Commandes Unix plus avancées

V. Berry, Université Montpellier 2

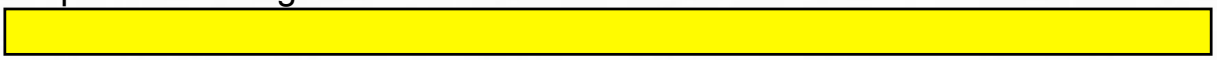
Programme

- [x] Notion d'architecture d'un ordinateur
- [x] Notion de Systèmes d'exploitation («SE»)
- [x] Manipulation du Système de Gestion de Fichiers («SGF»)
- [x] Le terminal de commandes
- [x] Les processus
- [x] Les éditeurs nano & emacs
- [ ] Commandes avancées & re-directions
- [ ] Manipulation d'archives
- [ ] Variables d'environnement et fichiers de configuration
- [ ] Scripts
- [ ] Notions de réseau

Méta caractères du shell

Ce sont des caractères qui ont un sens spécial :

`!` `^` `*` `?` `[]` `\` `;` `&` `<` `>` `|` `>>` `espace`

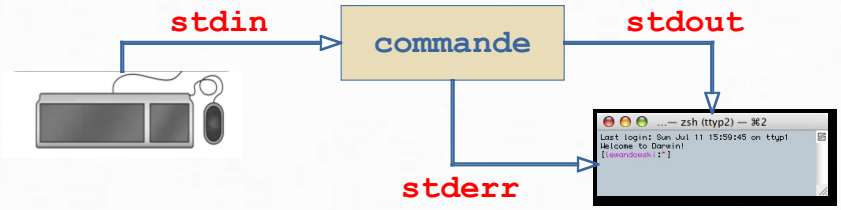
- 
- L'astérisque ou étoile: `*`
  
- Le point d'interrogation: `?`
  
- Le point-virgule: `;`
  - Séparateur de commandes
- L'anti-slash: `\`

  - Exemple : `ls Metallica\ Sonate\ 9\ si\ majeur.mp3`

- Texte entre `'` `'` (simples quotes): le texte n'est pas interprété par le shell (donc peut contenir des caractères spéciaux), il est considéré comme un seul *mot*

Les re-directions

- Une commande ouvre 3 descripteurs de fichiers ; par défaut :



- Une redirection consiste à remplacer un de ces canaux par défaut par/vers une autre commande ou un fichier

<f1	
>f2	
>>f2	
c1 |	
2>f1	
&>f2	

Les re-directions

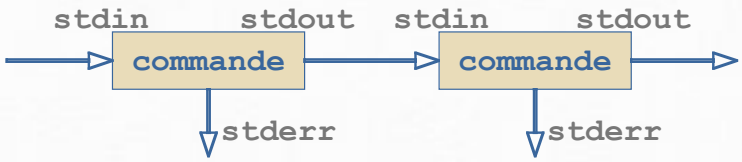
<	redirige l'entrée standard
>	redirige la sortie standard
>>	concatène la sortie standard
|	redirige la sortie standard vers l'entrée d'une autre cmde
2>	redirige la sortie d'erreur
&>	redirige la sortie standard **et** la sortie d'erreur

Exemples :	Effet
`ls > liste`	écrit (écrase) dans le fichier `liste` la liste des fichiers et répertoires du rép. courant
`ls .. >> liste`	ajoute à la fin du fichier liste le contenu du rép. parent
`wc -l < liste`	compte le nb de lignes du fich. `liste`

Les re-directions

Le tube (*pipe*) permet de réutiliser le résultat d'une commande comme entrée d'une autre commande

|	redirige la sortie standard vers l'entrée d'une autre cmde



QUIZZ

Exemples : «combien de fichiers dans le rép. courant ?»

sans pipe:

avec un pipe:

Les filtres

Un **filtre** est une commande capable de traiter un flux de données et d'en effectuer un affichage formaté et/ou sélectif

`cat`	– affiche le contenu des fichiers passés en paramètres (par défaut, **`stdin`**)
`more`	– affiche *page par page* le contenu des fichiers passés en paramètres (par défaut, **`stdin`**)
`grep`	– affiche les lignes du fichier passé en paramètre qui vérifient une expression régulière donnée
`cut`	– sélectionne uniquement certaines colonnes du fichier passé en paramètre

on gagne à utiliser la cmde less plutot que la cmde more (moins user-friendly)

Les filtres

Exemple de commandes d'**affichage** de contenu d'un fichier :

«afficher le contenu du fichier **`MonAppli.cpp`** en montrant aussi les caractères non visibles et les fins de lignes» :

```
cat -v MonAppli.cpp
```

«afficher le contenu du fichier **`essai.txt`** en groupant les lignes blanches (-s) » :

```
more -s essai.txt
```

- '=' pendant la visualisation permet de connaître le numéro des lignes affichées
- '/' pendant la visualisation permet de chercher les occurrences d'un mot dans le texte (mise en surbrillance)
- Contrairement à ce qu'indique son nom, la commande **`less`**, fait plus que la commande **`more`** (même syntaxe, retours arrières, meilleure gestion des gros fichiers).

Les filtres

Commandes de **sélection de parties** d'un fichier :

egrep ***motif*** **[fichier]** *

recherche, dans le fichier passé en paramètre, les lignes vérifiant un motif donné.

Les motifs recherchés sont parfois composés de caractères non connus à l'avance.

Exemple (extrait de la commande `last`) : «quels sont les utilisateurs qui se sont connectés le 21 septembre avant 10h ?»

```
vberry   pts/2        tp5-pc01 Fri Sep 22 15:15   still logged in
mcvill   pts/0        tp3-pc12 Fri Sep 22 08:01   still logged in
fiorio   pts/0        dir-pc02 Thu Sep 17 19:58 - 00:32  (04:33)
pochard  pts/2        dir-pc01 Thu Sep 21 09:21 - 10:21  (01:00)
fiorio   pts/0        tp3-pc13 Thu Sep 17 10:57 - 18:09  (07:12)
```

Une expression régulière permet de décrire un motif cherché de façon flexible en utilisant des méta-caractères :

*egrep est une variante étendue de **grep**, fournissant une plus grande puissance d'expression pour définir les motifs cherchés*

Les filtres

`egrep`

Commandes de **sélection de parties** d'un fichier.

Méta caractères utilisables dans une expression régulière :

`.`	un caractère quelconque
`+`	une ou plusieurs occurrences
`*`	zéro, une ou plusieurs occurrences
`[<liste>]`	au choix parmi les possibilités indiquées
`[^<caractère>]`	tout sauf un certain caractère
`{x}`	exactement x fois le caractère précédent
`{x,y}`	entre x et y fois le caractère précédent
`^`	(en début d'expression) = début de la ligne
`$`	fin de la ligne
`\`	pour *dé-spécialiser* un méta-caractère

Les filtres

Exemples de **sélection de parties** d'un fichier :

QUIZZ

- «quelles sont les lignes du fichier **`documentation.txt`** qui contiennent le mot **`options`** ?» :

- «quelles sont les lignes du fichier **`HOWTO`** qui commencent par une minuscule ?»

«combien de sous-répertoires du répertoire **`Musiversal`** contiennent un fichier **`readme.txt`** ?»

Les filtres

Commandes de **sélection de parties** d'un fichier.

QUIZZ

Exemple : «**quels sont les utilisateurs qui se sont connectés (cmde `last`) le 21 septembre avant 10h ?**»

```
vberry   pts/2        tp5-pc01 Fri Sep 22 15:15   still logged in
mcvill   pts/0        tp3-pc12 Fri Sep 22 08:01   still logged in
fiorio   pts/0        dir-pc02 Thu Sep 17 19:58 - 00:32  (04:33)
pochard  pts/2        dir-pc01 Thu Sep 21 09:21 - 10:21  (01:00)
fiorio   pts/0        tp3-pc13 Thu Sep 17 10:57 - 18:09  (07:12)
```

Pour d'autres méta-caractères, voir le TP

Manipulation d'archives

Archivage et compression

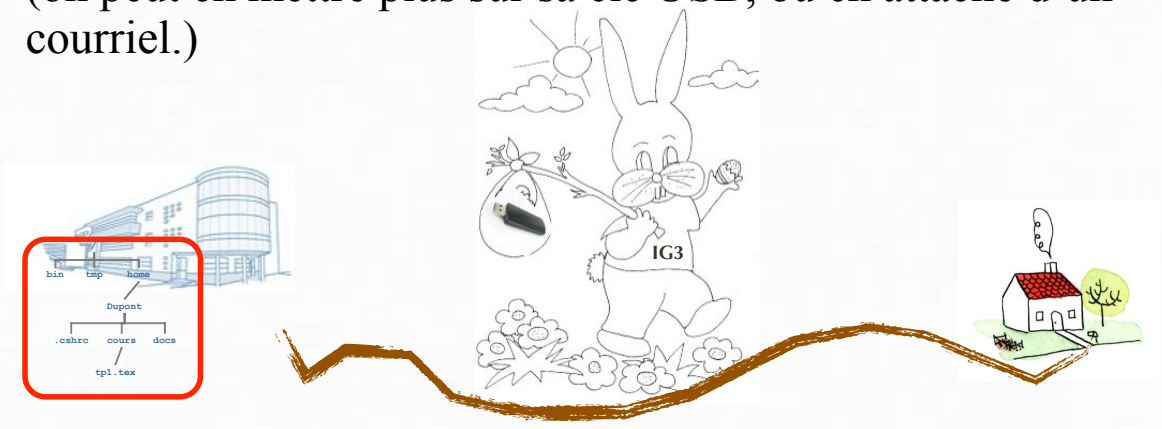
Définitions

- La compression consiste à regrouper un ensemble de fichiers et à réduire leur taille le plus possible afin de faciliter leur transfert.
- Le fichier contenant l'ensemble des autres fichiers compressés est généralement appelé fichier archive.
- Il existe plusieurs formats pour les archives :
  - sous Windows : zip et rar sont les plus répandus
  - sous Unix : .tar.gz et tar.bz2

Archivage et compression

Les raisons

- Simplifie le transfert (ou l'envoi par mél) : un seul fichier plutôt que plusieurs.
- Réduit la quantité de données à transférer, à sauvegarder (on peut en mettre plus sur sa clé USB, ou en attache d'un courriel.)



Archivage et compression

Les outils

Il existe de nombreuses applications qui permettent d'archiver-et-compresser :

- Pour Windows : il s'agit principalement d'outils réalisés en complément du SE par des éditeurs/programmeurs divers :
  - *WinZip* ou *Winrar*
  - *PowerArchiver* (shareware)
  - *7-zip* (**freeware**)
- Pour Unix : on distingue généralement **l'assemblage** d'un ensemble de fichiers/répertoires en un seul fichier (*archive*)
  - commande `tar`

... et la phase de **compression** de l'archive :

- commandes `zip`, `gzip`, `bzip2`, `compress`, ...

Archivage et compression

La commande `tar` :

L'essentiel de la commande :

`tar [c|t|x][z][v][f device][fichier(s)|répertoire(s)]`

- c : (*create*) crée une nouvelle archive. (efface d'éventuels fichiers/réps présents dans l'archive auparavant
- t : (*list*) affiche le contenu de l'archive
- x : (*extract*) extrait le(s) fichier(s) désigné(s) de l'archive et le(s) restaure en local sur le système. Désigner un répertoire par son nom aboutit à désarchiver son contenu récursivement.
- z : pour une archive compressée (par `gzip`)
- v : affiche d'informations lors de la manipulation de l'archive
- f *device* : force `tar` à utiliser l'argument suivant comme nom d'archive

les fichiers désarchivés appartiennent à celui qui les désarchive (peut importe l'endroit (répertoire) ou le créateur de l'archive

Archivage et compression

La commande gzip (voir aussi bzip2) :

`gzip [-d] [-l] [-r]` *`fichier(s)`*

-d **d**écompresse le fichier (mais ne désassemble pas dans le cas d'une archive *.tar*

-l donne une **l**iste d'information sur chaque fichier : taille quand compressé et non compressé, ...

-r procède récursivement, i.e. traite (séparément) tous les fichiers trouvé dans la sous-arborescence

Exemples : «combien de fichiers dans le rép. courant ?»

créer un fichier compressé :

```
gzip uneArchive.tar
```

décompresser un fichier :

```
gzip -d uneArchive.tar.gz
```

Variables et scripts

Variables

Variables d'environnement

Des informations sur l'environnement dans lequel s'exécutent les processus sont stockées dans des variables :

- informations liées au S.E. et aux logiciels installés (`OSTYPE`,`JAVA_HOME`...)
- informations liées à l'utilisateur : (`USER`,`HOME`,`PWD`,`SHELL`,...)

Ces variables communiquent des informations entre programmes

Exemples :

```
[vberry@io ~]$ echo $HOME
/auto_home/vberry
[vberry@io ~]$ echo $SHELL
/bin/bash
[vberry@io ~]$ env
OSTYPE=linux-gnu
PATH=/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/
  usr/local/atelierb/AB/bbin/:/usr/local/java/bin:.:/auto_home/
  vberry/bin
UID=25587
```

Variables et scripts

Fonctionnement :

Quand un processus démarre son exécution, le processus qui le lance lui transmet les variables de son environnement sous la forme de couples **nom=valeur**

La commande `set` permet à un **shell de type *bash*** de définir de nouvelles variables.

La commande `export` permet d'ajouter une variable dans son environnement. Celle-ci sera exportée vers ses processus fils, *lors de leur création* :

- les commandes lancées depuis ce shell (ex: `ls ~`)
- les scripts exécutés depuis ce shell (ex: `running.pl`)

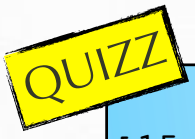
Variables et scripts

variable PS1 = prompt affiché par le shell

- **\u** – Nom de l'utilisateur
- **\h** – Nom de la machine
- **\w** – chemin complet du répertoire courant
- \W - nom du répertoire courant
- **\e[** - début d'une coloration du prompt
- **x;ym** - Indique le code couleur
- **\e[m** - fin d'une coloration du prompt
- $(linux_command). exemple : `$(date +%k:%M:%S)`

codes couleurs

```
Black 0;30
Blue 0;34
Green 0;32
Cyan 0;36
Red 0;31
Purple 0;35
Brown 0;33
```



Comment obtenir le prompt suivant ?

```
[15:47]  vberry Cours >
```

export set PS1="[\$(date +%k:%M)] \e[0;36m \u\e[m \e[0;33m\W\e[m> "

Variables et scripts

Alias :

La pratique des commandes d'un Système d'Exploitation (cf les TPs), montre que pour certaines commandes, on a tendance

- à utiliser souvent les mêmes options : `ls -l`
- à enchaîner souvent certaines commandes :

```
cd <nomrep> ; ls
```

Il est possible de définir des alias (sortes de raccourcis) pour ces combinaisons utilisées fréquemment :

```
[vberry@i17 ~]$ alias ll = "ls -l"
[vberry@i17 ~]$ ll ~
total 40
drwx------ 2 amurillo amurillo 4096 sep  9 13:49 gconfd-amurillo
drwx------ 2 epourtau epourtau 4096 sep 15 11:52 gconfd-epourtau
drwx------ 2 jhabert  jhabert  4096 sep  9 10:11 gconfd-jhabert
drwx------ 2 utest    utest    4096 sep  9 14:57 gconfd-utest
.....
```

Variables et scripts

Alias :

Lancer la commande sans argument donne la liste des raccourcis de commandes connus par le shell :

```
[vberry@i17 ~]$ alias
cd      chdir !*; ls ; set prompt="! `basename $cwd` % "
em      emacs -font 10x20 !* &
l       ls -lgF
la      ls -a
ll      ls -l
lm      ls -l !* | less
ls      ls -F
m       less
mpa     mosps ax
mps     mosps
pr2reve enscript -r -2 --fancy-header -E -Preve !*
prBigFont       a2ps -p -1 -ns -nL -nH -F12 -nn !* | lpr -Preve
prll    enscript --fancy-header -E -Pll !*
```

Fichier de configuration

Pour que les variables et raccourcis qu'on définit soit conservés d'une fois sur l'autre, il est utile de les indiquer dans un des **fichiers de configuration du shell** (`.bashrc` ou `.bash_profile`) :

```
[vberry@i17 ~]$ cat .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
  . ~/.bashrc
fi

# User specific environment and startup
  programs
export PATH="$PATH:$HOME/scripts:."
export CLASSPATH="$CLASSPATH:."
```

Il est possible de désigner les paramètres indiqué au lancement d'un alias par la syntaxe suivante (propre à `bash`*) :* `$1` `$2` etc ou `$*`

Fichier de configuration

Exemple : le fichier **~/.nanorc**

```
set mouse
set autoindent
include "/usr/share/nano/c.nanorc"
```

Question

Comment faire si le fichier ~/.nanorc n'existe pas encore ?

Fichier de configuration

Exemple : **`c.nanorc`** : coloration syntaxique de codes C et C++ pour l'éditeur nano

```
syntax "c" "\.(c(c|pp|xx)?|C)$" "\.(h(h|pp|xx)?|H)
$" "\.ii?$"
color brightred "\<[A-Z_][0-9A-Z_]+\>"
color green "\<(float|double|bool|char|int|short|
long|...|inline)\>"
...
## Coloration des commentaires
color brightblue "//.*"
color brightblue start="/\*" end="\*/"
```

Scripts

- Un **script** est un fichier texte contenant des commandes respectant une certaine syntaxe
- L'**exécution** de ces commandes, les unes à la suite des autres, se fait à chaque exécution du script
- Le script lui-même s'exécute quand on tape son nom dans un terminal de commandes (!!! s'il a les droits d'exécution, voir commande **`chmod`**).
- **Intérêt** : mémoriser une série de commandes fastidieuse et pouvoir l'exécuter avec des arguments différents d'une fois sur l'autre

Exemple

```
~ > cat affiche.sh

echo -n "Nom du fichier à afficher : "
read fichier
./format.pl $fichier
more $fichier
```

Structures de ctrl pour scripts shell

Exemple du shell `bash` :

- Instructions conditionnelles :

```
if condition
   then {suite de commandes 1}
   elif condition2
   then {suite de commandes 2}
   else {suite de commandes n}
fi
```

- Dans l'expression de la condition :

  ! : négation -o : opérateur *ou* -a : opérateur *et*

- Instructions d'itération (boucles) :

```
while <condition>
do
   {suite de commandes}
done
```

Planification de jobs

Intérêts :

- ne pas mobiliser les ressources de la machine à une heure d'affluence (administration, calculs gourmands, etc)
- lancer régulièrement le même processus (ex : nettoyage du répertoire `/tmp`, sauvegarde automatique, etc).

Une seule exécution mais retardée :

```
at -f <script> <date d'exécution>
```

où le format de la date peut être spécifié de façon suivante :

- `HHMM` ou `HH:MM` pour l'heure
- `MMJJAA`, `MM/JJ/AA` ou `JJ.MM.AA` pour le jour
- la date peut aussi être de type : `now+ x unités` avec une unité en `minutes`, `hours`, `days` et `weeks`.

Autre commandes associées :

- `atq` : liste des commandes planifiées par la commandes `at`
- `atrm <iemejob>` : enlève le ieme job du planificateur

Planification de jobs

Exécution régulière

```
crontab {-l | -r | -e} <file>
```

Elle utilise un fichier dans un format particulier :

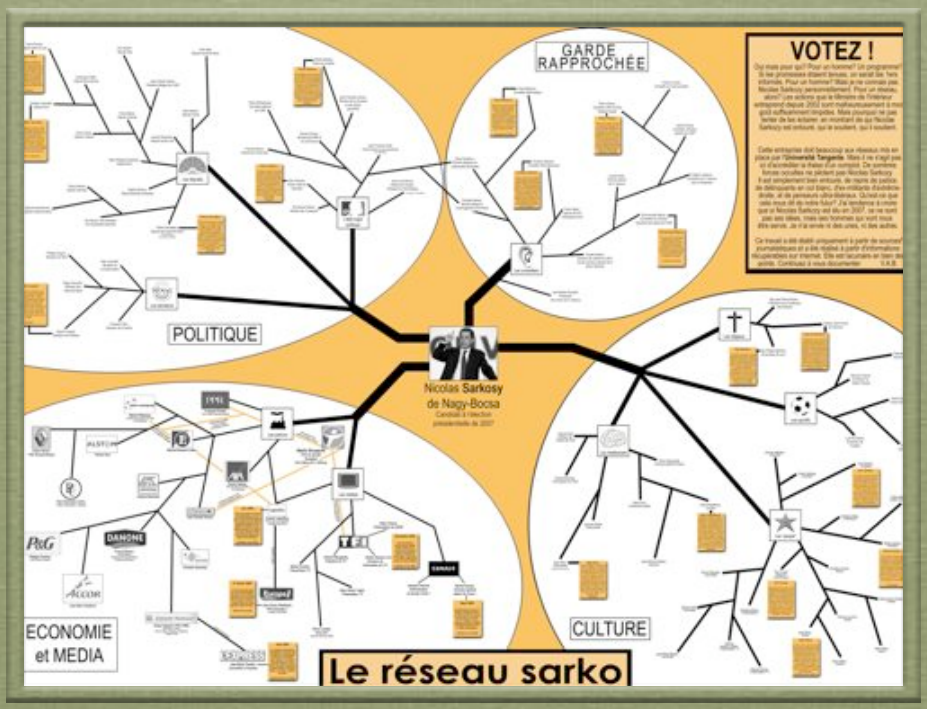
- lignes `var=valeur` pour initialiser des vars. d'environnement
- lignes de commandes pour le démon crond, définissant en 6 champs

les travaux à lancer périodiquement :

```
1) minutes 2) heures 3) jours 4) mois
5) jour de la semaine 6) tâche à exécuter
```

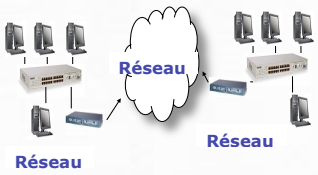
Exemple :

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/bin
MAILTO=berry # pour tte commande renvoyant une sortie
HOME=
32      7   *   *   1        tous_les_lundis_a_7h32.sh
0-59/5  *   *   *   *        toutes_les_5_minutes.sh
4       3   *   *   sat,sun  tous_les_sam_dim_a_3h04.sh
```



Notions de Réseaux

Notions de réseaux



Besoins des utilisateurs :

- des applications qui communiquent. Pour communiquer entre eux et partager des ressources communes. Ceci implique dans tous les cas un échange de données entre des applications.

Notions de réseaux

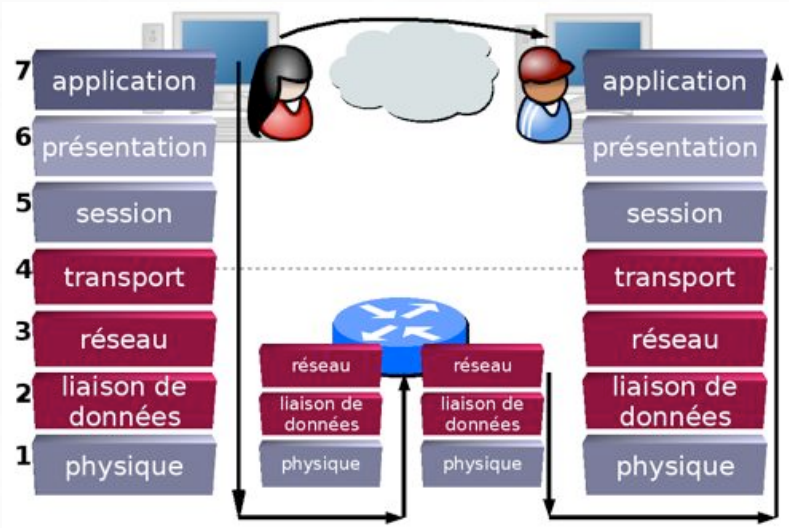
- Des **protocoles** : accords sur des règles permettant aux entités communicantes de se comprendre

Exemple : le téléphone :

1. Le protocole commence à s'appliquer lorsque la personne que l'on cherche à joindre décroche
2. la personne jointe doit dire quelque chose : «*allo*», «*bonjour*»,
3. la personne appelante répond pour démarrer la conversation;
4. en cours de discussion, le protocole impose de ne pas parler tous les deux à la fois
5. pour terminer la conversation, une des deux personnes annonce à l'autre sa volonté de finir la conversation
6. la conversation se termine quand l'un des deux raccroche

Exemples

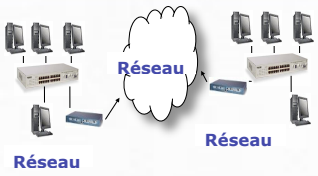
- TCP/IP est un ensemble des deux protocoles (TCP et IP) qu'un ordinateur d'un réseau utilise pour communiquer avec d'autres ordinateurs reliés au même réseau.



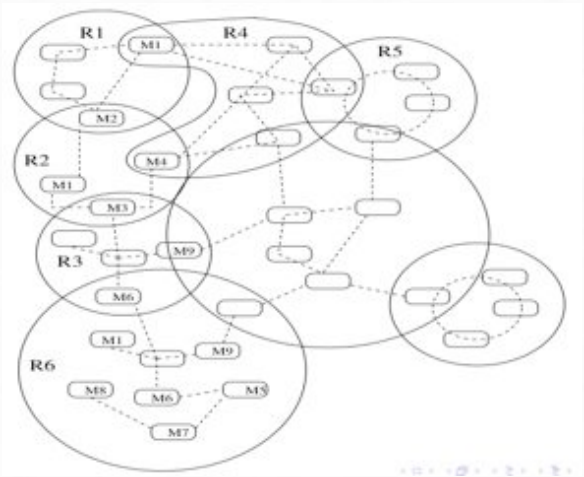
- IP est une implémentation particulière de couche 3 : "réseau" : met en oeuvre des techniques d'adressage pour faire en sorte que les paquets de donnés envoyés par tout hôte du réseau parviennent au destinataire malgré la complexité du réseau

- TCP est une implémentation particulière de couche 4 : "transport" : il s'assure que tous les paquets envoyés par l'hôte expéditeur sont reçus par le destinataire dans le bon ordre et sans doublons.

Notions de réseaux



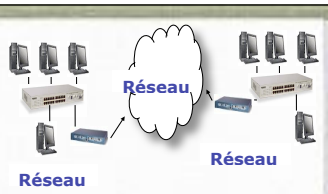
Internet = interconnexion de réseaux



Elle est assurée par différents types d'ordinateurs (simples, **hubs**, **switchs**, **routeurs**, **passerelles**,...).

Leur rôle = assurer le transfert de données : déterminer le prochain intermédiaire et lui faire suivre le paquet.

Notions de réseaux



Nommage hiérarchisé par domaine :

`nomHôte.sousdomaine.domaine.domaineRacine`

- Les noms de domaine et de racine sont gérés par des autorités
- Les noms de machines hôtes sont décidés par les administrateurs du réseau concerné, puis communiqués aux autorités

Adressage

- Les noms sont un bon moyen de désigner les hôtes, mais un très mauvais pour acheminer les paquets.
- A chaque hôte, on associe un entier de 4 octets dans le protocole IP (version IPv4).

Notions de réseaux

Commandes Unix à distance :

Connexion sur une machine distante : secure shell

```
ssh nomlogin@hôte.sousdom.dom.domRacine
```

- depuis un terminal.
- Identification réciproque : vérification de la validité de la clef de la machine distante et on indique notre mot de passe.

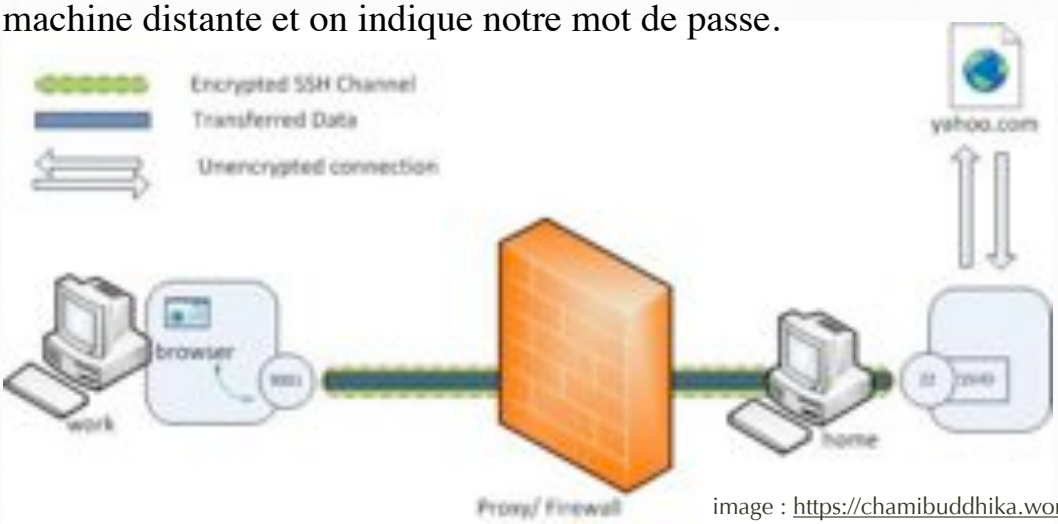


image : https://chamibuddhika.wordpress.com

Notions de réseaux

Commandes Unix % réseau :

Exemple de l'utilisateur Johnny :

```
ssh admin@hôte.sousdom.dom.domRacine
```

Question **Quel mot de passe indiquer à la connexion ?**

Exécution de commandes sur la machine distante

- On est alors connecté au Système de fichiers de la machine distante
- On peut ensuite **exécuter des commandes** sur la machine distante sans être physiquement devant.
- Exemples : se connecter à un serveur de calcul pour lancer des traitements coûteux en temps (intégration ou analyse de données sur un entrepôts de données), se connecter chez un hébergeur, ...

Notions de réseaux

Transfert de fichiers depuis/vers une machine archive :

`ftp nomserveur.site.domaine`

(si pas de compte : login `anonymous/ email`)

Permet de parcourir les fichiers distants : Commandes : `cd, put, mput, get, mget, delete, mdelete`

`scp <fichierSource> <fichierDestination>`

où chaque `désignation` a le format suivant :

`[nomlogin@][machine.site.domaine:][chemin/fichier]`

Exemple : *scp projet.tar v240.polytech.univ-montp2.fr*

Notions de réseaux

QUIZZ

Par quelles commandes éditer le fichier de configuration du SGBD Oracle situé sur la machine v240 du sous-réseau IG de Polytech ?

QUIZZ

Par quelles commandes mettre à jour toutes les pages webs de mon site *FierDetreEnIG.com,* hébergé chez OVH et dont j'édite une copie en local sur ma machine ?