Unix/Bash – système d'exploitation

Ordinateur

Un modèle en couche

- une couche matériel, une couche système d'exploitation et une couche application

Un ordinateur ?

- une machine programmable servant au traitement de l'information (les données)
- stocke le programme en exécution dans la mémoire ainsi que les données en cours de traitement (mémoire centrale)
- unité de traitement exécutant les instructions du programme et modifiant les données : le(s) processeurs(s)
- ensemble de périphériques permettant d'échanger avec l'extérieur : souris, clavier, etc (périphériques d'entrée/sortie, in/out)

Système d'exploitation (SE)

Programme complexe réalisant les fonctionnalités de base indispensables à tous les utilisateurs (interactions avec le système, les périphériques)
Ex. de SE : Windows, MacOS, Linux, DOS, etc.

- SE mono-processus, mono-utilisateur : exécute un programme à la fois avec un seul utilisateur (DOS)
- SE mono-utilisateur, multi-processus (Windows) : plusieurs fenêtres simultanées correspondant à des programmes différents. SE répartie le processeur à tour de rôle entre les applications.
- SE multi-processus et multi-utilisateurs. Plusieurs utilisateurs peuvent lancer en même temps leurs programmes sur la même machine (Unix, Linux, Solaris, ...).

Les stations de travail à l'ENSEIRB-MATMECA sont de ce type

Système de type Unix

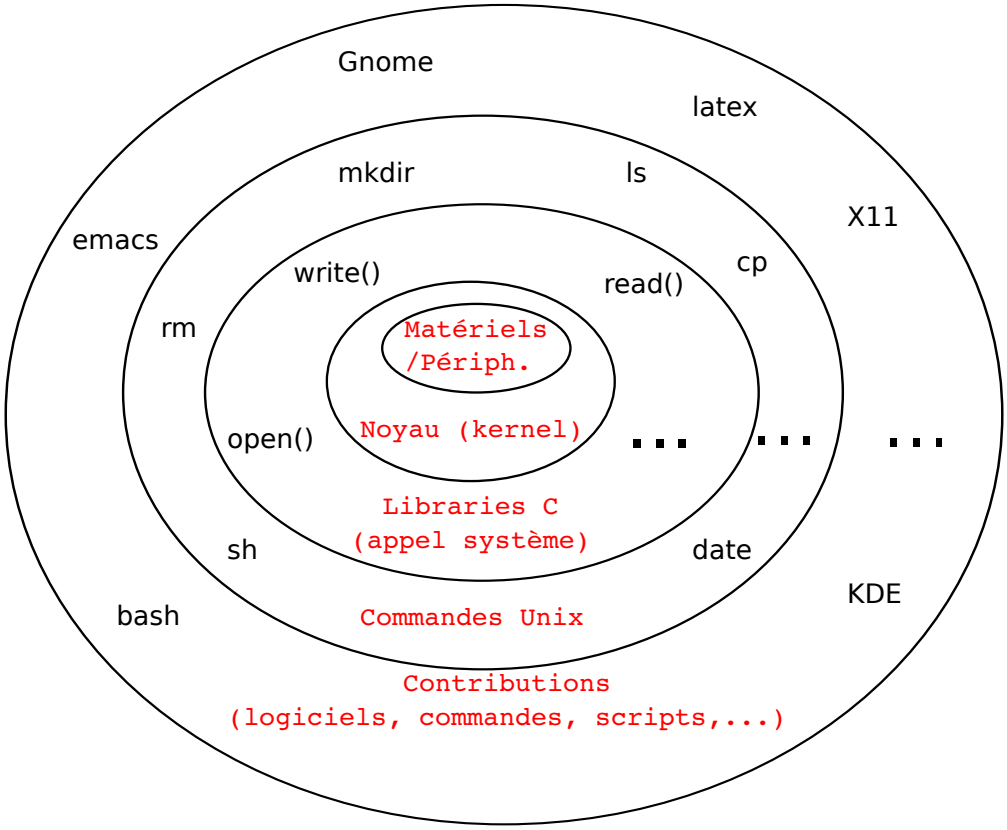
- Plusieurs distributions (versions) par ex : Knoppix, Ubuntu, Linux Mint, Fedora, Gentoo, Debian, ...
- Modèle client/serveur
- Plusieurs terminaux/utilisateurs se connectent à une machine plus puissante, le serveur, à travers le réseau

processus : toute occurrence en mémoire d'un programme en cours d'exécution
Quelques caractéristiques d'UNIX :

- système à temps partagé : exécution concurrente de processus, ordonnancement
- système multi-utilisateur : compte, connexion (login/password), environnement personnalisé
- partage des ressources : arborescence globale des fichiers, mécanismes de protection
- gestion des communications : entre processus, entre utilisateurs
- simple, portable, extensible

Organisation générale

Un noyau central qui gère : système de fichiers, processus, mémoire, communications, etc. Autour un ensemble de commande et logiciels.



Système de fenêtrage

pas de fusion entre SE et système d'interface graphique (différent de Windows)

- Système de fenêtrage indépendant
- Possibilité de le modifier (KDE, Xfce, Gnome, Unity, ...)
- Possibilité de lancer des fenêtres à distance via la commande `ssh -X`
- Possibilité d'utiliser l'ordinateur même sans interface graphique

Compte utilisateur et protection

- connexion à un système multi-utilisateur : nécessité un compte.
- le système connaît la liste des comptes/utilisateurs autorisés
- à chaque compte est associé : un nom **(login)** un mot de passe **(password)** un espace privé **(home directory, ~)**

Rappel sur la protection des comptes/mots de passe

- **Vous ne devez pas** prêter votre compte
- **Vous ne devez pas** prêter votre mot de passe (même pour échanger copier des fichiers)
- **Vous êtes responsable** de ce qui se trouve dans votre compte
- **Vous êtes responsable** de ce que vous faîtes sur le réseau de l'école (logs analysés)
- **Votre mot de passe** doit être complexe (majuscule, minuscule, alpha numérique, etc)
- **Votre mot de passe** doit être résistant à des attaques type brute force ou dictionnaire → pas de noms, pas de mots usuels, ...

Exercice

Première connexion

- Démarrer la machine et choisir l'item 1 (démarrage de la Debian), le noyau est chargé en mémoire, c'est la phase de boot
- Un fois terminé la phase de boot : le système d'exploitation démarre et vous obtenez une invite de commande : connectez vous (login/password)
- mot de passe validé via `http://aaa.ipb.fr`
- si ok : le window manager se lance (gnome), bienvenue sous Linux, vous êtes dans votre *homedirectory*
- chercher dans *Applications/Accessoires* et lancer l'application *Terminal* (interpréteur de commande/console)
- vérifier que vous êtes chez vous en tapant la commande `pwd` (`print name of current/working directory`) (vous devez voir quelque chose de la forme `/truc/machin/votrelogin`)
- lancer dans la console le navigateur internet `firefox`
- vérifier que votre connexion à internet fonctionne

Unix/Bash – le shell

Interpréteur de commande

- shell = interpréteur de commande (shell : coquille qui entoure le noyau)
- interpréteur de commande = programme (processus) qui fournit une interface texte entre l'utilisateur et le noyau
- rôle = lire la ligne de commande et exécuter les actions associées
- plusieurs types = `sh`, `csh`, `zsh`, ...Ici `Bash` (Bourne-Again shell)

Fonctionnement général – le shell exécute la boucle ∞ :

```
POUR toujours FAIRE :
  | Affichage du prompt (prêt à lire) et attente d'une commande
  | Vérification de la syntaxe
  | SI la syntaxe est correcte ALORS :
  |    exécuter la commande
  |    attente de la fin de la commande en cours
  | SINON
  |    afficher un message d'erreur
```

Votre prompt : `'type_de_shell'-'login':` (parfois '$' au lieu de ' :' à la fin)

Une commande ?

- Toute commande Unix a la même syntaxe : (séparateur : l'espace)
  `nom-de-la-commande options arg-1 arg-2 ...arg-n`
- `enter` lance l'interprétation de la commande
- résultat de la commande dans le terminal
- options de la forme `-options` (ex. -a) `--options` (ex. `--verbose`)
- on peut séparer deux commande par ';' - grouper des commandes avec les parenthèses → `(cmd1 ; cmd2; cmd3)`
- le prompt ne revient que lorsque la commande est terminée ; on peut utiliser '&' à la fin de la ligne de commande pour lancer la commande en tâche de fond et reprendre la main

Obtenir de l'aide

la commande `man` :

- affiche et formate l'aide avec `less`
- `man <cmd>` : affiche le manuel de `<cmd>`
- `man -h` : affiche une courte aide sur `man`
- `man <n> <com>` : affiche une page spécifique
  → Tester la différence entre `man 3 sleep` et `man sleep`
- manipuler `less` : 'q' pour quitter, 'space' pour faire défiler d'un écran ; 'up/down' pour faire défiler l'écran ; '/ pattern' faire une recherche ; 'n/N' : résultat suivant/précédent

Autre : `http://www.explainshell.com/`

Historique et raccourcis utiles

Le shell intègre des fonctions de gestion de l'historique des commandes exécutées

- C-r (reverse search) : recherche avec motif
- C-p (previous) : commande précédente (up)
- C-n (next) : commande suivante (down)
- history : affiche l'historique des commandes
- !num : relance la commande de numéro donné dans l'historique

Quelques raccourcis utiles pour le shell

- C-k (kill) : coupe et mémorise le texte depuis la position du curseur jusqu'à la fin de la ligne
- C-y (yank) : insère le texte précédemment supprimé
- C-a : début de ligne
- C-e : fin de ligne
- C-c : termine le processus en cours
- C-z : endort le processus en cours
- C-d : quitte le shell en cours (=`exit`)
- Tab : active la complétion **(à utiliser sans modération)**

Exercices

Les commandes `echo` et `sleep`

- que font ces commandes ?
- que font les commandes suivantes :

```
echo ''Hello''
echo -n ''Hello''
echo -n ''Hello''; echo ''Folks''
sleep 5 ; echo ''Hello''
(sleep 5 ; echo ''Hello'')
```

La commande `xeyes`

- que fait cette commande ? lancer la
- terminer le programme par un raccourci clavier et reprendre la main
- relancer la commande ; endormir le processus avec un raccourci clavier pour récupérer la main
- réactiver le programme pour le mettre en tâche de fond (`bg`)
- tuer/arrêter le programme en console :
  → grâce à la commande `ps` (processus status)
  → déterminer le processus id (pid)
  → grâce à la commande `kill` envoyer le signal `KILL` pour tuer le processus

Exercices

- Essayer les commandes suivantes et interpréter les résultats :
  `dite`
  `date`
  `whoami`
  `pws`
  `pwd`
  `cal`
  `cal 2012`
  `cal 10 2012`
  `who`
  `uname -a`
  `uptime`
  `top` (comment quitter cette commande ?)
  `time sleep 5`
  `history`
- Grâce à la complétion rechercher une commande qui nettoie l'écran (`cl...`) vérifier à l'aide du manuel
- Grâce à la complétion rechercher une commande qui permet de quitter le shell (`ex...`). Vérifier à l'aide du manuel

Unix/Bash – arborescence et système de fichier

Système de fichier

Le système de fichier est un arbre dont la racine est '/' :

- les noeuds : des répertoires
- les feuilles : des fichiers

Petit lexique

- fichier : file
- répertoire : directory
- chemin : path
- répertoire de connexion : home directory (tilde)
- répertoire courant : working directory (`pwd`))

Particularité : sous Unix tout peut être vu comme un fichier

- fichier régulier : fichier contenant du texte, du codes source, des commandes exécutable, ...
- répertoire : un fichier contenant d'autres fichiers
- fichiers spéciaux : lien vers des périphériques, canaux de communications entre processus

Arborescence standard

Dans la majorité des cas (variantes possibles selon les versions)

- les répertoires de fichiers exécutables : `/bin, /usr/bin, /usr/local/bin`
- répertoires de bibliothèques : `/lib, /usr/lib, /usr/local/lib`
- pages de manuel : `/usr/man`
- répertoire relatif à l'administration `/etc` ex. `/etc/passwd` (contenant des infos sur les utilisateurs)
- divers : `/tmp`, `/dev` (fichiers spéciaux : drivers)

Chaque processus lancé possède un répertoire courant : sa position dans l'arborescence

- Ex. lors la connexion, le répertoire courant du shell est le répertoire de l'utilisateur (home directory ou ~)
- Pour modifier cette position courante on utilise la commande : `cd (change directory)`

Notion de chemin

Un chemin :

- identifie un fichier (répertoire, driver, ...) dans le système
- correspond à un déplacement dans l'arborescence des fichiers
- préfixe + un nom de base : `basename`, le nom après le dernier '/'
- le préfixe est toujours le chemin d'un répertoire
- est valide si son préfixe est le chemin d'un répertoire existant

Deux noms réservés, soit R un répertoire

- `'.'` : référence le répertoire R ; `'..'` : référence le répertoire père de R

Soit l'arborescence suivante

```
/usr/
 |__local/
    |--bin/
    |  |__emacs
    |--lib/
       |__libm.so
```

Chemin relatif : déplacement par rapport à la position courante
Si le répertoire courant est `/usr/local/` les chemins

- `../local/bin` référence un répertoire équivalent à `bin` ou `./bin`
- `bin/emacs` et `../local/lib/libm.so` référencent des fichiers

Chemin absolu : déplacement par rapport à la racine '/' (indépendant de la position actuelle)

- `/usr/local/bin/emacs` référence un fichier ; `/usr/local/lib` référence le répertoire

Commandes de manipulation de fichiers

Conventions de nommage des fichiers

- pas d'espaces : utiliser "_" (underscore) ou "-" (minus)
- pas de caractères spéciaux : pouvant être interprété par le shell
- ajouter un suffixe pour désigner les types de fichiers : .txt, .html, .c, .sh, .tgz, ...

Commandes de manipulation de fichier à connaître

- cd (change the working directory)
- pwd (print name of current/working directory)
- ls, -l, -la, -G (list directory contents)
- cp, -r (copy files and directories)
- mkdir (make directories)
- mv (move (rename) files)
- rm, -r (remove files or directories) (**attention rm -rf**)
- touch (change file timestamps) si le fichier n'existe pas création d'un fichier vide
- file (détermine file type)
- cat (concatenate files and print on the standard output)
- less (opposite of more)
- wc, -l, -w, (print newline, word, and byte counts for each file)
- chmod (change file mode bits)

Droits, protection et accès

3 types de propriétaires

- `user 'u'` (le propriétaire) , `group 'g'` (pas forcément le même le user) et `others 'o'`

3 types d'accès

- fichier : `read 'r', write 'w', execute 'x'` ;
  répertoire : `read 'r, write 'w', use 'x'`

Le mode d'accès

- définit l'accès en lecture ; écriture ; exécution ; pour le propriétaire ; le groupe et les autres
- 9 informations élémentaires : 3 champs, 3 flags
- `rwx rwx rwx (u g o)` "-" indique un accès non autorisé

Visualiser les droits : `ls -l, ls -ld`
Modification : commande `chmod` *options mode file/dir*
Mode symbolique : Qui ? `u, g, o, a` Opération ? `=, +, -` Permission ? `r, w, x`
par ex : `a+rwx; u=rwx,g+w,o-r`
Mode octal : codage pour chaque type de propriétaire :
$r \times 2^2 + w \times 2^1 + x \times 2^0 = 4 + 2 + 1$
par ex : `777`, `640`

Démonstration : travail à distance

Connexion sécurisée à une machine distante

- `ssh login@ssh.enseirb-matmeca.fr` (secure shell)

Connexion sécurisée à une machine distante avec export graphique

- `ssh -X login@ssh.enseirb-matmeca.fr` (X-Windows)

Copier un fichier vers une machine distante

- `scp fichier login@ssh.enseirb-matmeca.fr:/path` (secure copy, ' :')

Copier un répertoire vers une machine distante

- `scp -r rep login@ssh.enseirb-matmeca.fr:/path` (récursivement)

Copier un fichier de la machine à distante vers la machine locale

- `scp login@ssh.enseirb-matmeca.fr:/path  /path`

Utilisation avancée

- Utilisation de clés : ssh-keygen
- Configuration du client ssh (`man ssh_config`)

Exercices

Essayer la séquence de commandes suivantes et donner/interpréter les résultats :

- `cd`
- `pwd`
- `ls -la`
- `cd .`
- `pwd`
- `cd ..`
- `pwd`
- `ls -la`
- `cd ..`
- `pwd`
- `cd /etc`
- `ls -la`
- `cat passwd`
- `cd -`
- `cd ..`
- `pwd`
- `cd ..`
- `pwd`
- `cd`
- `pwd`

Exercices

Après chacune des commandes suivantes, vérifier que la commande a été validée grâce à la commande `ls`

- Créer un répertoire nommé `subdir`
- Copier le fichier `/etc/passwd` dans `subdir`
- Copier le fichier `/etc/passwd` dans `subdir` en le nommant `mypasswd`
- Allez dans le répertoire `subdir`
- Vérifier la présence des fichiers `passwd` et `mypasswd`
- Renommer le fichier `passwd` en `passwd.bak`
- Afficher le contenu de ce `passwd.bak`
- Remonter au répertoire père de `subdir`
- Copier `subdir` en `subdir.bak`
- Supprimer `subdir`
- Enlever le bit de lecture sur le fichier `passwd.bak` et vérifier à nouveau les droits
- Afficher le contenu de ce `passwd.bak`
- Enlever le bit autorisant le parcours sur le répertoire `subdir.bak`
- Supprimer le répertoire `subdir.bak`
- Faîtes les modifications nécessaires pour pouvoir supprimer ce répertoire

Exercices

1.
   - Expliquer `-rw-------`
   - Expliquer `drwx--x--x`
   - Afficher le contenu du fichier `~mfaverge/tmp/file.txt`
2.
   - A qui appartient le fichier `/etc/passwd` ?
   - Pourquoi n'avez vous pas le droit de le modifier ?
   - Quels sont vos droits sur ce fichier
3. Donner deux façons possible d'obtenir les droits suivants (en symbolique et en octal)
   - `-rw-r--r--`
   - `-rwxr-xr-x`
   - `-rwxr-xr--`
   - `-r--r--r--`
   - `-r----x---`
4. Soit le répertoire `drwxr-xr-x 2 user staff 68B Mar 23 12:01 tmp` Donner et expliquer le résultat des commandes suivantes sur `tmp` en supposant que `tmp` ce trouve dans le homedir de `user`
   - `chmod 355 tmp; ls tmp`
   - `cd tmp; pwd`
   - `cp /etc/passwd .; ls`
   - `cd ..; chmod 755 tmp; ls tmp`
   - `chmod 655 tmp; cd tmp`
   - `chmod 555 tmp; rm -f tmp/passwd`

   Faîtes les modifications nécessaires jusqu'à suppression de ce répertoire.

Unix/Bash – substitution, variables, entrée-sortie

Substitutions et expansions

Analyse de la ligne de commande par le shell : différentes phases Ex. ~ = chemin vers `homedir`, `*.c` = tous les fichiers se terminant par `'.c'` dans le répertoire courant.

Expansion des accolades :

- `a{b,c,d}e` = `abe ace ade`

Expansion des variables, variables d'environnement :

- syntaxe d'une variable : `$VAR`, `${VAR}`, `${VAR:-défaut}`
- affectation : `export VAR=value` (rend accessible la variable à l'environnement) ou `VAR=value` (pas forcément le cas)

Substitution de commandes (entre backquote) :

- `` `cmd` `` exécute `cmd` dans un sous-shell et renvoie les résultat de l'exécution de la commande

Coupures des mots :

- toute séquence de blancs (espaces, tabulations, ou retours de ligne) délimite chaque mot (paramètre, option) de la commande
- pas d'espaces dans les noms de fichiers
- caractère d'échappement '\' : protège le caractère qui le suit.
  Ex. `mkdir my\ dir`
- `'une chaîne de caractère'` : pas d'interprétation du contenu
- `"une chaîne de caractère"` : interprétation du contenu

Substitutions et expansions

Expansion des noms de fichiers

- Analyse de la ligne de commande et recherche des caractères :
  `'*'`, `'?'`, `'['`, `']'`.
- `*` : n'importe quelle chaîne, même la chaîne vide ;
- `?` : n'importe quel caractère ;
- `[...]` : n'importe quel caractère spécifié entre les crochets :
  - suite de caractères signifie = l'un ou l'autre de ces caractères
  - le tiret '-' (s'il n'est pas le premier caractère) sert à définir un domaine de valeur ex. `a-i`

Exercices

- Que fait `echo projet/{src/{module1,module2},lib,bin}` ?
- Que produit la séquence suivante ?
```
echo ${BROL:-Je ne le connais pas}
export BROL="Monsieur Brol"
echo ${BROL:-Je ne le connais pas}
```
- Que produit `echo $PATH` ? Que représente cette variable d'environnement ?
- Que fait la séquence suivante : `export $VAR=‘date‘; echo $VAR` ?
- Que fait la séquence suivante : `export BROL="Monsieur Brol"`
```
echo "$BROL et trol"
echo '$BROL et trol'
```
- Que fait `ls -l /usr/lib/lib[a-jt-z]??e*`
- Que fait la séquence suivante : `RAC=/bin`
```
echo $RAC
echo $RAC/pwd
$RAC/pwd
```

Exercices

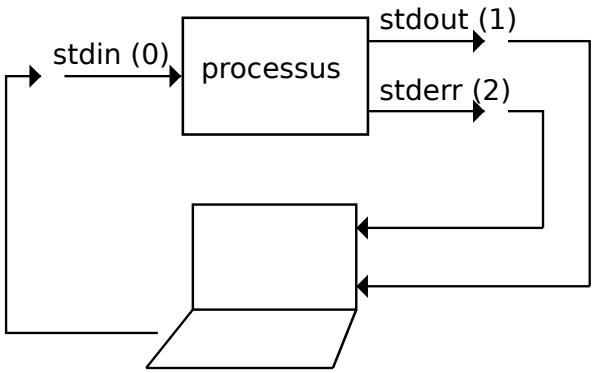
- Que fait la séquence suivante : `echo /bin/[a-c]?`
- Que fait la séquence suivante : `echo '/bin/[a-c]?'`
```
N='/bin/[a-c]?'
echo $N
```
- Que fait la séquence suivante : `A=un`
```
bash; echo /$A/
^D
export A; bash; echo /$A/
A=deux; B=trois; echo /$A/$B
^D
echo /$A/$B
```

Mécanismes d'entrées/sorties

Généralement une opération d'E/S :

- transfert de données entre une zone mémoire et un fichier ou un périphérique
- Ex. un processus effectue des E/S : depuis le clavier vers l'écran

A sa création chaque processus possède 3 flots E/S standards

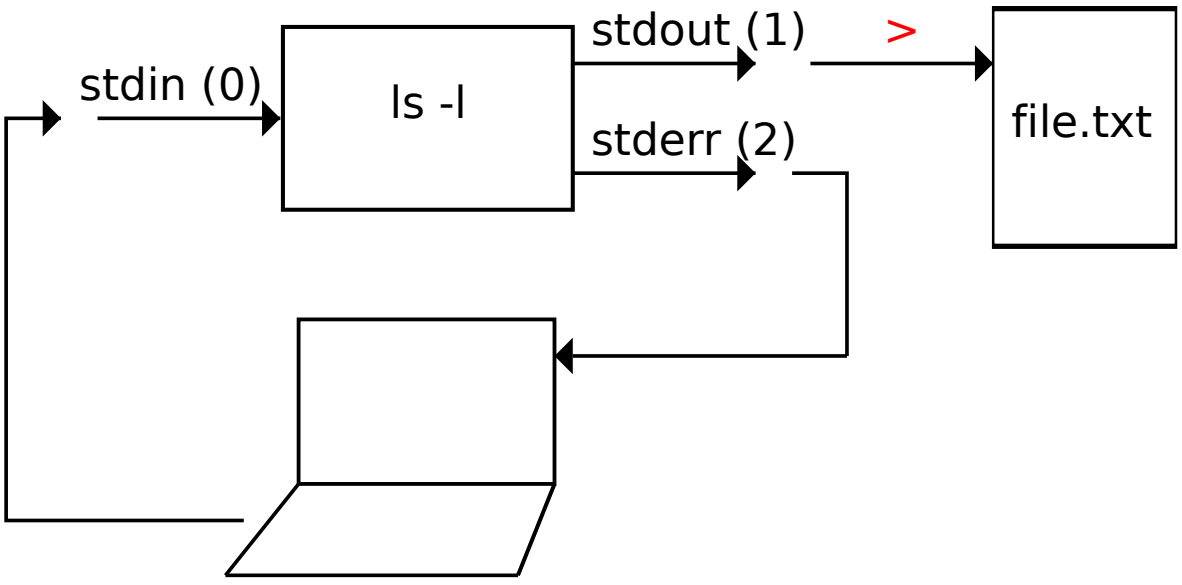


- `stdin (n° 0)` : entrée standard opérations lecture, par défaut le clavier
- `stdout (n° 1)` : sortie standard opérations d'écriture, par défaut l'écran
- `stderr (n° 2)` sortie d'erreur opérations d'affichage des messages d'erreurs, par défaut l'écran

Unix, permet à l'utilisateur de modifier et de reconnecter les flots d'E/S
c'est le mécanisme de redirection

Mécanismes de redirection (1)

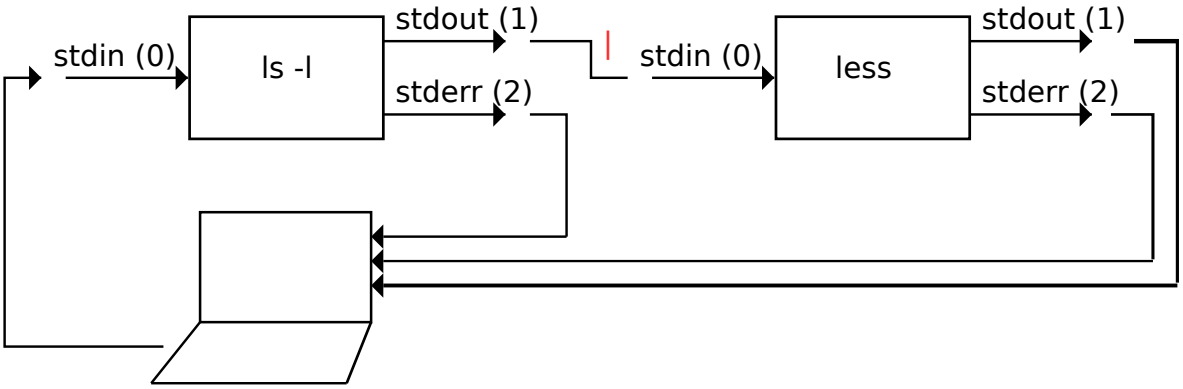
Rediriger un processus (programme) vers un fichier via '>'.
Ex : `ls -l > file.txt`



Si le fichier existe il est écrasé

Mécanismes de redirection (2)

Rediriger un processus (programme) vers un autre processus (programme) via le `pipe` (tube) : `'|'`
Ex. `ls -l | less`



Possibilité de créer des pipelines complexes en une seule ligne de commande

Exercices

D'autres mots clés existent : les plus communs

- `< file` : redirection de l'entrée standard dans un fichier
- `> file` : redirection de la sortie standard dans un fichier (déjà vu)
- `>>` : idem, avec positionnement à la fin du fichier s'il existe
- `cmd | cmd` : le pipe (déjà vu)
- `2 > file` : redirige la sortie d'erreur vers `file`
- `2 >& 1` : fusionne la sortie d'erreur sur le sortie standard

1. Que fait la ligne de commande
   `date > tmp.txt; ls -l  /.??* | wc -l >> tmp.txt.`
   Que contient finalement `tmp.txt`
2. Que fait la ligne de commande
   `echo ‘date‘ : ‘who | wc -l‘ util. sur ‘hostname‘`
3. Rediriger la sortie de `ls -la` vers un fichier
4. Placez vous dans votre `homedir`
   - Déplacez vous dans le répertoire `dir`, un message d'erreur est-il apparu ? si oui où ?
   - Recommencer mais en redirigeant la sortie d'erreur vers le fichier `err`.
   - Un message est-il apparu ? le fichier `err` a-t-il été créé ? Que contient le fichier `err` ?
   - Recommencer mais cette fois avec en positionnant le curseur à la fin. Que contient le fichier `err` ?

University of Minnesota
Supercomputing Institute
for Digital Simulation and Advanced Computation
1200 Washington Ave S • Minneapolis, MN 55415 • Tel: (612)625-1818

UNIX Reference Card

Warnings!!

When a file has been **DELETED** it can only be restored from a backup. The original is gone!

When a file is **OVERWRITTEN** it has been changed forever! It can only be restored from a backup.

Directory Abbreviations

~	home directory (tilde)
~*username*	another user's home directory
.	current working directory
..	parent of current working directory

Communication

ssh [*options*] hostname
Ssh (Secure Shell) a program for logging into a remote host. Replaces telnet, rlogin, and rsh
options:

-l *login_name*	specifies the user to log in on the remote machine

scp [*options*] user@host1:file1 user@host2:file2
Secure copy files between hosts on a network; uses ssh for data transfer.
options:

-p	preserve modification times
-r	recursively copy entire directories

Comparison

diff [*options*] *file1 file2*
Compare two text files.
options:

-a	treat all files as text files
-b	ignore repeating blanks and end-of-line blanks; treat successive blanks as one
-i	ignore case in text comparison
-q	output only whether files differ

File Management

cat [*options*] [*files*]
Read one or more *files* and print them on standard output. Use the > operator to combine several files into a new file; use >> to append files to an existing file.
options:

-n	print the number of the output line to the line's left
-s	squeeze out extra blank lines

cd [*dir*]
Change working directory to dir; default is the users home directory.

chgrp *newgroup files*
Change the group of one or more *files* to *newgroup*. *newgroup* is either a group ID number of a group name. Only the owner can change the group.
options:

-c	print information about those files that are affected
-R	recursively apply changes to subdirectories

chmod [*options*] *mode files*
Change the access *mode* (permissions) of one or more *files*. Only the owner of a file or a privileged user may change its mode.
options:

-c	print information about affected files
-R	recursively apply changes to subdirectories

mode:
can be numeric

4	read
2	write
1	execute

or an expression of the form *who opcode permission*. *who* is optional (if missing, default is **a**)
who

u	user
g	group
o	other
a	all (default)

opcode

+	add permission
–	remove permission
=	assign permission

permission

r	read
w	write
x	execute
X	set execution permission only if executable by user

cp [*options*] *file1 file2*
cp [*options*] *files directory*
Copy *file1* to *file2*, or copy one or more *files* to the same names under *directory*.
options:

-a	preserves attributes of original files
-f	remove existing files in the destination
-i	prompt before overwriting destination files
-r	recursively copy directories
-s	make symbolic links instead of copying

file [*options*] *files*
Classify the named *files* according to the type of data they contain.

less [*options*] [*filename*]
A program for browsing or paging through files or other output. Can use arrow keys for scrolling forward or backward.
options:
see **man** pages for options (type: **man less**)

ln [*options*] *sourcename* [*destname*]
ln [*options*] *sourcenames destdirectory*
Create links for files, allowing them to be accessed by different names.
options:

-b	backup files before removing originals
-i	prompt for permission before removing files
-s	create a symbolic link. This lets you see the name of the link when you run **ls -l** (otherwise there is now way to know the name that a file is linked to).

ls [*options*] [*names*]
List the contents of a directory. If no *names* are given, the files in the current directory are listed.
options:

-a	list all files, including hidden files
-c	list files by status change time
-l	long format listing (permissions, owner, size, modification time)

mkdir [*options*] *directories*
Create one or more directories.
options:

-m *mode*	set the access *mode* for new directories. See **chmod** for *mode* formats.
-p	create intervening parent directories if they don't exist

more [*options*] [*files*]
Display the content of the named *files* one screen at a time. See less for an alternative.
options:
see **man** pages for options (type: **man more**)

pwd
Print the full pathname of the current working directory.

scp [*options*] user@host1:file1 user@host2:file2
Secure copy files between hosts on a network; uses ssh for data transfer.
options:

-p	preserve modification times
-r	recursively copy entire directories

mv [*options*] *sources target*
Move or rename files and directories. The *source* and *target* determine the result.

source	*target*	*result*
file	*name*	rename file as *name*
file	existing file	overwrite existing file with source file
directory	*name*	rename directory as *name*
directory	existing	move directory to be a
directory	subdirectory of existing directory	

options:

-b	back up files before moving
-f	force the move
-i	query user before removing files

rm [*options*] *files*
Delete one or more *files*. ***Once a file or directory has been removed it can only be retrieved from a backup!***
options:
-d remove directories, even if they are not empty
-f remove files without prompting
-i prompt for file removal
-r recursively remove an entire directory and its contents, including subdirectories. ***Be very careful with this option.***

Miscellaneous

! Repeat the last command

!*string* Repeat the last command beginning with *string*.

cal [*-jy*] [[*month*] *year*]
Print a 12-month calendar for the given *year* or a one-month calendar of the given *month* and year. No arguments, print a calendar for the current month.
options:
-j display Julian dates
-y display entire current year

clear
Clear the terminal display

history
Display list of most recently executed commands

kill [*option*] *IDs*
Send a signal to terminate one or more process *IDs*.
options:
-l list all signals
-signal the signal number (from **ps -f**) or name (from **kill -l**). You can kill just about any process with a signal number of 9.

man *command*
Display information from the online reference manuals.

jobs [*options*] *job_id*
Display status of jobs in the current session. Simply specifying jobs returns the status of all stopped jobs, running background jobs, and all suspended jobs.
options:
-l provide more information about each job listed
-p display only the process IDs for the process group leaders of the selected jobs

whereis *command*
Locate a *command*; display the full pathname for the *command*.

which [*commands*]
List which files would be executed if the named *commands* had been run.

Searching

egrep [*options*] [*regexp*] [*files*]
grep [*options*] [*regexp*] [*files*]
Search one or more *files* for lines that match a regular expression *regexp*. To include characters such as +, ?, |, ( ), blank spaces, etc. enclose these expressions in quotes. See **man** pages for the differences between **egrep**, **fgrep**, and **grep**.
options:
-c print only a count of matched lines
-i ignore case
-l list filenames but not matched lines
-n print lines and their line numbers
-v print all lines that do not match *regexp*

find [*pathnames*] [*conditions*]
Useful for finding particular files. **find** descends the directory tree beginning at each *pathname* and locates files that meet the specified *conditions*.
options:
-name *pattern* find files whose name matches *pattern*
-print print the matching files and directories using their full pathnames
see **man** pages for options (type: **man find**)

Storage

compress [*options*] [*files*] – compress file
uncompress [*options*] [*files*] – uncompress compressed file
compress reduces the size of the named *files*. When possible the resulting compressed file will have the file extension **.Z**. Compressed files can be restored using **uncompress**.
options:
-d uncompress file, same as **uncompress**
-v prints the percentage reduction
-V prints the version of compress

gzip [*options*] [*files*] – compress file
gunzip [*options*] [*files*] – uncompress gzipped file
GNU compression utility. Renames compressed files *filename.gz*. Uncompress with **gunzip**.
options:
-d uncompress file, same as **gunzip**
-r recursively compress or decompress files within a directory
-v print name and percent size reduction for each file

tar [*options*] [*tarfile*] [*other-files*]
Copy *files* to or restore *files* from an archive. If any files are directories, **tar** acts on the entire subtree.
options:
-c create a new archive
-d compare the files stored in tarfile with other-files
-r append other-files to the end of an existing archive
-t print the names of files in archive
-v verbose, print filenames as they are added or extracted
-x extract *other-files* from archive, or extract all files if *other-files* not specified

System Status

Control-C
Stop (interrupt) job running in the foreground

Control-Z
Suspend job running in the foreground

date [*options*] [*+format*] [*date*]
Print the current date and time. You may specify a display *format*.
options:
see **man** pages for options (type: **man date**)

df [*options*] [*name*]
Report the amount of free disk space available on all mounted file systems or on a given *name*.
options:
-k print sizes in kilobytes

du [*options*] [*directories*]
Print disk used by each named directory and its subdirectories.
options:
-k print sizes in kilobytes
-s print only the grand total for each directory

env [*option*] [*variable=value ...*] [*command*]
Display the current environment or, if an environment *variable* is specified, set it to a new *value* and display the modified environment.
option:
-u unset the specified *variable*

ps [*options*]
Report on active processes.
options:
-a list all processes except processes not associated with the terminal
-e list all processes
-l produce long format listing
-u *list* list for usernames in *list*

quota [*option*]
Display disk usage and limits
option:
-v report quotas even if they haven't been exceeded

Contact Information

Phone: 612-626-0802

Email: help@msi.umn.edu

WWW: http://www.msi.umn.edu

http://www.msi.umn.edu/user_support