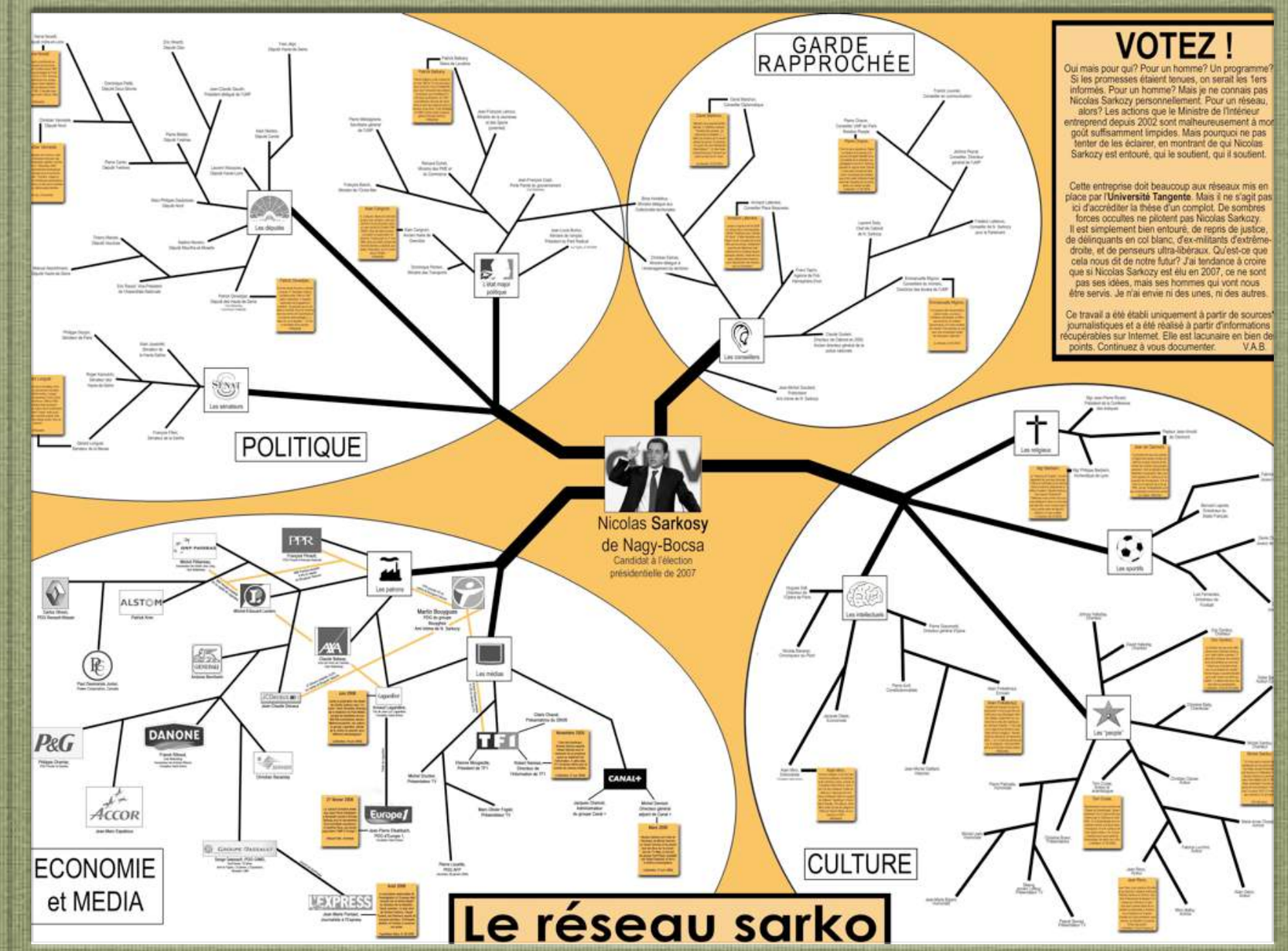
Introduction aux Systèmes d'exploitation III

V. Berry

Université Montpellier

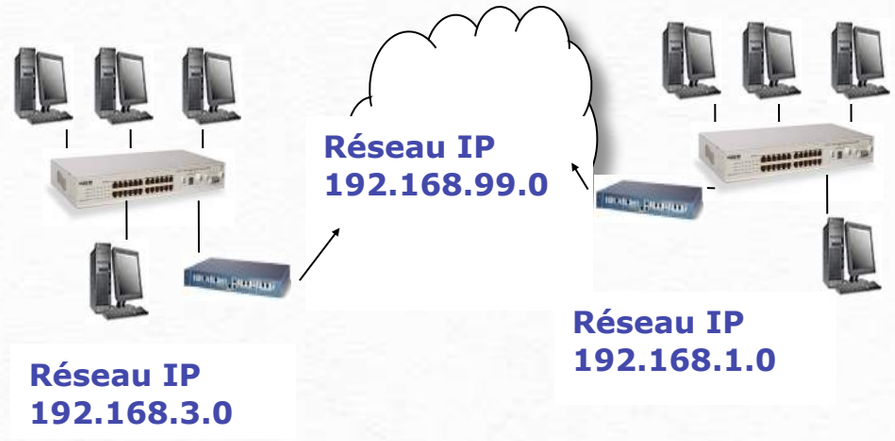
Programme

- ☑ Notion de système d'exploitation
- ☑ Architecture d'un système Unix
- ☑ Gestion des fichiers
- ☑ Un éditeur de texte
- ☐ Notions de réseau - quelques commandes
- ☐ Variables d'environnement et fichiers de configuration
- ☐ Scripts & programmation shell



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.

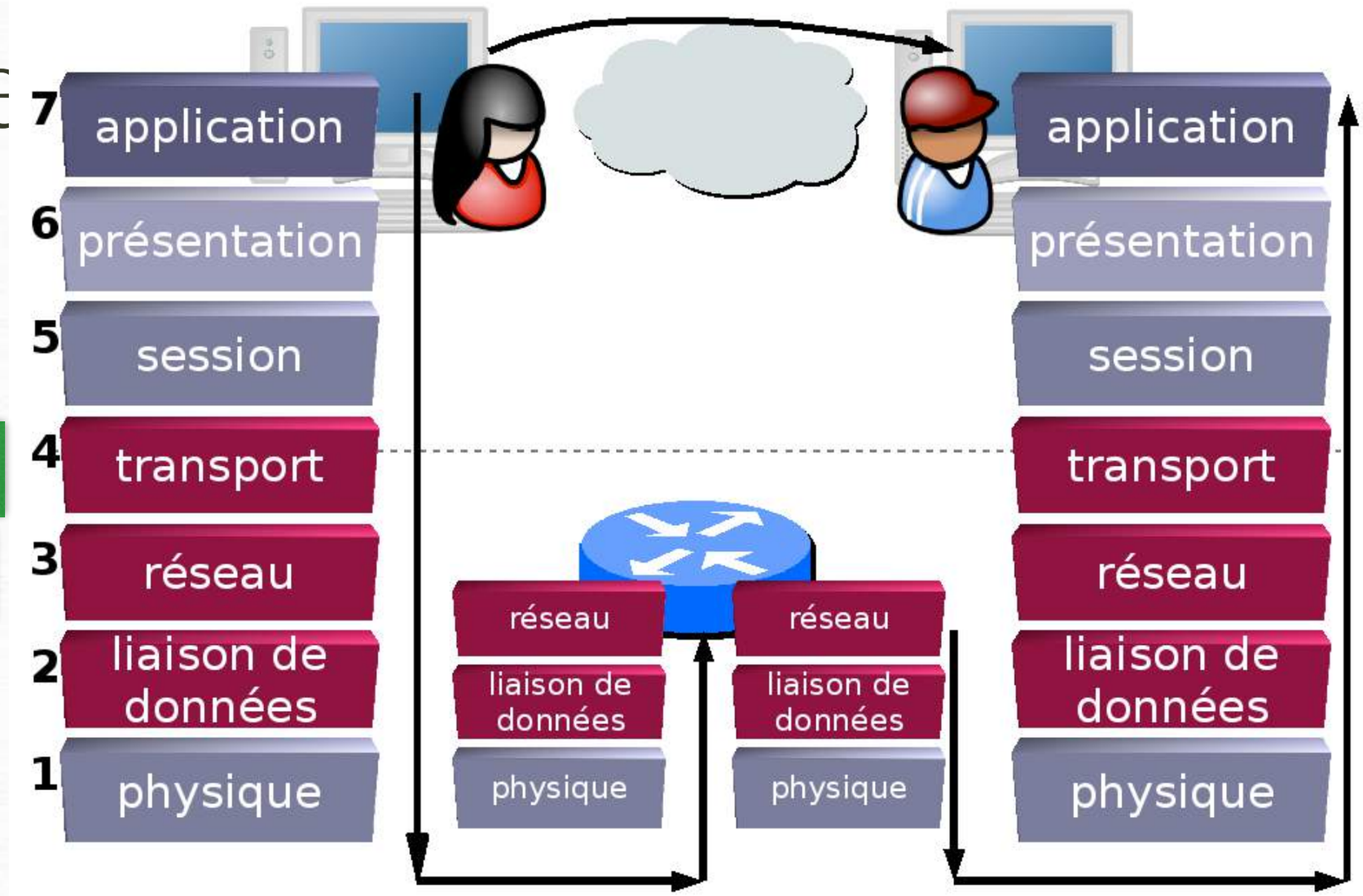
Notions de réseaux

- 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*», «*t où ?*»,
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 (surtout dans les tunnels ;)
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

Protocole



TCP

IP

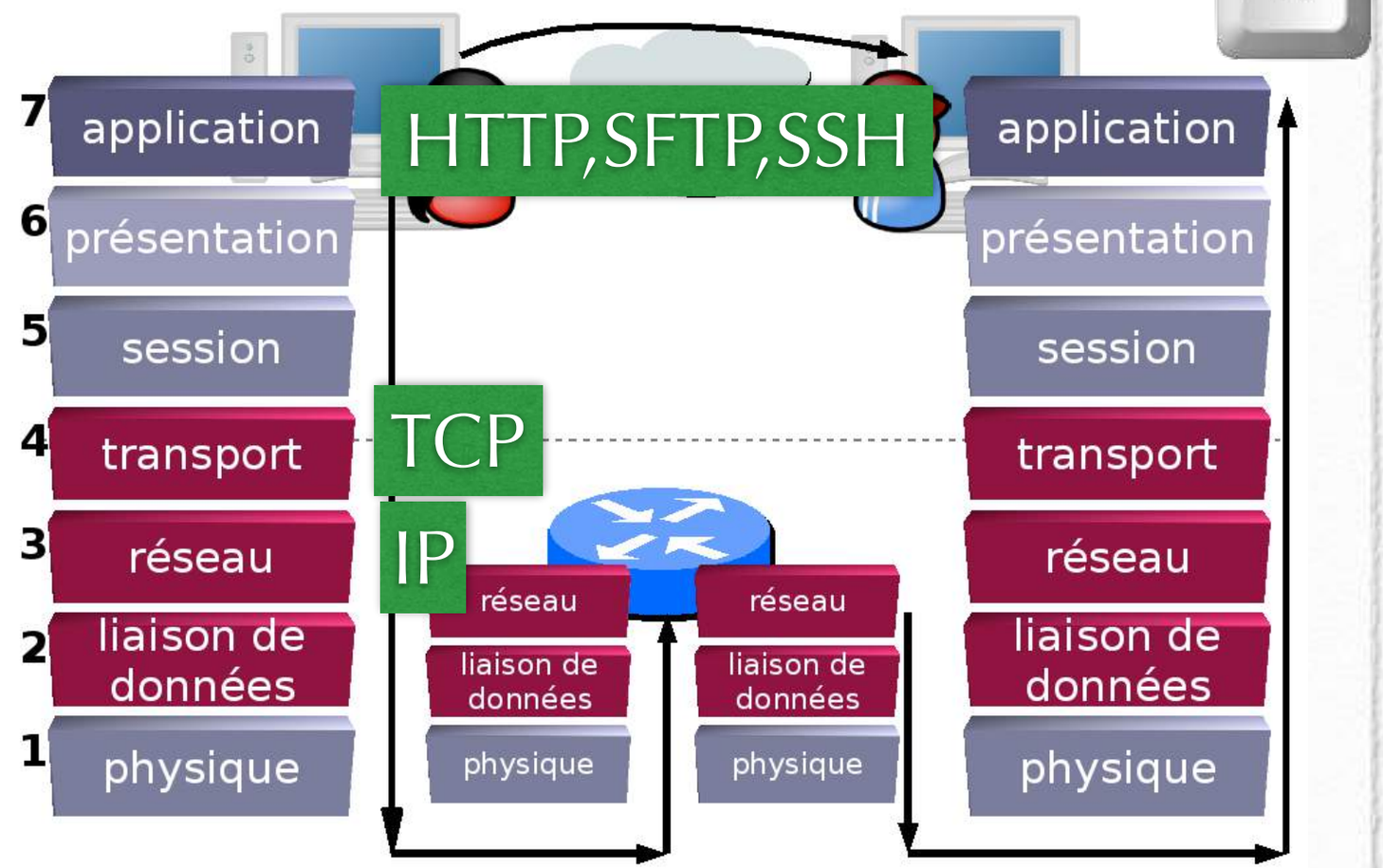
Protocole

- **IP** est un protocole de la couche "réseau" : techniques d'adressage pour que les paquets de données d'un noeud du réseau parviennent au destinataire malgré la complexité du réseau
- **TCP** est un protocole de la couche "transport" : il s'assure que tous les **paquets** de l'expéditeur soient remis au destinataire dans le bon ordre et sans doublons.

Protocoles

- HTT**P(S)** : « protocole de transfert hypertexte » = un protocole de communication client-serveur

- SFT**P** : protocole de transfert de fichiers fonctionnant au dessus de SSH (Secure Shell)



1. Récupération d'une page web en ligne de commande :
   utilisez la commande `curl` pour récupérer le contenu de la page d'accueil du site http://htmlbordel.fr/creer-une-page-html (*allô le manuel ?*)

2. Comment faire pour que le contenu de la page téléchargée s'enregistre dans un fichier page.html ?

3. Récupération de l'entête de l'échange HTTP : quelle option ajouter à `curl` pour récupérer l'entête http de la page téléchargée ?
   Utilisez grep pour savoir si cette page a voulu envoyer des cookies avec la page. Si oui, utilisez la commande grep pour savoir combien de cookies sont reçus

4. Récupération d'un fichier depuis une adresse (click droit sur une page web -> copier le lien) :
   essayez l'option **--output** de curl pour récupérer le fichier à l'adresse suivante : https://plage.igpolytech.fr/sujets/jour1/archive_1_39.tar.gz

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
- **traceroute** `nomHôte` permet d'avoir des infos sur son domaine…

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

image : https://chamibuddhika.wordpress.com

Commandes Unix à distance :

Suis-je seul au monde ? ICMP protocole de couche 3

`ping hôte(.sousdom).dom.domRacine`

1. Pouvez-vous « pinguer » les machines suivantes :
   - www.lirmm.fr , mouse.isim.intra , google.fr , google.com
2. Y a-t-il un temps de réponse significativement différent ?
3. Où sont situés ces machines ?

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 clef de la machine distante + on indique notre mot de passe / ou bien on partage une clef d'identification

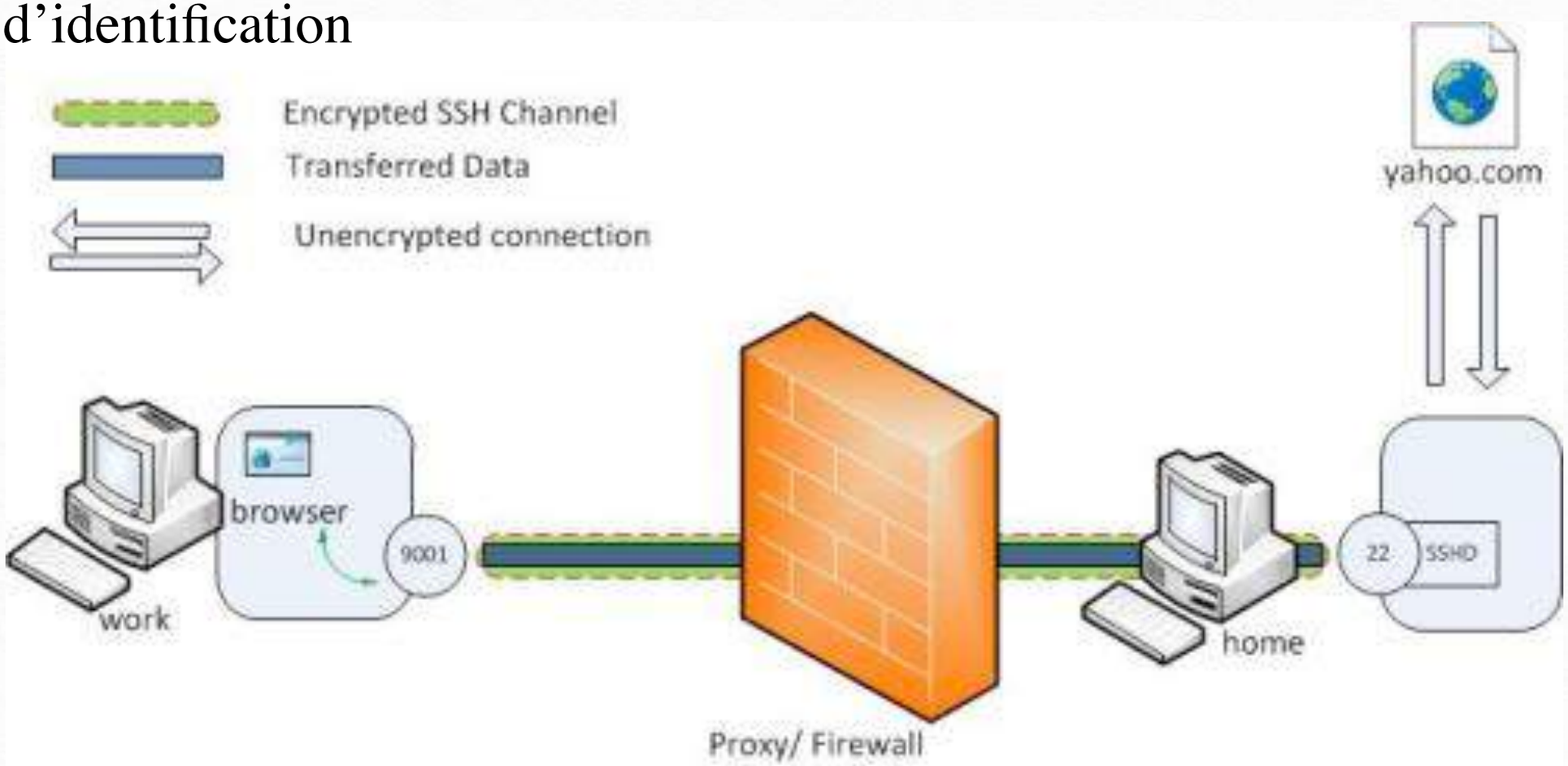


image : https://chamibuddhika.wordpress.com

Notions de réseaux

Commandes Unix % réseau :

Exemple de l'utilisateur Johnny (compte local) :

```
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 raspberry ne disposant pas d'interface clavier/écran, se connecter à un serveur de calcul pour lancer des traitements coûteux en temps, se connecter chez un hébergeur de sites web, ...

Notions de réseaux

Transfert de fichiers depuis/vers une machine archive :

`scp` `[-r] <fichSource> <fichDestination>`

où chaque **`désignation`** a le format suivant :

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

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

Question

1) **Quelle identité utilisée dans la commande ci-dessus ?**
2) **Qu'est-ce que cela suppose ?**
3) **Quel mot de passe indiquer quand scp le demande ?**

Notions de réseaux

Transfert de fichiers depuis/vers une machine serveur de fichiers :

`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`

Exemple :

- Connectez vous par ftp ou sftp au site `test.rebex.net` en utilisant l'identifiant/mdp : *demo*/*password*
- Récupérez sur votre machine une copie du fichier readme.txt qui se situe dans le dossier pub/example (pas celui situé à la racine du serveur)

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 ?

Variables , fichiers de configuration & scripts

Variables d'environnement

Des informations sur le S.E. et l'environnement de travail sont stockées dans des **variables**

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**)

Bash : pour définir de nouvelles variables : **nom=valeur**

Sans espaces !

et `echo` permet de voir la valeur d'une variable : `echo $nom`

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

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

Variables d'environnement

Informations 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`,...)

Expérimentez sur votre poste de travail :

- que vaut la variable $HOME ?
- Est-ce que $PWD change quand vous changez de dossier avec la commande cd ?

Ces variables communiquent des informations aux programmes

Exemple : qu'obtenez-vous ici ?

```
$ python
>>> import os
>>> print os.environ['HOME']
```

*Aide : **CTRL+D** pour sortir ensuite de l'interpréteur Python*

Variables d'environnement

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
```

QUIZZ

Comment obtenir le prompt suivant ?

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

Variables d'environnement

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 d'environnement

Alias :

Lancer la commande **alias** 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
...
```

Fichiers de configuration

- Quand on lance un programme il a accès aux variables d'environnement
- mais on peut aussi lui indiquer informations **spécifiques** dans un **fichier de configuration**

✓ ne pas polluer **l'environnement** utilisé par **tous** les programmes

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

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

Question

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

Fichiers de configuration

Pour que les variables et raccourcis soient conservés d'une fois sur l'autre, on les déclare dans un **fichier de configuration du shell** (`.bashrc` ou `.bash_profile`) :

```
[vberry@i17 ~]$ cat .bash_profile
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
  source ~/.bashrc
fi

# User specific 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 $*

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 on a les droits d'exécution !)
- **Intérêt** : mémoriser une série de commandes fastidieuse et pouvoir l'exécuter avec des arguments différents d'une fois sur l'autre

Exemple

```
$ cat ./affiche-joliment.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 (entre crochets avec espaces)
   then {suite de commandes 1}
   elif condition2
   then {suite de commandes 2}
   else {suite de commandes n}
fi
```

`if [ -e tmp ]`

- Dans l'expression de la condition :

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

`-e $truc` : ce fichier/dossier existe

`-f $truc` : ce truc est un fichier `-d $truc` : ce truc est un dossier

Structures de ctrl pour scripts shell

Exemple du shell `bash` :

- Instructions d'itération (boucles) :

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

- Pour chaque fichier de type .txt :

```
for f in *.txt
  do commande $f
done
```

Exemple : nom des fichiers de type txt

```
for f in *.txt
  do echo ${f%.txt};
done
```

Action !

- Assistez votre enseignant (oui, pour une fois, ça change !)
- Il veut faire un **script** qui permet de sauvegarder le contenu de son dossier ~/Documents dans une archive ~/SVG/mesDocs-11-09-18.13h21.tar.gz

Ecrivez le script en suivant cette démarche :

1. On crée un dossier pour placer les sauvegardes
2. On lance nano avec le nom du fichier à éditer
3. Dans un autre terminal on demande de l'aide sur la commande `date`
4. On fait quelques essais dans le terminal

Votre script de sauvegarde :

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).

1 - 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

2 - Exécution multiples (régulières)

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

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 :

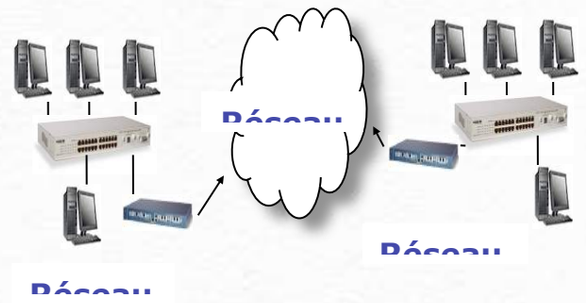
`1) minutes 2) heures 3) jours 4) mois`
`5) jour de la semaine 6) tâche à exécuter`

Exemple :

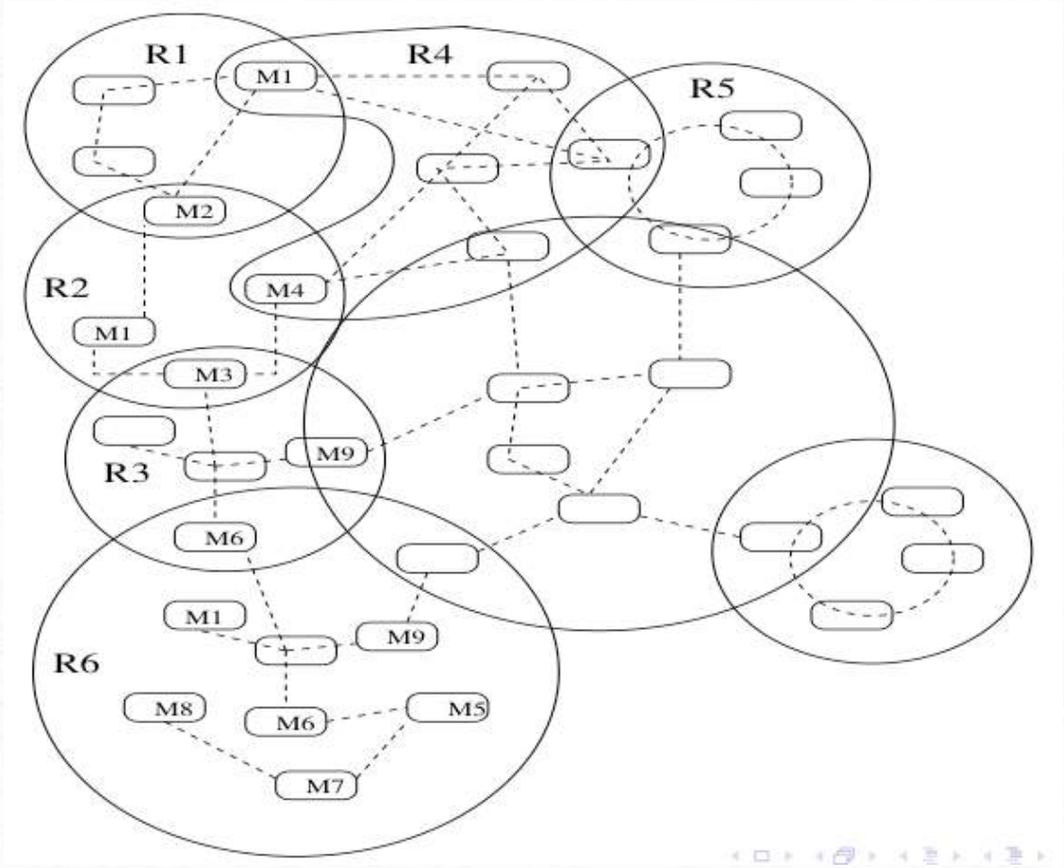
```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/bin
MAILTO=berry # pour tte commande renvoyant une sortie
HOME=
32      7   *   *   1         tous_les_lundis_a_7h32.sh
0-59/5  *   *   *   *         toutes_les_5_minutes.sh
4       3   *   *   sat,sun   tous les sam dim a 3h04.sh
```

…et pour quelques diapo$ de plus

Notions de réseaux



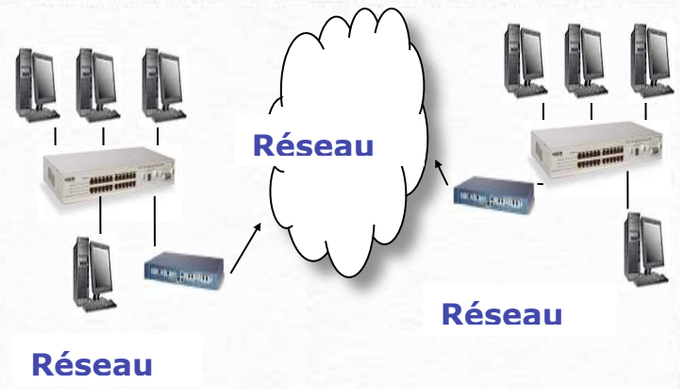
Internet = interconnexion de réseaux



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



Plusieurs topologies de réseaux :

étoile

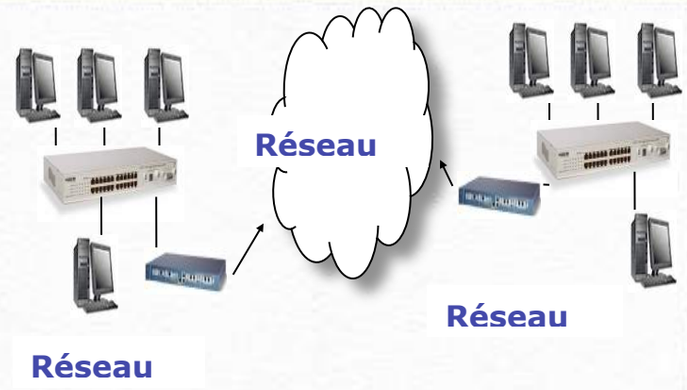
bus

anneau

arbre

QUIZZ **Avantage/Inconvénient de chaque topologie ?**

Notions de réseaux



Serveur de noms (DNS)

- **Problème** : connaissant le nom d'un hôte, trouver son adresse.
- **Principe** : une base de données distribuée où chaque admin met à jour les données relatives à son réseau. Il met en place une application appelée serveur de noms, qui répond à chaque requête contenant un nom, par l'adresse correspondante
- Quand un hôte (`tp3-pc13.polytech.univ-montp2.fr`) demande à contacter un hôte non-local (`c12.pentagon.afis.osd.mil`), son dns (`ns.polytech.univ-montp2.fr`) demande directement à un serveur racine, qui connait le domaine racine (`mil`), qui transmet au domaine concerné (`osd.mil`), et ainsi de suite, jusqu'à atteindre l'hôte cherché.