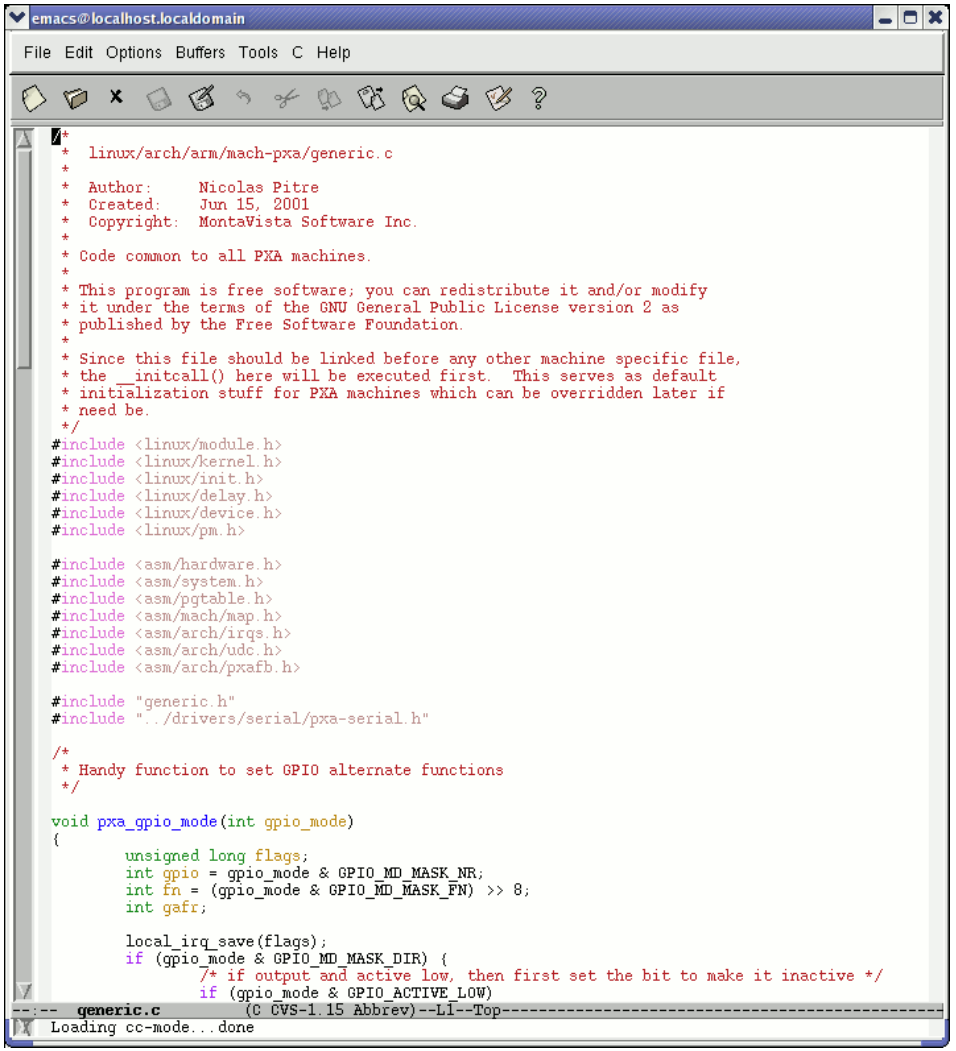
http://www.nedit.org/

- Le meilleur éditeur de texte pour ceux qui ne sont pas experts en `vi` ou `emacs`
- Quelques fonctionnalités attrayantes:
  - Sélection et déplacement de texte très facile
  - Mise en évidence de la syntaxe pour la plupart des langages et des formats. Peut être personnalisé en fonction de vos propres fichiers de journaux (log), pour faire ressortir certains messages d'erreur ou d'avertissement
  - Facile à personnaliser via des menus
- Pas installé par défaut sur toutes les distributions.

Capture d'écran de nedit



Emacs / Xemacs



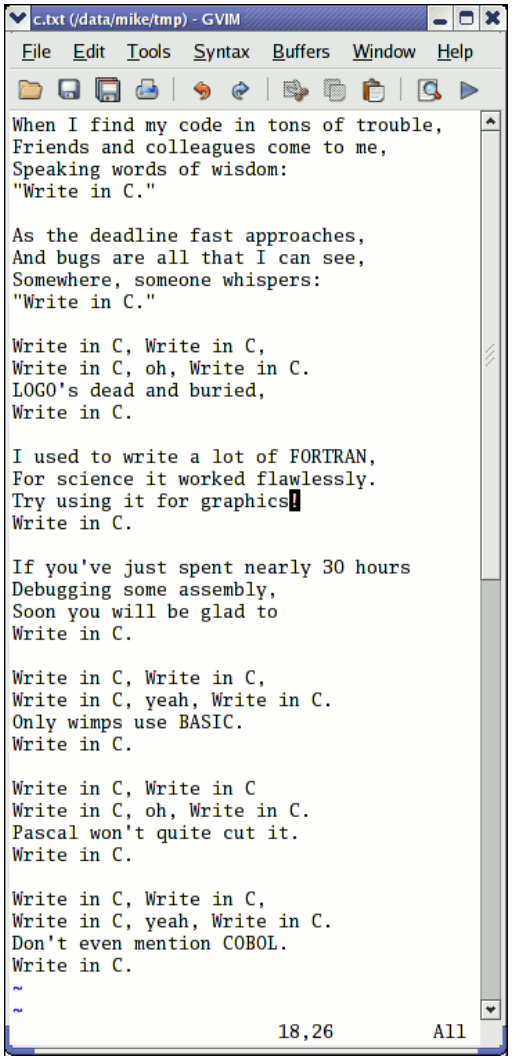
- Emacs et Xemacs sont très semblables (choisissez selon votre goût)
- Fonctionnalités d'éditeur de texte extrêmement puissantes
- Parfait pour les utilisateurs avancés
- Bien moins ergonomique que `nedit`.
- Raccourcis clavier non standards
- Bien plus d'un éditeur de texte (jeu, courrier, shell, navigateur)
- Besoin d'apprendre certaines commandes avancées

vi

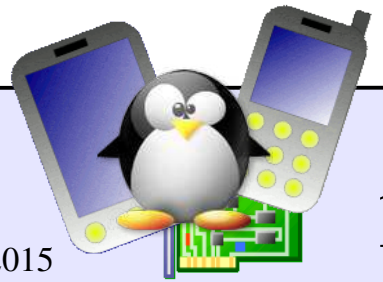
Éditeur de texte en mode texte disponible sur tous les systèmes Unix. Créé avant même l'apparition de la souris.

- Difficile à apprendre pour les débutants habitués aux éditeurs graphiques.
- Très productif pour les utilisateurs avancés
- Souvent incontournable pour modifier des fichiers en administration de système ou dans les systèmes embarqués, quand vous ne disposez que d'une console texte.

vim - vi improved (amélioré)



- Implémentation de `vi` maintenant disponible sur la plupart des stations de travail GNU / Linux
- Apporte de nombreuses fonctionnalités des éditeurs modernes: mise en évidence de la syntaxe, historique de commandes, aide, annulation sans limite et bien d'autres.
- Exemple de fonctionnalité sympa: peut ouvrir directement les fichiers compressés.
- Accompagné d'une interface graphique GTK (`gvim`)
- Hélas, pas un logiciel libre (à cause d'une petite restriction à la liberté d'effectuer des changements)



Commande de base de vi

vi basic commands

Summary of most useful commands

Entering command mode

[ESC] Exit editing mode. Keyboard keys now interpreted as commands.

Moving the cursor

h (or left arrow key) move the cursor left.
l (or right arrow key) move the cursor right.
j (or up arrow key) move the cursor up.
k (or down arrow key) move the cursor down.
[Ctrl] f move the cursor one page forward.
[Ctrl] b move the cursor one page backward.
^ move the cursor to the beginning of the current line.
$ move the cursor to the end of the current line.
G go to the last line in the file.
nG go to line number n.
[Ctrl] G display the name of the current file and the cursor position in it.

Entering editing mode

i insert new text before the cursor.
a append new text after the cursor.
o start to edit a new line after the current one.
O start to edit a new line before the current one.

Replacing characters, lines and words

r replace the current character (does not enter edit mode).
s enter edit mode and substitute the current character by several ones.
cw enter edit mode and change the word after the cursor.
C enter edit mode and change the rest of the line after the cursor.

Copying and pasting

yy copy (yank) the current line to the copy/paste buffer.
p paste the copy/paste buffer after the current line.
P Paste the copy/paste buffer before the current line.

Deleting characters, words and lines

All deleted characters, words and lines are copied to the copy/paste buffer.

x delete the character at the cursor location.
dw delete the current word.
D delete the remainder of the line after the cursor.
dd delete the current line.

Repeating commands

. repeat the last insertion, replacement or delete command.

Looking for strings

/string find the first occurrence of string after the cursor.
?string find the first occurrence of string before the cursor.
n find the next occurrence in the last search.

Replacing strings

Can also be done manually, search and replacing once, and then using n (next occurrence) and . (repeat last edit).

n,p s/str1/str2/g between line numbers n and p, substitute all (g: global) occurrences of str1 by str2.
1,$ s/str1/str2/g in the whole file ($: last line), substitute all occurrences of str1 by str2.

Applying a command several times - Examples

5j move the cursor 5 lines down.
30dd delete 30 lines.
4cw change 4 words from the cursor.
1G go to the first line in the file.

Misc

[Ctrl] l redraw the screen.

Exiting and saving

ZZ save current file and exit vi.
:w write (save) buffer to the current file.
:w file write (save) buffer to the file file.
:q! quit vi without saving changes.

Going further

vi has much more flexibility and many more commands for power users!
It can make you extremely productive in editing and creating text.
Learn more by taking the quick tutorial: just type vimtutor.
Many extra resources are also available on the net.

vi est extrêmement puissant, il contient 30 commandes facile à apprendre et suffisante pour 99% des besoins quotidiens!

Vous pouvez aussi suivre le tutoriel rapide en lançant `vimtutor`.

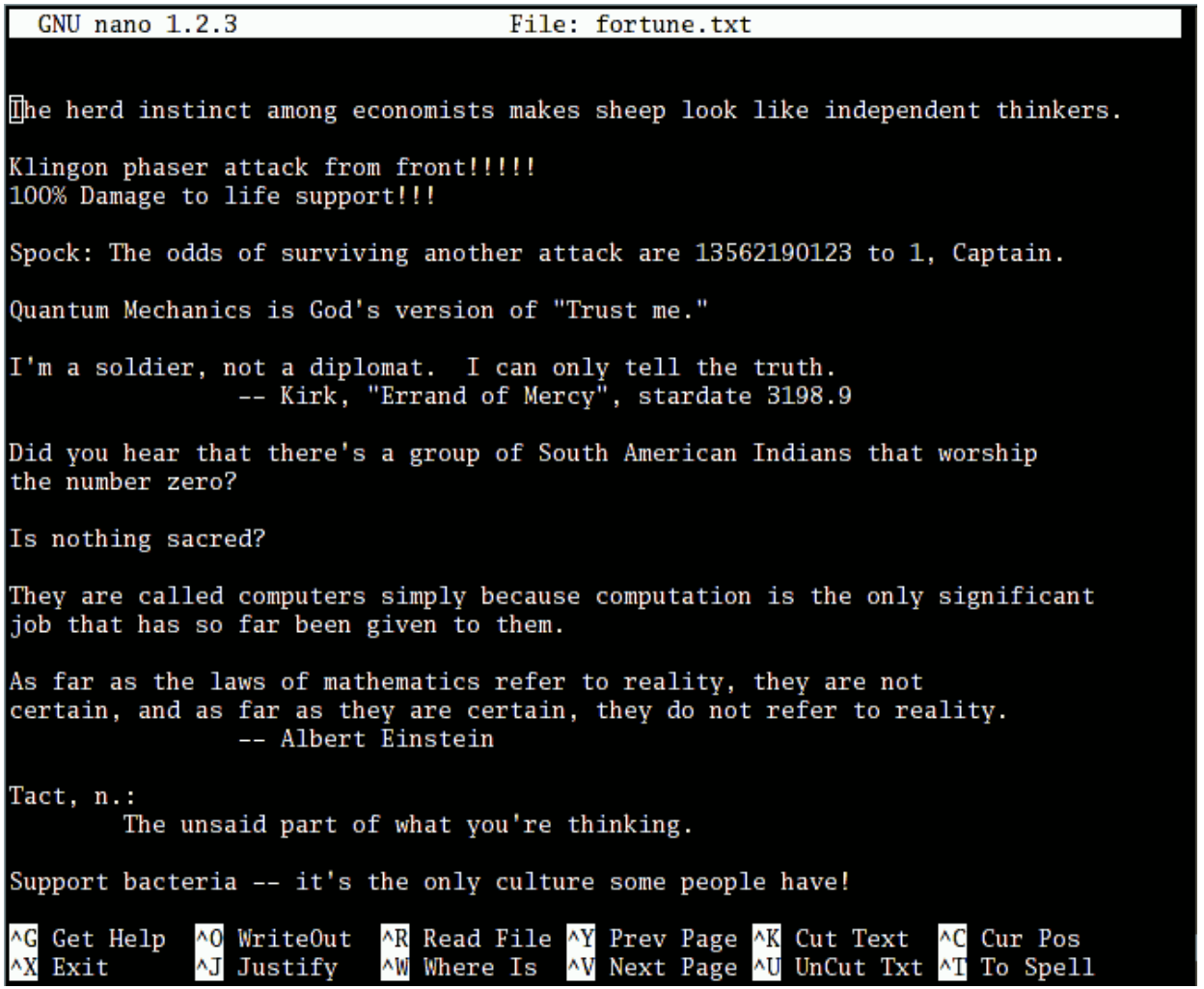
Récupérer votre fiche mémo sur vi si vous ne l'avez pas eue avec ce cours:
http://free-electrons.com/training/intro_unix_linux

GNU nano

http://www.nano-editor.org/

- Un autre éditeur de texte léger en mode texte
- Clone amélioré de Pico (éditeur non libre dans Pine)
- Convivial et plus facile à prendre en main grâce à un résumé des commandes affiché à l'écran.
- Disponible sous forme de paquetages binaires pour plusieurs plateformes.
- Une alternative à vi dans les systèmes embarqués. Cependant, pas encore disponible à travers busybox.

Capture d'écran de GNU nano



Divers

Compression et archivage

Mesure de la taille de fichiers

Attention : dépend de la taille du fichier!

- `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 sont 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 (semblable à celle de 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 en 1 seul.

- `tar`: à l'origine "tape archive" ("archive sur bande")
- Création d'une archive:
  `tar cvf <archive> <fichiers ou répertoires>`
  `c`: créer
  `v`: verbeux. Utile pour suivre la progression de l'archivage
  `f`: fichier. Archive créée dans un fichier (sinon utilise 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>`
  `t`: test
- Extraire tous les fichiers d'une archive:
  `tar xvf <archive>`
- Extraire seulement quelques fichiers d'une archive:
  `tar xvf <archive> <fichiers ou rép.>`
  Les fichiers ou répertoires sont donnés avec un chemin relatif au répertoire racine de l'archive.

Options supplémentaires dans GNU tar

`tar = gtar = GNU tar sous GNU / Linux`

Permet de compresser et décompresser des archives au vol.
Utile pour éviter de créer d'énormes fichiers intermédiaires.
Bien plus facile à faire qu'avec tar et bzip2!

- `j`: [dé]compresse au vol avec `bzip2`
- `z`: [dé]compresse au vol avec `gzip`
- Exemples (lequel retiendrez-vous?)
  - `gtar jcvf bugs_bill.tar.bz2 bugs_bill`
  - `tar cvf - bugs_bill | bzip2 > bugs_bill.tar.bz2`

La commande wget

A la place de télécharger des fichiers de votre navigateur, vous pouvez juste copier et coller leur url et les télécharger avec `wget`!

Principales caractéristiques de `wget`

- Supporte http et ftp
- Peut reprendre les téléchargements interrompus
- Peut télécharger des sites entiers ou au moins repérer les liens morts
- Très utile dans les scripts ou quand aucun graphique n'est disponible (administration de système, système embarqué)
- Supporte les proxy (variables d'environnement `http_proxy` et `ftp_proxy`)

Exemples wget

- `wget -c \ http://microsoft.com/customers/dogs/winxp4dogs.zip`
  Continue un téléchargement interrompu
- `wget -m http://lwn.net/`
  Fait un site miroir
- `wget -r -np http://www.xml.com/ldd/chapter/book/`
  Téléchargement récursif d'un livre en ligne pour les accès hors ligne.
  `-np`: "no-parent". Suit uniquement les liens dans le répertoire courant.

Vérifier l'intégrité des fichiers

Solution bon marché pour vérifier l'intégrité des fichiers

- `md5sum FC3-i386-disk*.iso > MD5SUM`
  Calcule un checksum MD5 (Message Digest Algorithm 5) 128 bit d'un fichier donné. Généralement redirigé vers un fichier.

- Exemple de sortie:
```
db8c7254beeb4f6b891d1ed3f689b412 FC3-i386-disc1.iso
2c11674cf429fe570445afd9d5ff564e FC3-i386-disc2.iso
f88f6ab5947ca41f3cf31db04487279b FC3-i386-disc3.iso
6331c00aa3e8c088cc365eeb7ef230ea FC3-i386-disc4.iso
```

- `md5sum -c MD5SUM`
  Vérifie l'intégrité des fichiers `MD5SUM` en comparant leur somme de contrôle MD5 actuelle avec celle d'origine.

Divers

Impression

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.

Commandes d'impression

- Variable d'environnement utile: `PRINTER`
  Définit l'imprimante par défaut sur le système. Exemple:
  `export PRINTER=lp`
- `lpr [-P<queue>] <fichiers>`
  Envoie les fichiers à la queue d'impression spécifiée. Les fichiers doivent être en format texte ou PostScript. Sinon, vous n'imprimerez que des déchets.
- `a2ps [-P<queue>] <fichiers>`
  "Any to PostScript" 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 [-P<queue>]`

  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      nsa_windows_backdoors.ps        60416 bytes
2nd     amoore  85      gw_bush_iraq_mistakes.ps        65024000 bytes
```

- `cancel <job#> [<queue>]`

  Retire la tâche spécifiée de la queue d'impression

Utilisation de fichiers PostScript et PDF

Visualisation d'un fichier PostScript

- Il existe des afficheurs PostScript, mais leur qualité est médiocre.
- Il vaut mieux passer en PDF avec `ps2pdf`:

```
ps2pdf algorithme_decss.ps
xpdf algorithme_decss.pdf &
```

Impression d'un fichier PDF

- Pas besoin d'ouvrir un afficheur de PDF!
- Il vaut mieux passer en PostScript avec `pdf2ps`:

```
pdf2ps rambaldi_pour_les_nuls.pdf
lpr rambaldi_pour_les_nuls.ps
```

Divers

Comparer des fichiers et des répertoires

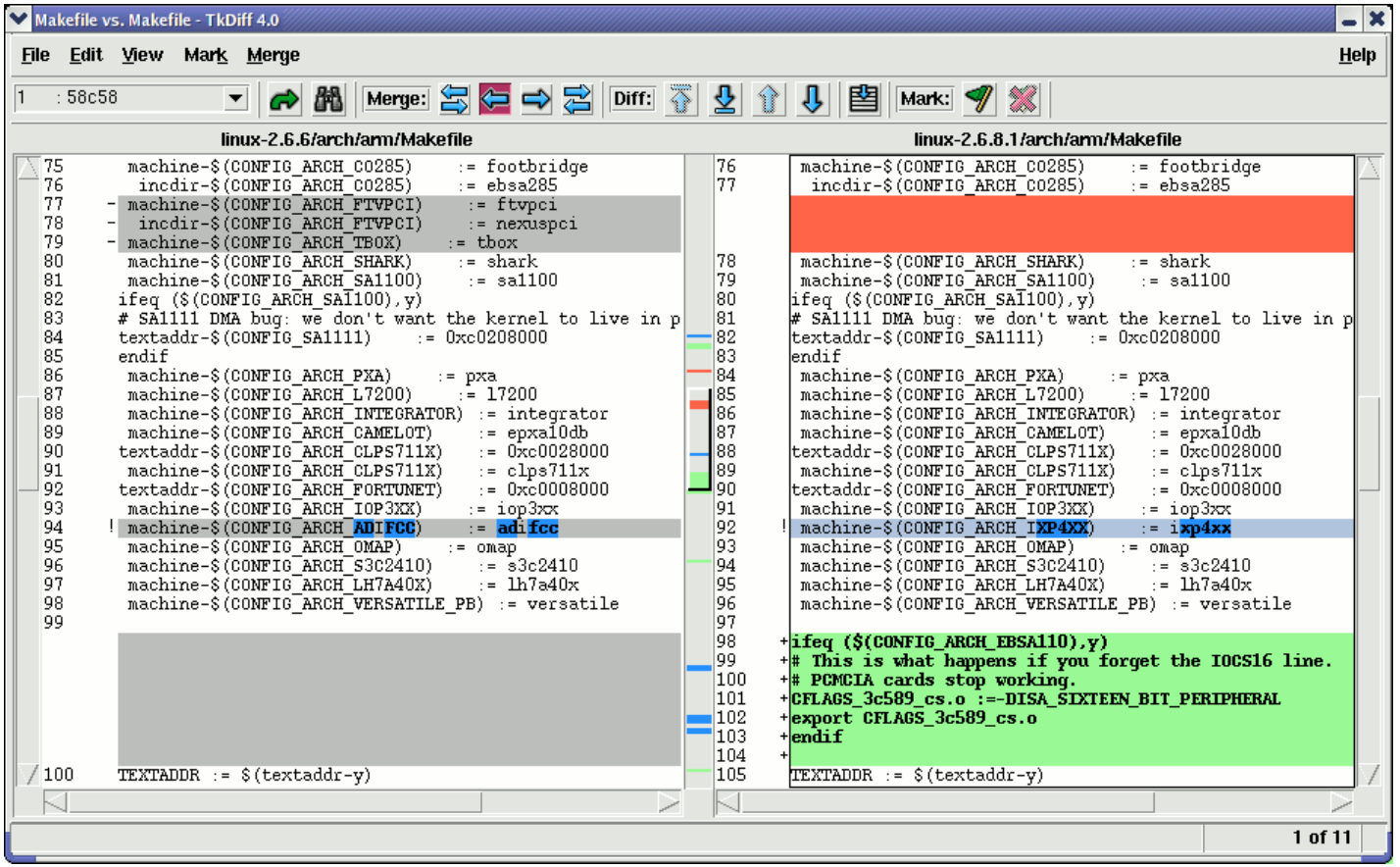
Comparaison de fichiers et répertoires

- `diff fichier1 fichier2`
  Affiche les différences entre 2 fichiers, ou rien si les fichiers sont identiques.
- `diff -r rep1/ rep2/`
  Affiche les différences entre fichiers de même nom dans les 2 répertoires.
- Pour examiner en détail les différences, mieux vaut utiliser des outils graphiques!

tkdiff

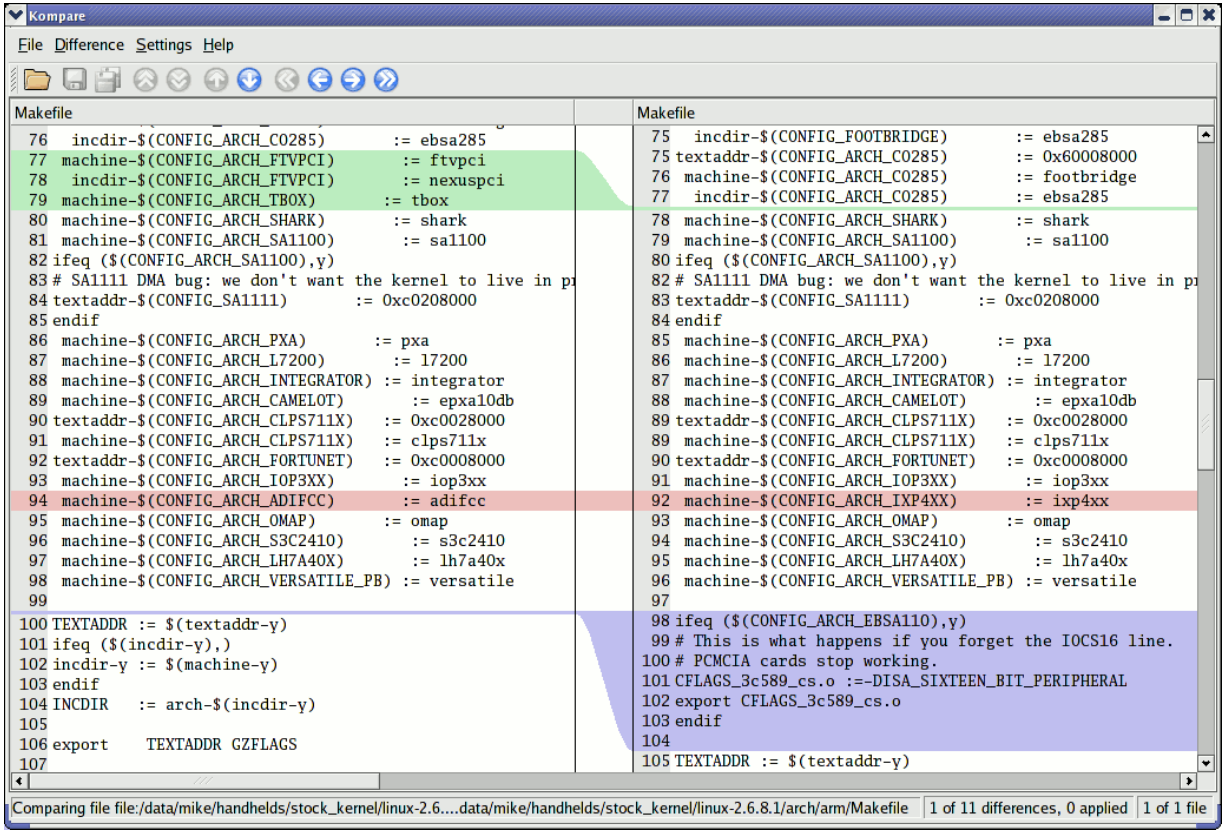
http://tkdiff.sourceforge.net/

Outil pratique pour comparer des fichiers et fusionner leurs différences



kompare

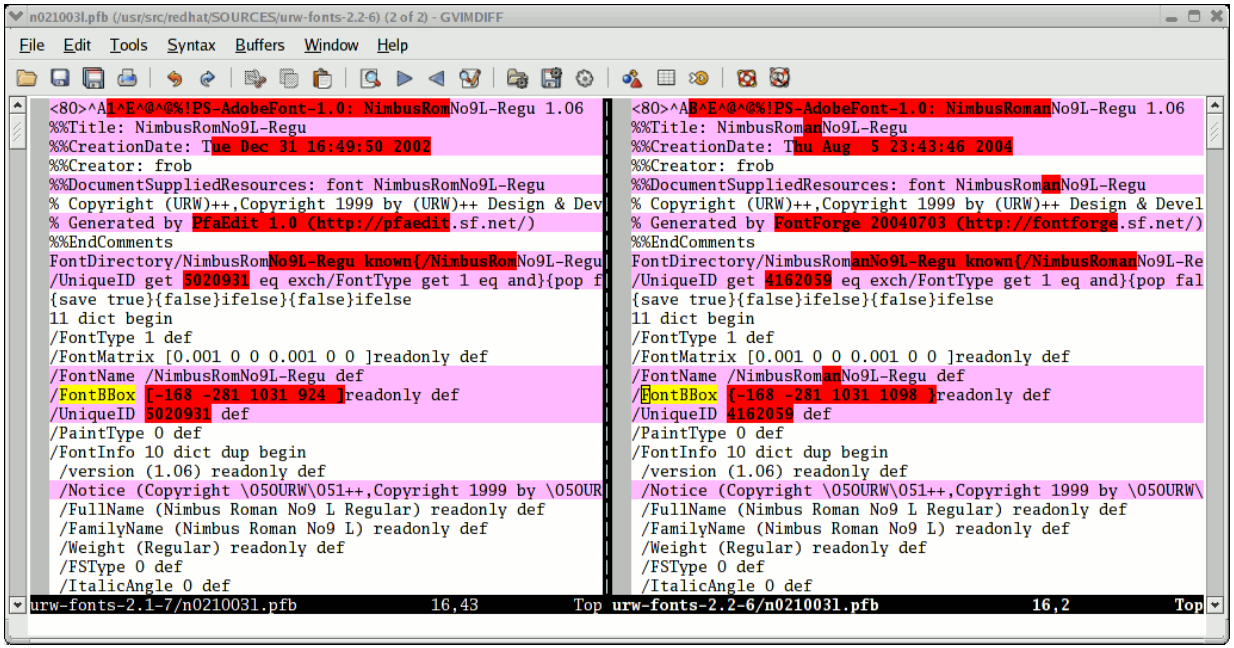
Un autre outil pratique pour comparer des fichiers et fusionner leurs différences. Fait partie du paquetage `kdesdk` (Fedora Core)



gvimdiff

Un autre outil pratique pour comparer les différences

Disponible dans la plupart des distributions avec `gvim`
Il n'utilise apparemment pas `diff`.
Pas de solutions avec les fichiers contenant des sections binaires!



Divers

Recherche de fichiers

La commande find

Plus facile à expliquer par quelques exemples!

- `find . -name "*.pdf"`
  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, une alternative à `find` beaucoup plus rapide.

- `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 bien plus rapide grâce à l'indexation de tous les fichiers dans une base de données dédiée, qui est mise à jour régulièrement.
- `find` est plus adapté aux recherches sur les fichiers créés récemment.

Divers

Commandes diverses

Informations sur les utilisateurs

- `who` (qui)
  Indique tous les utilisateurs connectés au système
- `whoami` (qui suis-je)
  Indique en tant que quel utilisateur je suis connecté
- `groups`
  Indique à quels groupes j'appartiens
- `groups <utilisateur>`
  Indique à quels groupes <utilisateur> appartient
- `finger <utilisateur>` (doigt)
  Fournit des détails (nom réel, etc) au sujet de <utilisateur>
  Désactivé sur certains systèmes (raisons de sécurité)

Changement d'utilisateurs

Vous n'avez pas besoin de vous déconnecter afin de vous connecter sur un autre compte utilisateur!

- `su hyde`
  (Rare) `hyde` devient le nouvel utilisateur, mais garde les paramètres de variables d'environnement de l'utilisateur courant.
- `su - jekyll`
  (Plus fréquent) `jekyll` devient le nouvel utilisateur, avec exactement les mêmes paramètres que l'utilisateur courant.
- `su -`
  Permet de devenir super-utilisateur.

Commandes diverses (1)

- `sleep 60` (dormir)
  Attend 60 secondes (ne consomme pas de ressources système)
- `wc report.txt` (“word count”: “compter les mots”)
  ```
  438  2115 18302 report.txt
  ```
  Compte le nombre de lignes, de mots et de caractères dans un fichier ou dans l’entrée standard.

Commandes diverses (2)

- `bc` ("basic calculator?")
  `bc` est une calculatrice maniable et complète. Elle inclut même un langage de programmation! Utiliser l'option `-l` pour faire du calcul avec virgule flottante.
- `date`
  Retourne la date courante. Utilisé dans les scripts pour indiquer quand la commande débute ou est terminé.

Bases d'administration système

Fichier propriétaire

- `chown -R sco /home/linux/src` (`-R`: recursive)
  Rend l'utilisateur `sco` le nouveau propriétaire des fichiers dans `/home/linux/src`
- `chgrp -R empire /home/askywalker`
  Rend le groupe `empire` le nouveau groupe de tout ce qui se trouve dans `/home/askywalker`
- `chown -R borg:aliens usss_entreprise/`
  `chown` peut être utilisé pour changer le propriétaire et le groupe en même temps.

Arrêter le système

- `shutdown -h +5` (`-h`: halt)
  Éteint le système dans 5 minutes.
  Les utilisateurs reçoivent un avertissement dans leur console.
- `shutdown -r now` (`-r`: redémarrer)
- `init 0`
  Un autre moyen d'arrêter le système (utilisé par `shutdown`)
- `init 6`
  Un autre moyen de redémarrer (utilisé par `shutdown`)
- `[Ctrl][Alt][Del]`
  Fonctionne aussi sur GNU/Linux (au moins sur PCs!)

Configuration réseau (1)

- `ifconfig -a`
  Affiche les informations sur toutes les interfaces réseau disponibles sur votre système.
- `ifconfig eth0`
  Liste les détails de l'interface eth0
- `ifconfig eth0 192.168.0.100`
  Assigne l'adresse IP `192.168.0.100` à `eth0`
  (1 adresse IP par interface)
- `ifconfig eth0 down`
  Eteint l'interface `eth0`
  (libère son adresse IP)

Configuration réseau (2)

- `route add default gw 192.168.0.1`
  Configure la route par défaut pour les paquets à destination de l'extérieur du réseau local. La passerelle (ici `192.168.0.1`) est responsable de l'envoyer à la prochaine passerelle, etc., jusqu'à la destination finale.
- `route`
  Listes les routes existantes
- `route del default`
  `route del <IP>`
  Supprime les routes données
  Utile pour redéfinir une nouvelle route.

Test réseau

- `ping freshmeat.net`
  `ping 192.168.1.1`
  Essaye d'envoyer des paquets à la machine donnée et retourne un paquet en accusé de réception.

```
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=0 ttl=150 time=2.51 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=150 time=3.16 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=150 time=2.71 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=150 time=2.67 ms
```

- Quand vous pouvez pinger votre passerelle, votre interface réseau fonctionne
- Quand vous pouvez pinger une adresse IP externe, vos paramètres réseau sont corrects!

Résumé de la configuration réseau

Uniquement pour les cas simples à interface unique, sans serveur dhcp...

- Connectez-vous au réseau (cable, carte wifi ou périphérique...)
- Identifier votre interface réseau:
  `ifconfig -a`
- Assigner une adresse IP à votre interface (supposons `eth0`)
  `ifconfig eth0 192.168.0.100` (exemple)
- Ajouter une route à votre passerelle (supposons `192.168.0.1`) pour les paquet sortant du réseau:
  `route add default gw 192.168.0.1`

Résolution de noms

- Votre programme a besoin de savoir quelle adresse IP correspond à un nom de domaine donné (comme `kernel.org`)
- Un Domain Name Server (DNS) s'occupe de cette procédure.
- Vous devez juste spécifier l'adresse IP d'un ou plusieurs serveurs DNS dans votre fichier `/etc/resolv.conf` :

```
nameserver 217.19.192.132
nameserver 212.27.32.177
```

- Les changements prennent effet immédiatement!

Création d'un système de fichier

Exemples

- `mkfs.ext2 /dev/sda1`
  Formate votre clé USB (`/dev/sda1`: données brutes de la 1ère partition) au format ext2
- `mkfs.ext2 -F disk.img`
  Formate un fichier image disque au format ext2
- `mkfs.vfat -v -F 32 /dev/sda1` (`-v`: verbose)
  Formate votre clé USB au format FAT32
- `mkfs.vfat -v -F 32 disk.img`
  Formate un fichier image disque au format FAT32

Une image disque vide peut être créée comme dans l'exemple ci-dessous:

```
dd if=/dev/zero of=disk.img bs=1024 count=65536
```


Monter un périphérique (1)

- Pour rendre un système de fichier sur n'importe quel périphérique (interne ou externe) visible sur votre système, vous devez le *monter* .
- La première fois, créer un point de montage sur votre système: `mkdir /mnt/usbdisk` (exemple)
- Maintenant, monter-le:
  `mount -t vfat /dev/sda1 /mnt/usbdisk`
  `/dev/sda1`: périphérique physique
  `-t:` spécifie le type de système de fichier (format)
  (`ext2`, `ext3`, `vfat`, `reiserfs`, `iso9660`...)

Monter un périphérique (2)

- La plupart des options sont disponibles, en particulier choisir les permissions, le propriétaire ou le groupe... Voir les pages du manuel de `mount` pour des détails.
- Les options de montage pour chaque périphérique peuvent être sauvegardées dans le fichier `/etc/fstab`.
- Vous pouvez aussi monter une image disque stockée dans un fichier ordinaire (*périphériques loopback*)
  - Utile pour accéder au contenu d'un CD-ROM ISO sans avoir à le graver
  - Utile pour créer une partition Linux sur un disque dur ne contenant que des partitions Windows

```
cp /dev/sda1 usbkey.img
mount -o loop -t vfat usbkey.img /mnt/usbdisk
```

Lister les systèmes de fichiers montées

- Utiliser simplement la commande `mount` sans argument:

```
/dev/hda6 on / type ext3 (rw,noatime)
none on /proc type proc (rw,noatime)
none on /sys type sysfs (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/hda4 on /data type ext3 (rw,noatime)
none on /dev/shm type tmpfs (rw)
/dev/hda1 on /win type vfat (rw,uid=501,gid=501)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
```

- Ou afficher le fichier `/etc/mtab`
  (résultat identique, mise à jour par les commandes `mount` et `umount` chaque fois qu'elles sont lancées)

Démonter un périphérique

- `umount /mnt/usbdisk`
  Finit toutes les écritures en cours et démonte le périphérique spécifié, qui peuvent être enlevé de manière sécurisée.
- Pour être capable de démonter un périphérique, vous devez fermer tous les fichiers ouverts dessus:
  - Fermer toutes les applications utilisant la partition montée
  - Vérifier qu'aucun de vos shells n'ont de répertoire de travail étant sur le périphérique monté.
  - Vous pouvez utiliser la commande `lsof` (list open files : liste des fichiers ouverts) pour voir quels processus utilisent encore des fichiers sur la partition montée.

GNU / Linux: paquetages des distributions

Comment trouver des paquetages

- Paquetages Debian : http://www.debian.org/distrib/packages
  Recherche par nom de paquetage ou de fichier
- rpmfind: http://rpmfind.net/
  Nombreux paquetages RPM pour Red Hat, Mandriva, Suse...

Identifier les paquetages

A quel paquetage appartient un fichier?

- Utile pour récupérer la plupart des informations, récupérer le code, trouver des versions récentes, rapports de problèmes...
- Distributions avec des paquetages RPM:
  (Red Hat, Fedora, Mandriva, Suse...)

```
> rpm -qf /bin/ls
coreutils-5.2.1-7
```

- Debian:

```
> dpkg -S /bin/ls
fileutils: /bin/ls
```

Information sur les packages

- Accéder à la description des paquetages, numéros de versions, sources, etc.
- Distributions basées sur RPM:

```
rpm -qi <package-name>
```

- Debian:

```
dpkg -s <package-name>
```

Pour aller plus loin

Aide sur les commandes

Certaines commandes Unix et la plupart des commandes de GNU / 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 forums et des archives de listes de discussions sont publics, et sont indexés très régulièrement par Google.
- Si vous cherchez la cause d'un message d'erreur, copiez-le tel-quel dans le formulaire de recherche, entre des guillemets (“message d'erreur”). Il est très probable que quelqu'un d'autre ait déjà rencontré le même problème.
- Pensez bien à utiliser Google Groups: http://groups.google.com/ Ce site indexe plus de 20 ans de groupes de discussion.

Recherche de ressources sur Internet (2)

Recherche de documentation

- Recherchez `<outil>` ou `<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` ou `<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.

Pour aller plus loin

Utilisation de GNU / Linux à la maison

Quelques applications de bureau

Faire une démonstration sur un écran avec un projecteur!

- Mozilla: navigateur Internet, client de courrier électronique et éditeur HTML
- Firefox: navigateur léger dérivé de Mozilla
- OpenOffice: suite bureautique complète compatible avec MS Office: traitement de texte, tableur, présentations, graphiques...
- Le GIMP: un éditeur graphique extrêmement puissant
- Gqview: afficheur de galerie de photos
- Evolution: client de messagerie et calendrier semblable à Outlook.

Alternatives aux outils sous Windows

Internet Explorer	Mozilla Firefox
IIS	Apache
Money	GNU Cash
MS Office	OpenOffice
MS Outlook	Evolution
MS Project	Mr Project (Planner)
Nero	k3b
Photoshop	The GIMP
WinAmp	xmms
W. Media Player	xine mplayer

Plus d' alternatives:
http://linux.ie/newusers/alternatives.php

GNU / Linux à la maison (1)

A la maison, GNU / Linux est aussi une alternative sérieuse à Windows

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.
- 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 autres membres de la famille ne peuvent ni toucher au système ni aux fichiers de quelqu'un d'autre. Ils ne peuvent endommager que leurs propres fichiers.
- Décourage les pirates
  Même si vous êtes connecté en permanence à Internet, votre système attire moins les pirates.

GNU / Linux à la maison (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.
- Les développeurs peuvent facilement être contacter pour leur suggérer de nouvelles fonctionnalités.

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!)
- Vous êtes libres d'aider votre entourage en partageant vos programmes avec lui.
- Vous êtes libres d'utiliser les mêmes programmes au travail également!

GNU / Linux à la maison (3)

Vous pouvez passer à GNU / 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)
- Mieux connaître les ordinateurs et la programmation

Si vous possédez encore une copie de Windows, vous pouvez la garder (option au démarrage) pour:

- Les jeux. Nombre d'entre eux 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 GNU / Linux

Utiliser les distributions GNU / Linux

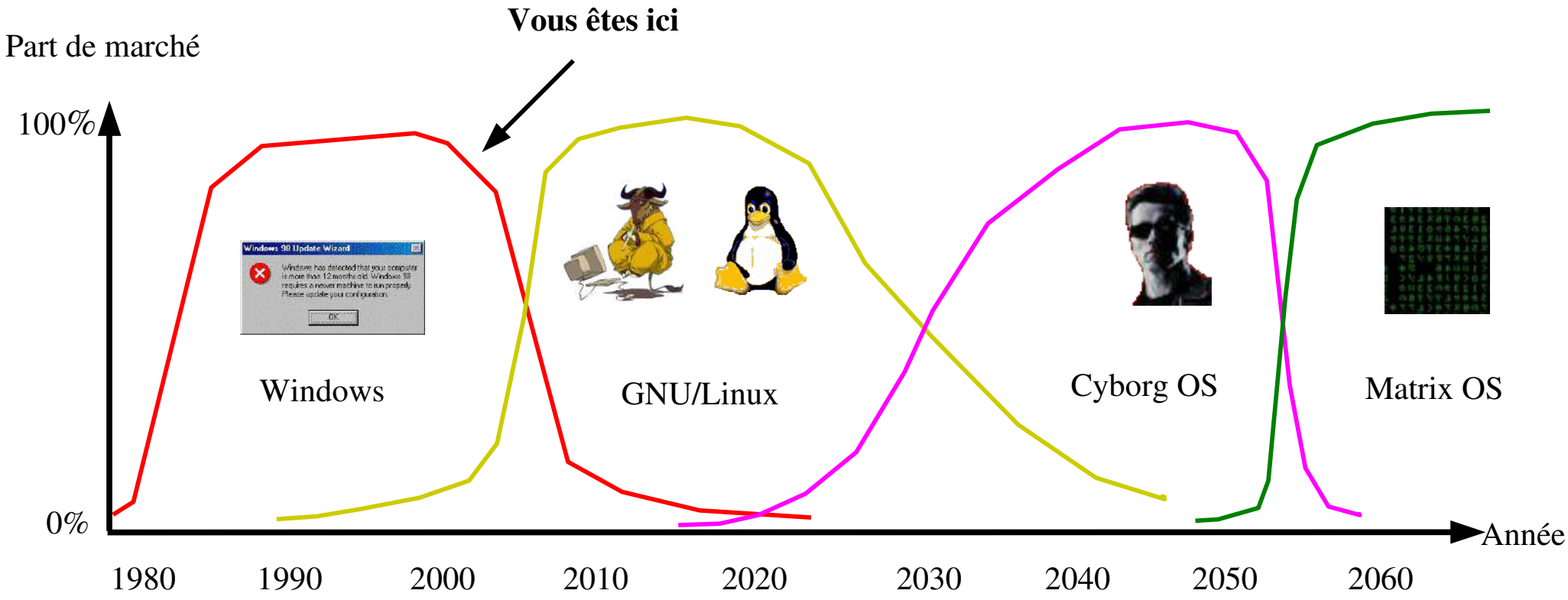
Utilisation de distributions GNU / Linux

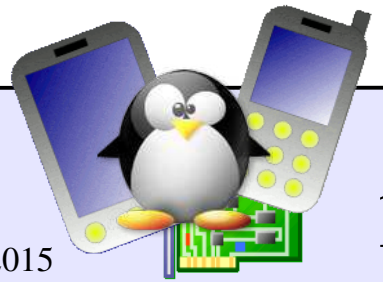
- Vous permettent d'installer GNU / Linux dans un emplacement libre sur votre disque dur, tout en gardant Windows ("double démarrage")
- Ont une interface très conviviale qui peut détecter automatiquement la plupart des matériels. Vous n'avez aucun pilote à installer!
- Vous permettent de choisir les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

Conclusion

Ne ratez pas le prochain train!

Histoire des systèmes d'exploitation





Travaux pratiques

Des travaux pratiques sont aussi disponibles au même endroit:
http://free-electrons.com/docs/command-line/

Ils sont un complément utile pour consolider ce que vous avez appris lors de cette formation. Ils ne vous indiquent pas *comment* faire les exercices. Cependant, ils ne font appel qu'à des notions et des outils présentés pendant le cours.

S'il vous arrive d'être bloqué pendant un exercice, cela prouve que quelque chose vous a échappé dans le cours, et que vous devez revenir aux présentations pour trouver ce que vous cherchez.

Remerciements

- Au projet OpenOffice.org pour ses outils de présentation et de traitement de texte qui ont satisfait toutes mes attentes.
- A la communauté Handhelds.org qui m'a apporté beaucoup d'aide et m'a donné tant d'occasions de me rendre utile.
- Aux membres de la communauté du Logiciel Libre, pour avoir partagé le meilleur d'eux-mêmes: leur travail, leur connaissance, leur amitié.
- Aux personnes ayant envoyé des commentaires ou des corrections : Laurent Thomas, Jeff Ghislain, Leif Thande, Frédéric Desmoulins, Przemysław Ciesielski