Systèmes informatiques

Franck Guingne,
sur la base du cours d'Olivier Lecarme

Cours Licence 1; Semestre 2

2009–2010

Deuxième cours : Fichiers et répertoires

Plan en cours

Ce qu'est un répertoire

- le *système de fichiers* est organisé en une unique arborescence
- les répertoires sont les nœuds de l'arborescence
- la racine est unique (contrairement à Windows) et notée `/`
- les périphériques sont cachés (contrairement à Windows)
- les fichiers sont cités dans les répertoires
- la commande `mkdir` sert à créer un répertoire

Répertoire courant

- tout processus travaille dans son *répertoire courant*
- concept particulièrement clair avec le shell :
  - la commande `pwd` (*print working directory*) affiche le *nom absolu* du répertoire courant : chemin complet depuis la racine
  - exemple :

    ```
    guingne@deptinfo ~/www> pwd
    /u/deptinfo/guingne/www
    ```

- existe également avec Emacs :

  ```
  M-x pwd RET
  Directory /u/profs/guingne/
  ```

Accès à un fichier

- un paramètre de commande qui est un fichier prend l'une de trois formes :
  - *nom absolu* : `/usr/dict/words`
  - *nom local* : `words` si le répertoire courant est `/usr/dict`
  - *nom relatif* : `../dict/words` si le répertoire courant est `/usr/local`
- deux notations spécifiques :
  - `..` représente le répertoire père
  - `.` représente le répertoire courant

Donner un nom dans Emacs

- une commande à paramètre de type fichier (par exemple `C-x C-f`) :
  - affiche dans le *mini-tampon* le nom absolu du répertoire courant
  - on peut taper à la suite si le fichier est dans ce répertoire
  - on peut descendre dans la hiérarchie
  - on peut effacer des noms de répertoires (`M-DEL`) pour remonter dans la hiérarchie
  - on peut ne rien effacer et taper un nom absolu à la suite, c'est-à-dire un nom commençant par `/` ou par `~`

Compléter un nom

- aussi bien dans le shell `zsh` qu'avec Emacs :
  - la touche `TAB` permet de compléter le nom dont on a tapé le début
  - si plusieurs noms commencent avec ce même début (ambiguïté), on obtient la liste des noms possibles
  - avec Emacs, on peut sélectionner le nom voulu (bouton 2 de la souris)

Changer de répertoire

- commande `cd` (*change directory*)
- `cd ..` remonte d'un niveau dans la hiérarchie
- `cd /` fait de la racine le répertoire courant
- `cd SI` va dans le répertoire `SI` du répertoire courant (s'il existe)
- `cd -` va au précédent répertoire courant
- `cd ../..` va au répertoire grand-père (s'il existe)
- `cd` va au répertoire personnel de l'utilisateur
- la commande existe dans Emacs mais est d'ordinaire inutile

Protections

- on ne peut pas aller dans n'importe quel répertoire :
  - il doit exister
  - on doit en avoir la permission
- le propriétaire d'un répertoire détermine les *permissions* (et interdictions) qui le concernent
- le même mécanisme s'applique aussi aux fichiers ordinaires

Le système de permissions

- trois types de permissions :
  - lecture, notée `r` (*read*)
  - écriture, notée `w` (*write*)
  - exécution, notée `x` (*eXecute*)
- pour un répertoire :
  - la lecture permet de consulter la liste des fichiers sur lesquels pointe le répertoire
  - l'écriture permet d'ajouter des références de fichiers au répertoire ou d'en enlever
  - l'exécution permet d'accéder aux fichiers répertoriés

Les types d'utilisateurs

- les permissions s'adressent à trois groupes d'utilisateurs :
  - le propriétaire du fichier ou du répertoire, noté `u` (*user*)
  - le groupe des utilisateurs proches du propriétaire, noté `g` (*group*)
  - l'ensemble des autres utilisateurs, noté `o` (*others*)
- l'ensemble des neuf permissions peut donc s'exprimer par une chaîne de caractères
- exemple :
  - `rwxr-x---`
  - si c'est un répertoire, le groupe peut le parcourir et passer aux fichiers répertoriés, les autres n'ont aucun droit

Masque de permissions

- le *masque de permissions* est utilisé à chaque création de fichier
- il n'a aucune influence sur les fichiers déjà créés
- il est formé de trois chiffres octaux, un par type d'utilisateurs
- c'est un masque, donc un bit valant 1 demande de mettre l'indicateur correspondant hors fonction
- exemples :
  - le répertoire de permissions `rwxr-x---` a pu être créé avec un masque courant valant `027`
  - un masque de `022` autorise lecture et exécution à tous, écriture uniquement au propriétaire
  - un masque de `077` interdit l'accès à tout autre que le propriétaire

Commande de définition du masque

- la commande `umask` fixe la valeur du masque
- on l'écrira sous forme octale seulement
- exemples :
  - `umask 022`
  - `umask 077`
- Attention : cela ne change rien aux permissions des fichiers existants
- remarque : la permission `x` n'est mise qu'aux répertoires créés, pas aux autres fichiers

Commande de changement de permissions

- la commande `chmod` change les valeurs des permissions d'un fichier
- elle n'est possible qu'avec la permission d'exécution sur le répertoire contenant le fichier
- deux arguments :
  - chaîne symbolique expliquant les modifications
  - liste des fichiers concernés
- comme dans toute commande, les arguments sont séparés par un ou plusieurs blancs

Notation symbolique

- suite d'indications séparées par des virgules (sans blancs)
- une *indication* comprend trois parties :
  - une ou plusieurs lettres indiquant le type d'utilisateur (`u`, `g` ou `o`), ou `a` (*all*) signifiant `ugo`
  - un caractère indiquant l'opération :
    - `+` pour ajouter une permission
    - `=` pour établir une permission
    - `-` pour retirer une permission
  - une ou plusieurs lettres indiquant le type de permission (`r`, `w` ou `x`)
- éviter la notation octale, contradictoire avec celle de `umask`

Exemples

- j'ai créé mon répertoire `SI/TP02` avec mon masque à `077`
- il a donc comme permissions `u=rwx`
- je veux le rendre accessible en lecture et exécution par tous :

  ```
  chmod go+rx SI/TP02
  ```

- autre manière :

  ```
  chmod u=rwx,go=rx SI/TP02
  ```

- je veux enlever la permission de lectures aux utilisateurs ne faisant pas partie de mon groupe :

  ```
  chmod o-r SI/TP02
  ```

Énumérer les fichiers d'un répertoire

- la commande `ls` énumère les fichiers d'un répertoire
- commande complexe, ayant beaucoup d'options
- cas le plus simple : énumération en ordre alphabétique, sur plusieurs colonnes, sans les fichiers dont le nom commence par un point
- cas fréquent : avec l'option `-l`, affiche des informations détaillées sur chaque fichier :
  - permissions
  - identité du propriétaire
  - groupe
  - taille
  - date et heure de dernière modification
  - nom
- sans argument, porte sur le répertoire courant
- pour les arguments qui sont des fichiers ordinaires, affiche des détails sur eux

Supprimer un répertoire

- la commande `rmdir` supprime un répertoire
- le répertoire doit être vide
- il faut être propriétaire du répertoire à effacer
- il faut avoir la permission d'écriture sur le répertoire parent

Plan en cours

Créer un fichier

- il faut avoir la permission d'écriture dans le répertoire qui désignera le fichier
- il existe de nombreuses manières de le faire :
  - la commande `touch` :
    - fixe la date de dernière modification du fichier au moment présent
    - si le fichier n'existe pas, la commande crée un fichier vide
  - avec Emacs, éditer un fichier inexistant (`C-x C-f`) puis le sauvegarder (`C-x C-s`) le crée
  - la commande `cp` copie un fichier existant vers un autre
  - la commande `mv` déplace un fichier existant ou le renomme
  - attention : pour créer un répertoire, il faut utiliser `mkdir`

Copier un fichier

- la commande `cp` (*CoPy*) sous sa forme la plus simple a deux arguments :
  - le fichier à copier, qui doit exister et pouvoir être lu
  - le fichier résultat, qui doit être placé dans un répertoire où on a le droit d'écriture
- si le fichier résultat existait déjà, il est remplacé
- si on met un nom de répertoire comme résultat, le fichier résultat a le même nom que le fichier à copier
- c'est une des utilisations du nom `.`
- on peut copier de cette manière plus d'un fichier :
  - la commande a n arguments
  - les $n-1$ premiers sont les noms des fichiers à copier
  - le dernier est le nom du répertoire où copier
  - les noms sont systématiquement les mêmes

Référence à plusieurs fichiers

- beaucoup de commandes acceptent une liste de fichiers
- les noms des fichiers sont séparés par des espaces, et constituent donc autant d'arguments
- le shell fournit une notation pour citer plusieurs fichiers à la fois :
  - le *joker* `*` représente une chaîne quelconque, éventuellement vide
  - le joker `?` représente un caractère quelconque
  - les caractères `.` (en début de nom de fichier) et `/` ne peuvent pas être reconnus par un joker
- la notation `~nom d'utilisateur` désigne le répertoire personnel de l'utilisateur cité
- la notation `~` représente le répertoire personnel de l'utilisateur courant

Exemples d'utilisation de jokers

- `*` représente tous les fichiers du répertoire courant, sauf ceux dont le nom commence par un point
- `toto??.text` représente tous les fichiers du répertoire courant dont le nom commence par `toto`, suivi de deux caractères quelconques et se termine par `.text`
- `~guingne/.*` représente tous les fichiers du répertoire `/u/profs/guingne` dont le nom commence par un point
- `/u/profs/*/.*` représente tous les fichiers dont le nom commence par un point dans tous les répertoires personnels des professeurs

Exemples de copies

- `cp /var/log/dmesg ~/CoursSI/mess` :
  - crée dans le répertoire `/u/deptinfo/guingne/CoursSI` un fichier `mess`, copie du fichier `/var/log/dmesg`
  - si ce fichier existait déjà, il est remplacé
- `cd /u/deptinfo/guingne/CoursSI ; cp mess messages` :
  - deux commandes sur la même ligne, séparées par le point-virgule
  - le fichier `messages` est une copie du fichier `mess`, dans le même répertoire
- `cp /var/log/dmesg .` :
  - le fichier `dmesg` du répertoire courant est une copie du fichier `/var/log/dmesg`

Déplacer un fichier

- la commande `mv` (*MoVe*) sous sa forme la plus simple a deux arguments :
  - le fichier à déplacer, qui doit exister et pouvoir être lu et supprimé
  - le fichier résultat, qui doit être placé dans un répertoire où on a le droit d'écriture
- si le fichier résultat existe déjà, il est remplacé
- si les deux fichiers sont dans le même répertoire, cela revient à en changer le nom (renommer)
- comme pour `cp`, il peut y avoir n arguments, le dernier étant un répertoire

Liens

- il existe deux sortes de *liens* dans Unix :
  - un *lien dur* (ou lien normal) est un pointeur sur un fichier
  - un *lien symbolique* est un fichier spécial, dont le contenu est la chaîne de caractères désignant le fichier

Lien dur

- créer un fichier, c'est d'abord placer un lien dur dans le répertoire correspondant
- plusieurs liens peuvent pointer sur le même fichier
- un fichier n'est donc pas dans un répertoire
- la commande `ln` (*LiNk*) crée un lien sur un fichier :
  - `ln initial pointeur` crée un lien nommé `pointeur` sur le fichier nommé `initial`
  - les deux noms peuvent être absolus, relatifs ou locaux
  - il faut pouvoir écrire dans le répertoire où est placé le lien
  - comme pour les commandes `cp` et `mv`, on peut avoir n arguments, le dernier étant un répertoire

Lien symbolique

- un lien dur ne peut pointer que sur un fichier existant
- de plus, un lien dur étant un pointeur, c'est-à-dire la référence à un bloc de données dans un espace physique donné, il ne peut pas permettre une référence entre *volumes* différents
- un *lien symbolique* est donc une référence indirecte :
  - il peut désigner un fichier situé sur un autre volume
  - il est clairement distinct d'un lien dur
  - mais il peut désigner un fichier qui n'existe pas
- on le crée simplement avec la commande
  `ln -s initial pointeur`
  avec les mêmes règles que pour la forme sans option
- la forme avec un seul argument crée un lien symbolique de même nom dans le répertoire courant

Supprimer un fichier

- la commande `rm` sert à supprimer les fichiers mentionnés en argument
- en fait, la commande supprime non pas un fichier mais le lien dur mentionné en argument
- un fichier n'est supprimé que quand plus aucun lien dur ne le désigne
- effacer un lien symbolique n'efface que le lien lui-même, et n'a aucune influence sur le fichier désigné
- la commande `rm` est dangereuse
  - si un fichier est supprimé, il n'est pas récupérable
  - nous forcerons la commande à être interactive : il faudra approuver explicitement chaque suppression
  - Emacs demande systématiquement confirmation

Plan en cours

Fonctionnement du mode

- mode spécifique donnant accès à toutes les manipulations de répertoires et fichiers
- entrée par plusieurs commandes :
  - `C-x d` pour ouverture dans la fenêtre courante
  - `C-x 4 d` pour ouverture dans une autre fenêtre du même cadre
  - `C-x 5 d` pour ouverture dans un nouveau cadre
  - `C-x C-j` pour remplacer le tampon courant par le répertoire courant (nécessite le package dired-x).
- la fenêtre affiche le résultat de la commande `ls -l`

Principes d'utilisation

- le tampon n'est normalement pas directement modifiable
- les commandes utilisent surtout les caractères ordinaires
- toutes les commandes à effet irréversible sont effectuées en deux étapes : marquer puis appliquer
- certaines commandes sans danger ont un effet immédiat
- les commandes s'appliquent :
  - au fichier où est le curseur s'il n'y a aucune marque
  - à l'ensemble des fichiers marqués
  - au fichier où est le curseur et aux $n-1$ suivants (resp. précédents) si la commande est précédée du nombre n (resp. $-n$)
- on peut déplacer le curseur comme d'ordinaire, mais aussi avec les commandes des outils de consultation (`SPC` ou `n`, `DEL` ou `p`)

Ce qu'on voit dans la fenêtre

- le répertoire est complet, y compris les fichiers commençant par un point
- les répertoires sont repérés par un `d` en tête
- les commandes `<` et `>` reculent ou avancent d'un répertoire à l'autre
- la commande `i` inclut dans le tampon en cours la liste du répertoire où est le curseur
- les touches `^` et `i` permettent de monter et descendre dans la hiérarchie
- les fichiers de sauvegarde d'Emacs sont entre deux `#` : sauvegarde automatique après n frappes de caractères
- les fichiers d'archive se terminent par un `~` : sauvegarde automatique de l'état précédent d'un fichier

Actions simples

- le répertoire est normalement trié en ordre alphabétique
- la commande `s` bascule entre cet ordre et l'ordre de date décroissante de dernière modification
- la commande `$` masque (ou démasque) le répertoire où se trouve le curseur
- la commande `g` reconstitue l'affichage du tampon pour tenir compte de modifications faites extérieurement à ce tampon
- commandes principales de marquage (marquent et passent à la ligne suivante) :
  - `d` marque le fichier pour effacement
  - `m` pose une marque à tout faire
  - `u` efface la marque
  - `DEL` passe à la ligne précédente et efface la marque

Actions différées

- la commande `x` effectue les effacements demandés, avec nécessité de réponse `yes RET` à la question posée
- les commandes suivantes portent sur les fichiers marqués, en donnent la liste et demandent confirmation :
  - `C` pour copier (`cp`)
  - `R` pour renommer (`mv`)
  - `D` pour effacer (`rm` ou `rmdir`)
  - `H` pour mettre un lien dur (`ln`)
  - `S` pour mettre un lien symbolique absolu (`ln -s`)
  - `Y` pour metre un lien symbolique relatif
  - `M` pour changer les permissions (`chmod`)
  - etc.
- si le cadre courant contient deux fenêtres en mode Dired, la destination des opérations `C R H S` est par défaut l'autre fenêtre

Autres opérations

- certaines opérations ne concernent que le fichier courant :
  - `e`, `f` ou `RET` ouvrent le fichier dans un tampon qui remplace celui du répertoire
  - `v` l'examine (modifications impossibles, défilement par `SPC` et `DEL`)
  - `o` ouvre le fichier dans l'autre fenêtre
  - des commandes commençant par `%` marquent ou manipulent les fichiers correspondant à certains critères
  - la commande `?` résume les commandes les plus utiles
  - la commande `h` fournit une aide plus détaillée
  - la commande `!` applique une commande de Unix aux fichiers marqués :
    - Emacs suggère une ou plusieurs commandes
    - on peut passer de l'une à l'autre par les commandes `M-p` et `M-n` ou ↑ et ↓

Encore d'autres commandes

- la commande `#` marque pour effacement les fichiers de sauvegarde
- la commande `~` marque pour effacement les fichiers d'archive
- la commande `+` crée un répertoire (`mkdir`)
- la commande `=` compare deux fichiers (`diff`)
- la commande `A` recherche une sous-chaîne dans tous les fichiers marqués ; `M-,` passe successivement sur toutes les apparitions
- la commande `Q` effectue de plus un remplacement à la demande (`SPC` accepte, `DEL` refuse)
- la commande `t` inverse les marques
- et bien d'autres...

Plan en cours

Examiner le contenu d'un fichier

- on veut voir le contenu sans le modifier (en particulier parce qu'on n'en a pas la permission)
- dans Emacs :
  - commande `v` du mode Dired
  - commandes `C-x C-r`, `C-x 4 r` ou `C-x 5 r`
- dans un shell :
  - commande `more`, toujours disponible
  - commande `less`, plus perfectionnée

Comportement général

- dans tous les cas :
  - `SPC` avance d'une page
  - `d` avance d'une demie page
  - `RET` avance d'une ligne
  - `q` sort du mode ou de la commande
  - `/ chaîne` avance jusqu'à la première apparition de la chaîne indiquée
- avec `less` ou le mode View d'Emacs :
  - `DEL` recule d'une page
  - `u` recule d'une demie page
  - `y` recule d'une ligne
  - `<` va en début de fichier
  - `>` va en fin de fichier
  - `?` recule comme `/` avance

Recherche de chaîne dans Emacs

- recherche directe :
  - on tape la chaîne et Emacs déplace le curseur jusqu'à la première apparition
  - pas de commande abrégée, possibilité peu intéressante
- recherche incrémentale :
  - le curseur se déplace au fur et à mesure qu'on tape la chaîne
  - on peut retirer des caractères de la chaîne (`DEL`), ce qui fait remonter dans l'historique
  - répéter la commande de recherche va à la prochaine apparition
  - `C-s` cherche vers la fin
  - `C-r` cherche vers le début
  - `RET` termine la recherche et laisse le curseur en place
  - `C-g` termine la recherche et revient au point de départ

Recherche par expression régulière

- utilisation :
  - fonctionne de manière directe ou incrémentale
  - mécanisme puissant, décrit pour l'instant de manière incomplète
  - utilisable aussi avec les commandes `A` et `Q` du mode Dired
  - utilisable aussi avec la commande `/` (et `?`) de `more` (et `less`)
- principes :
  - le caractère `.` décrit n'importe quel caractère : `t.t.`
  - l'opérateur `*` décrit 0 à n apparitions du caractère précédent : `to*to`
  - le caractère `\` change l'interprétation du caractère qui suit : `t\.t\*`
  - `\(` et `\)` servent de méta-parenthèses : `\(to\)*`
  - etc.
- attention ! Ne pas confondre *jokers* et *expressions régulières*

Imprimer un fichier

- imprimer un fichier sans vraie transformation
- deux familles de commandes aboutissant au même résultat :
  - `lpr` ou `lp` (*line printer*) pour ajouter un fichier à la file d'attente d'une imprimante
  - `lprm` ou `cancel` pour retirer un fichier de la file d'attente
  - `lpq` ou `lpstat` pour examiner la file d'attente
- les options sont équivalentes
- la mise en file d'attente implique en général une transformation cachée, adaptée au modèle d'imprimante
- la forme standard passe par le codage en Postscript, langage des imprimantes modernes
- la commande `a2ps` (*ASCII to Postscript*) effectue des transformations plus élaborées, en bonne partie automatisées

Imprimer depuis Emacs

- commande `P` en mode Dired (appelle `lpr`)
- la commande `!` en mode Dired permet d'appeler `a2ps`
- le menu `File` comprend une série d'entrées pour l'impression :
  - `Print Buffer` donne un résultat le plus simple possible
  - `Postscript Print Buffer` donne une image plus exacte du contenu du tampon (en couleur)
  - `Postscript Print Buffer (B+W)` fait la même chose en noir et blanc
  - les mêmes commandes existent pour la *région* (vue plus tard)
  - `a2ps` appelle l'outil du même nom

Examiner le contenu d'un fichier

- quelques opérations simples :
  - `file` détermine la nature du fichier
  - `cmp` compare deux fichiers, simplement pour dire s'ils sont identiques ou non
  - `diff` (`=` en mode Dired d'Emacs) montre toutes les différences entre deux fichiers
  - `wc` (*word count*) compte les caractères, mots et lignes du fichier
- opérations plus complexes, pouvant éventuellement traiter une hiérarchie entière :
  - `find` parcourt un répertoire entier, selon certains critères
  - `grep` recherche des expressions régulières dans des fichiers
  - le mode `Ediff` d'Emacs (menu `Tools`) effectue un travail très fin de comparaison de deux fichiers, permettant éventuellement d'adapter l'un en fonction de l'autre

La commande `find`

- liste d'arguments de forme particulière :
  - en premier un répertoire
  - puis liste de mots-clés suivis de valeurs pour sélectionner les fichiers concernés :
    - `-type` et une lettre indiquant la sorte de fichier (par exemple `d` pour un répertoire)
    - `-group` et le nom du groupe propriétaire
    - `-user` et le nom de l'utilisateur propriétaire
    - `-name` et une partie du nom à trouver
    - etc.
  - puis une action à effectuer :
    - `-print` affiche le nom de chaque fichier sélectionné
    - `-exec` applique une commande à chaque fichier sélectionné (syntaxe compliquée)
    - etc.
- on peut même enchaîner plusieurs sélections et actions

La commande `grep`

- le premier argument est une expression régulière
- le deuxième argument est une liste de fichiers
- la commande affiche la liste des lignes des fichiers qui contiennent une chaîne décrite par l'expression régulière
- options variées pour compter, inverser le test, chercher l'expression régulière dans un fichier, etc.
- la commande `M-x grep` (ou menu `Tools`) d'Emacs est plus perfectionnée :
  - la liste des apparitions apparaît dans un nouveau tampon, dans l'autre fenêtre
  - la commande `C-x ‘` amène successivement dans la fenêtre de départ les lignes sélectionnées