

Internet et Outils
L1/IO2 2007 - 2008
S2-IO2
REVISIONS

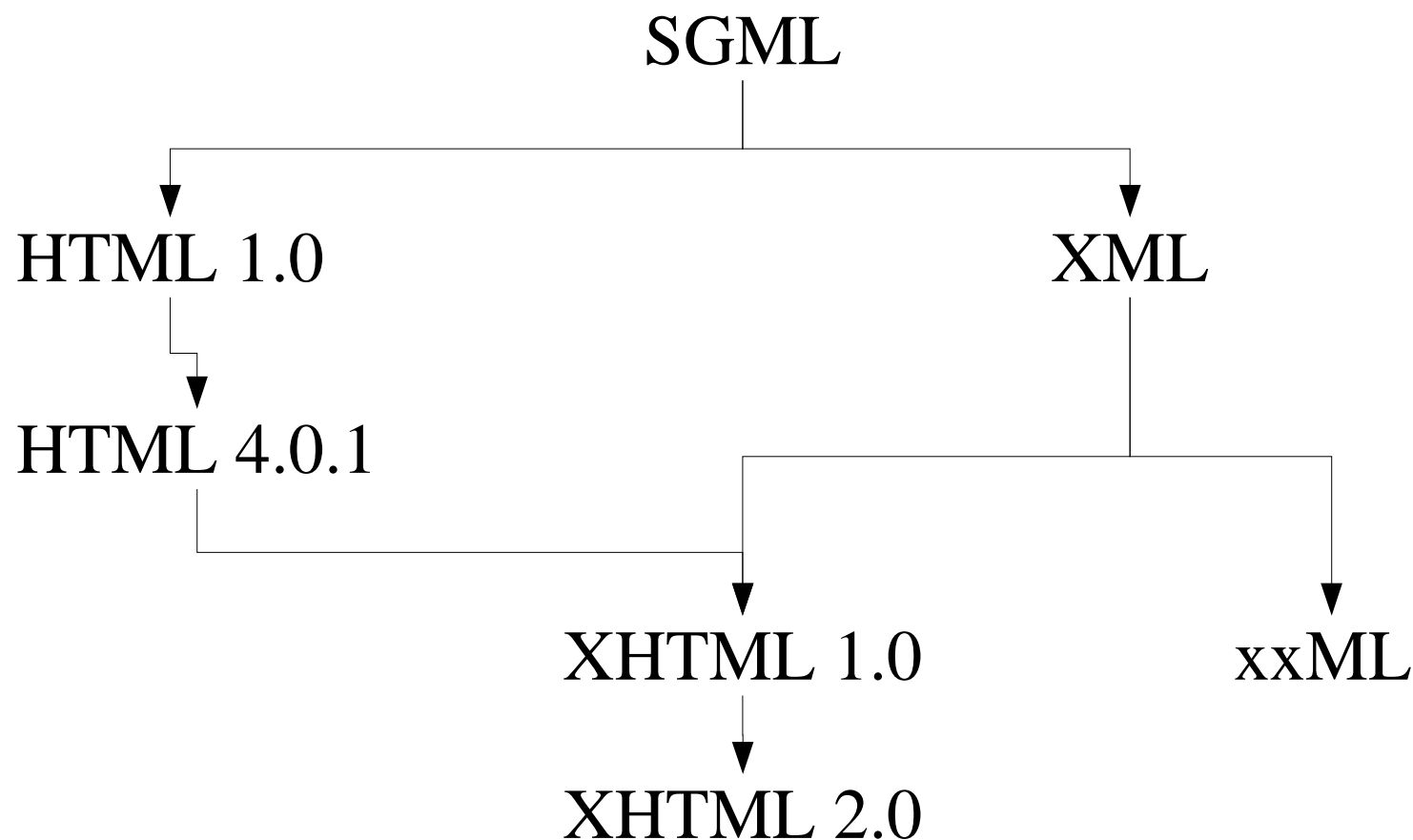
François Armand
`armand@informatique.univ-paris-diderot.fr`

Révisions: Plan

- [X]HTML
- CSS
- PHP
- SQL
- Réseaux

Une famille de langages

(Vision très simplifiée)



Markup Language: Principes

- Enrichir le document (le texte) avec des informations complémentaires au document:
 - Informations sémantiques
 - Mise en forme,
- [X]HTML, permet de distinguer les directives de mise en forme du texte, du contenu lui-même:
 - Utilisation de « balises » (Markup Language)
 - **Indiquer la fonction**: titre, paragraphe, citation, légende...
 - **Indiquer la présentation**: couleur, taille, police de caractères,...

Markup Language: Principes

- Langages de description de données
- Dans le cas de [X] HTML:
 - Interprétation par l'outil affichant le document: le navigateur.

Formes des Balises XHTML

- Forme générale 1:

<cont> texte du document **</cont>**

« Conteneur »: encadre une partie du texte (élément) et affecte les caractéristiques de cet élément.

ex : **<i>**Italique**</i>** => *Italique*

- Forme générale 2:

<bal /> ou **<bal/>**

N'affecte pas le texte du document : élément vide

Ex: **<hr />** => « Horizontal ruler »

- Règles de fermeture

Balises XHTML

```
<html>  
  <head>  
    <title> Exemple simple de XHTML </title>  
  </head>  
  <body bgcolor="blue">  
    <center><h1> Paragraphe N°1 </h1></center>  
    <hr />  
    Essais de variations: <b>Gras</b>  
    <i>Italique</i>  
  </body>  
</html>
```

Attributs de balise

- Les balises peuvent être modifiées par des attributs
- `<bal att1="valeur" att2="valeur_2"> ... </bal>`
 - Ex: `<abbr title="respectivement">resp.</abbr>`
- Quelques attributs « standards »:
 - class, id, style, title
 - dir (ltr, rtl), lang, xml:lang

Listes

• ...

- Liste non numérotée : marquée par une puce

Premier élément de la liste

Deuxième élément de la liste

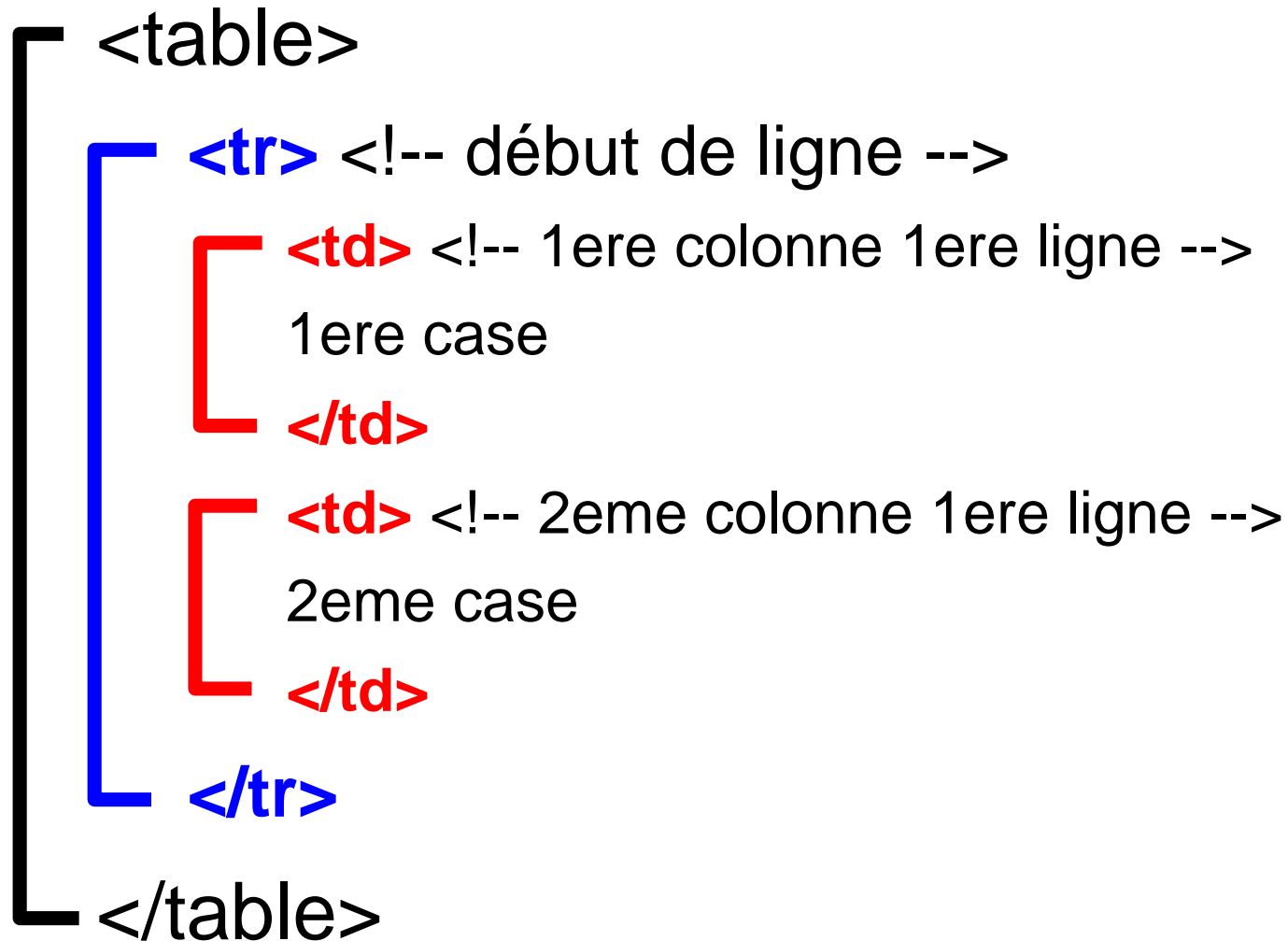
• ...

- Liste numérotée

Premier élément de la liste

Deuxième élément de la liste

Tableau en XHTML



Liens HyperTexte

- Permet de rendre un document « interactif »
 - Une zone du document (image, texte,...) sensible aux opérations de l'utilisateur (clic)
 - Est associée à une ressource sur le web
- Le navigateur télécharge cette ressource quand l'utilisateur sélectionne (clique sur) la zone sensible (hypertexte ou hypermedia)
- La ressource est désignée par un lien (URL)

Liens

```
<a href="http://www.w3.org/MarkUp/Guide">
```

```
Tutoriel HTML du W3C</a>
```

- On peut aussi référencer des points précis

- dans le document courant (doc1):

```
<a name="chapitre_1"> Chapitre 1</a>
```

```
<a href="chapitre_1">Aller au chapitre 1 direct!</a>
```

- dans un autre document:

```
<a href="http://www.foo.org/index.html#chapitre_1">
```

Uniform Resource Locator

- **Forme générale: URL**
 - Uniform Resource Locator (IETF RFC 1738 -1994)
 - Parfois appelé "Universal" au lieu de Uniform
- **<schéma>:<partie spécifique au schéma>**
- **schémas usuels:**
 - http, file, ftp, mailto, telnet..
- **"Nouveau": URI Uniform Resource Identifier**
 - IETF STD 0066 ou RFC 3986

Uniform Resource Locator

- Partie spécifique usuelle sur Internet
- `//<user>:<password>@<host>:<port>/<url-path>`
Chemin d'accès:
 - `<url-path>` dépend du "schéma",
 - Avec HTTP: chemin d'accès au fichier relatif à un point de départ connu du serveur HTTP sur la machine cible
 - **HTTP: HyperText Transfer Protocol**

URL

- Avec http, on peut faire suivre le "url-path" par
 - `<urlpath>?requête#fragment`
- Syntaxe de la requête:
 - `param=value1&2ndparam=value2`

Quelques différences (1/2)

XHTML

HTML

Balises minuscules : XML sensible à la casse maj/min	Balises indifféremment maj/min
Balises fermées dans l'ordre	Balises pas forcément fermées, pas forcément en ordre
Balises vides: <bal />	Balises vides <bal>
Attributs "quotés" et non réduits	Attributs pas forcément "quotés" et possiblement réduits

Quelques différences (2/2)

XHTML

HTML

<code> texte en gras </code>	<code> texte en gras </code>
<code> <i> txt </i></code>	<code> <i> txt </i></code>
<code>
</code>	<code>
</code>
<code>==> en pratique
</code>	
<code><body bgcolor="red"></code>	<code><body bgcolor=red></code>
<code><input checked= "checked"></code>	<code><input checked></code>

De plus, les documents XHTML doivent être bien formés.

XHTML bien formé

- Éléments obligatoires d'un document XHTML

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML  
1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-  
transitional.dtd">
```

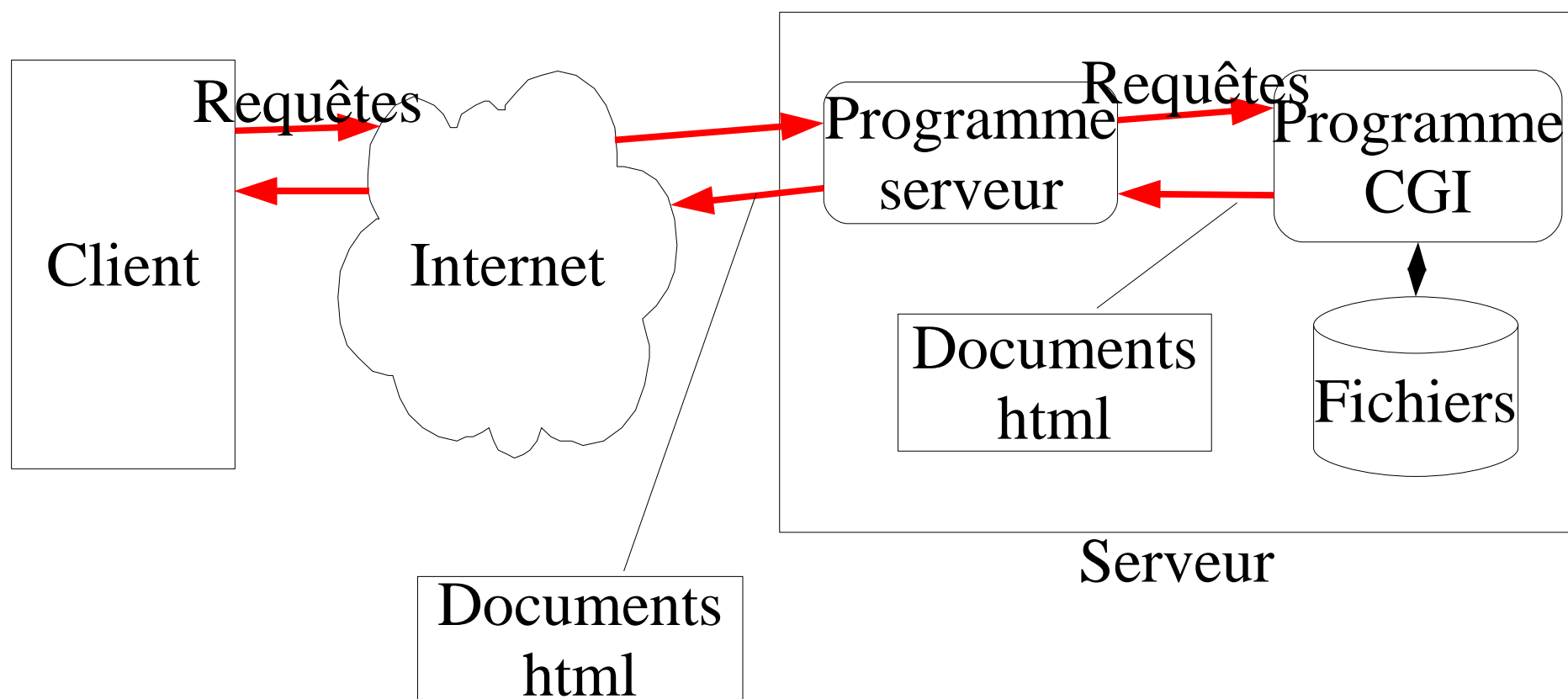
```
<html>
```

```
  <head> <title>Mon titre</title> </head>
```

```
<body>Mon texte </body>
```

```
</html>
```

Architecture CGI



Formulaires

- `<form>...</form>`
 - Peut inclure toute balise html et des balises appropriées pour la saisie
- 3 attributs principaux:
 - action: url du programme à exécuter sur le serveur
 - method: mode de transmission des paramètres
(get ou post)
 - enctype: type d'encodage des données du formulaire
 - application/x-www-form-urlencoded: nom=valeur[&nom=valeur]*
 - multipart/form-data: quand on transmet des fichiers avec la requête

Source (partiel) du formulaire

```
<form action="tst_action.php" method="get">
```

```
Prénom <input type="text" name="prenom" size="20" />
```

```
Mot de passe <input type="password" name="passwd" size="16" />
```

```
<br />PAYS<br />
```

```
FR<input type="radio" name="pays" value="fr" checked />
```

```
UK<input type="radio" name="pays" value="uk" />
```

```
<br/>
```

```
Theatre <input type="checkbox" name="loisir" value="T" />
```

```
<input type="submit" value="Envoyé c'est pesé!" name="go" />
```

```
<input type="reset" value="On recommence tout" />
```

```
</form>
```

Chaîne envoyée (méthode get)

xxxx/exemple_action?prenom=jules&passwd=secret&pays=be&loisir=C&loisir=T&age=5&go=Envoyer

- A charge pour le programme CGI (C, script, java...) de décoder cette chaîne pour retrouver les arguments et pour générer en retour le document html approprié.

Révisions: Plan

- [X]HTML
- CSS
- PHP
- SQL
- Réseaux

Forme Générale des Règles

- sélecteur [,sélecteur]* {propriété: valeur;
[propriété: valeur;]*}
- Où sélecteur est :
 - Balise,
 - Classe,
 - Identificateur...
- Exemples de propriété:
 - background-color, font-size, color, text-align

CSS Exemples

```
a, h1, h2, h3 {color:#CA0000}
```

```
/* couleur rouge pour ancrs et entêtes */
```

```
caption {font-size:large; color:#CA0000}
```

```
/* Deux attributs pour tous les titres de tableaux */
```

```
tr.a0 {background-color: blue}
```

```
tr.a1 {background-color: yellow}
```

```
/* a0 et a1 définis comme attribut class dans le  
html */
```

```
/* commentaires pour feuilles de styles */
```

Attribut class Exemple

```
<table>
```

```
<caption><b>Quelques films<b></caption>
```

```
<tr><th>Titre</th><th>Année</th><th>Réalisateur</th></tr>
```

```
<tr class="a0"> <td>Alien</td><td>1979</td><td>Scott Rileley</td></tr>
```

```
<tr class="a1"><td>Vertigo</td><td>1958</td><td>Hitchcock  
Alfred</td></tr>
```

```
<tr class="a0"> <td>Psychose</td><td>1960</td><td>Hitchcock  
Alfred</td></tr>
```

```
<tr class="a1"> <td>Titanic</td><td>1997</td><td>Cameron  
James</td></tr>
```

```
</table>
```

Exemple Attribut class

The screenshot shows a browser window titled "Exemple simple de tableau XHTML - Mozilla Firefox". The page content includes the heading "Une table" and a sub-heading "Quelques films". Below this is a table with three columns: "Titre", "Année", and "Réalisateur". The table contains four rows of data. Red arrows point from labels "tr.a0" and "tr.a1" to the first and second rows of the table, respectively.

Titre	Année	Réalisateur
Alien	1979	Scott Rildey
Vertigo	1958	Hitchcock Alfred
Psychose	1960	Hitchcock Alfred
Titatnic	1997	Cameron James

Référence à une CSS

```
<head>
```

```
<title>
```

Exemple simple de tableau XHTML

```
</title>
```

```
<!-- Au sein de la section head -->
```

```
<link rel="stylesheet" href="film.css"  
type="text/css" />
```

```
</head>
```

Style interne

```
<head>
```

```
<title>
```

Exemple simple de tableau XHTML

```
</title>
```

```
<!-- Au sein de la section head -->
```

```
<style type="text/css">
```

```
h1 {text-align:center; background-color:yellow}
```

```
</style>
```

```
</head>
```

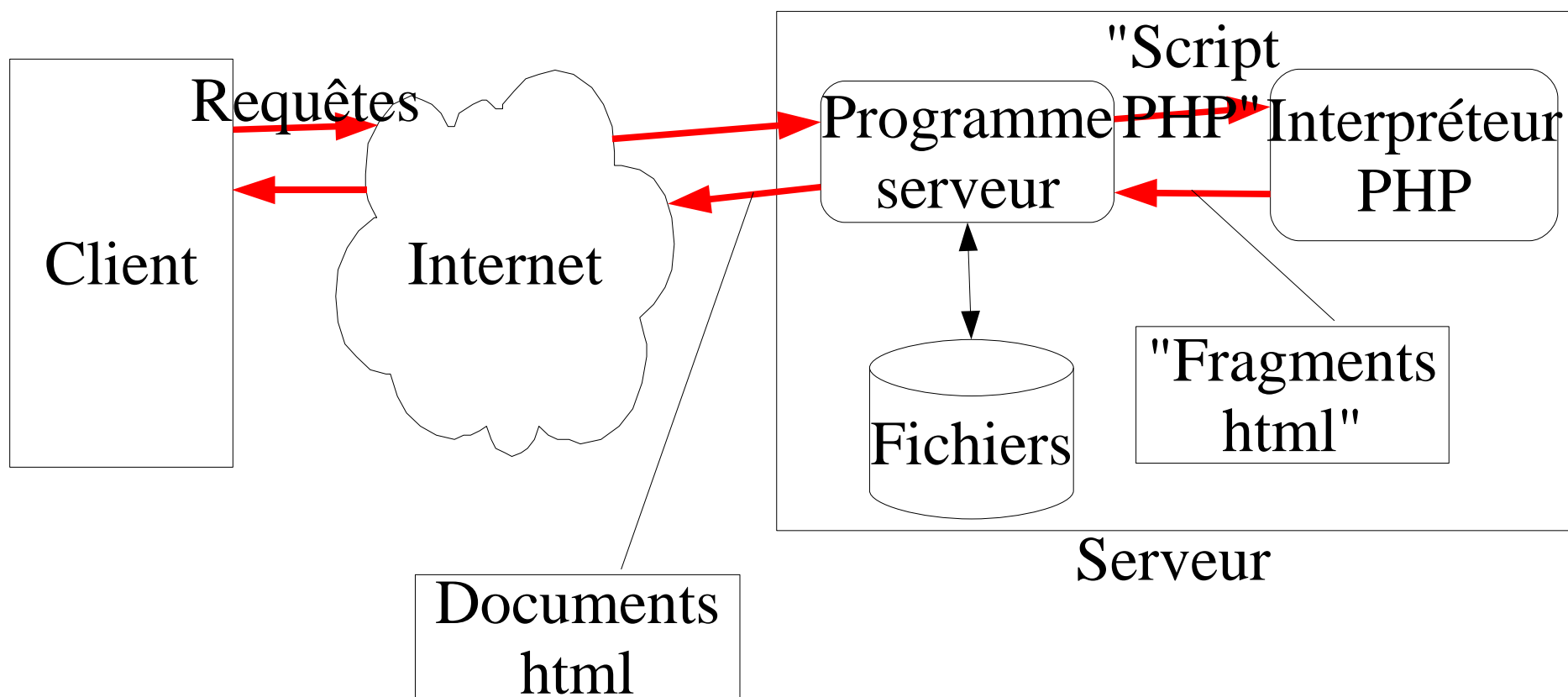
Valider un document XHTML

- Site du W3C
 - <http://validator.w3.org/>
- Outil de "nettoyage" des documents HTML
 - tidy

Révisions: Plan

- [X]HTML
- CSS
- **PHP**
- SQL
- Réseaux

Architecture CGI



Rôle du serveur

- Avec du [x]html, le serveur se "contente" de trouver le fichier contenant le document correspondant à l'URL et de le renvoyer
- Avec PHP, le serveur doit d'abord analyser les documents pour:
 - trouver toutes les occurrences de `<?php... ?>`
 - faire interpréter les commandes php par un interpréteur php
 - Substituer les chaînes `<?php... ?>` par le résultat de l'interprétation

Chaînes

- Comprises entre
 - Guillemets: "abcd \" ef ' g \$var "
 - => substitution de \$var par sa valeur
 - Apostrophes: 'abcd \' ef " g \$var '
 - => pas de substitution de \$var
- Concaténées par .
 - "abc"."abc" => "abcabc"

Variables PHP

- Noms:
 - Commencent par \$
 - Longueur quelconque
 - Composés de lettres, chiffres, _
 - Lettre ou _ après le \$
 - Majuscules et Minuscules ne sont pas identiques
- Ex:
 - \$ma_var , \$Ma_Var, \$mon_1er_nom, \$_jyvai

Variables

- En PHP une variable peut-être:
 - définie avec une valeur, `$var="toto";`
 - définie et vide, `$var=""`;
 - non définie
- Types des variables:
 - Entier: `$note=1728; // Bravo c'est excellent!`
 - Nombre à virgule flottante: `$temp=37.2; // Le matin`
 - Chaîne de caractères:
 - `$str ="1780" //chaîne pas entier!`
 - Booléen: `$condition=TRUE; $condition=FALSE;`

Vérifier le contenu d'une variable

- `isset($var)` // Vrai si définie, même vide
- `empty($var)` // Vrai si vide, 0, ou pas définie
- `is_int($var)` // Vrai si type entier
- `is_float($var)`
- `is_numeric($str)` // Vrai si chaîne numérique: 0..9
- `is_string($str)`
- `ereg("motif", $chaine)`

Classe: Exemple

```
class Utilisateur {  
  var $Nom, $Prenom,  
      $Adresse, $Email,  
      $Tel, $Pays,  
      $Pseudo,  
      $Password, // Pas besoin du double  
      $Age;  
};
```



Tableaux

- Créer avec "index":
 - `$Films[0]="Shrek";`
 - `$Films[1]="Shrek 2";`
 - Pas d'obligation de commencer à 0
- Créer avec "clefs":
 - `$capitale['FR']="Paris";`
 - `$capitale['UK']="Londres";`
 - `$capitale['DE']="Berlin";`

Tableaux

- Créer sans index explicite:
 - `$Livre[]="Les Misérables";`
 - `$Livre[]="Harry Potter";`
 - `$Livre[]="Le seigneur des anneaux";`
- `echo "Livre[0]";`
Les Misérables
- `$Livre= array ("Les Misérables", "Harry Potter", "Le seigneur des anneaux");`

Tableaux

- `$rue= array(4 => "Rue de Lapp", "Rue de la Paix");`
- `echo $rue[5]`
 - Rue de la Paix
- `$annees= range(2001, 2038);`
- On peut aussi créer des tableaux multi-dimensionnels
 - `$Notes['Jules'] ['Exo 1']; $Notes['Jules'] ['Exo 2'];`
 - `$Notes['Jim'] ['Exo 1']; $Notes['Jim'] ['Exo 2'];`

Tableau: Parcourir

```
foreach($Livre as $index => $Titre)
```

```
{  
    echo "Livre Numéro ",$index," : $Titre<br/>";  
};
```

Livre Numéro 0 : Les Misérables

Livre Numéro 1 : Harry Potter

Livre Numéro 2 : Le seigneur des anneaux

Tableau: Parcourir

- Manuellement:

- `current($tableau);`
- `next($tableau); previous($tableau)`
- `end($tableau); reset($tableau)`

- Exemple:

```
reset($Livre);
```

```
echo "current($Livre)"; //=> Les Misérables
```

```
next($Livre);
```

```
echo "current($Livre)"; // => Harry Potter
```

Tableau

- Taille:
 - `$size= count($Livre);`
 - `$size= sizeof($Livre);`
- On peut aussi:
 - Convertir un tableau en chaîne et réciproquement
 - `$Tableau=explode("x", $str) // x est un séparateur`
 - `$str=implode("x",$Tableau);`

Tableaux prédéfinis

- **\$_REQUEST** (et /ou **\$_POST** **\$_GET**)
 - Contient les variables renvoyées par POST ou GET depuis un formulaire HTML
 - `<input type="text" size="20" name="prenom" />`
 - paramètre pour PHP: `$_REQUEST['prenom']`
- **\$_SERVER**:
 - Informations relatives au serveur
 - Ex : `$_SERVER["HTTP_USER_AGENT"]`
 - Ex: `$_SERVER["SERVER_NAME"]`

Contrôle du flux d'exécution

```
if (condition) {  
    bloc;  
} else {  
    bloc;  
}
```

```
switch ($var) {  
    case Valeur1: bloc; break;  
    case Valeur2: bloc; break;  
    default: bloc;  
}
```

Contrôle du flux d'exécution

- Conditions:
 - $\$v1 == \$v2$ // les 2 valeurs sont-elles égales?
 - $\$v1 === \$v2$ // même valeur et même type?
 - $\$v1 != \$v2$ // valeur différentes?
 - $\$v1 !== \$v2$ // valeur ou type différents?
 - $>$, $<$, $>=$, $<=$

Rappels: Fonction

- Définition de fonction

```
function verif_champ($valeur, $champ, $liste)
{
    if (empty($valeur)) {
        if (!empty($liste)) {
            $liste = $liste.", ";
        }
        $liste = $liste.$champ;
    }
    return $liste;
}
```

- Appel (utilisation) de la fonction:

```
$liste = verif_champ($email, "E-mail", $ChampsManquants);
$liste = verif_champ($pays, "Pays", $ChampsManquants);
```

Révisions: Plan

- [X]HTML
- CSS
- PHP
- **SQL**
- Réseaux

Création d'une table

```
mysql> CREATE TABLE FilmSimple
```

```
-> (Titre          VARCHAR (30) ,
```

```
->   Année         INTEGER ,
```

```
->   Nom           VARCHAR (30) ,
```

```
->   Prenom       VARCHAR (30) ,
```

```
->   Naissance    INTEGER
```

```
-> ) ;
```

```
Query OK, 0 rows affected (0.05 sec)
```

Description d'une table

```
mysql> DESC FilmSimple;
```

Field	Type	Null	Key	Def	Xtr
Titre	varchar(30)	YES		NULL	
Annee	int(11)	YES		NULL	
NOM	varchar(30)	YES		NULL	
Prenom	varchar(30)	YES		NULL	
Naiss.	int(11)	YES		NULL	

```
5 rows in set (0.00 sec)
```

Insérer des données

- Note: on peut ne saisir que quelques attributs, et dans un ordre différent de celui défini pour la table.

```
mysql> INSERT INTO FilmSimple  
-> (Nom, Titre)  
-> VALUES ('Allen',  
->          'Match Point'  
-> );
```

```
Query OK, 1 row affected (0.00 sec)
```

Interroger la table

```
mysql> SELECT * FROM FilmSimple;
```

```
+-----+-----+-----+-----+-----+
|Titre   |Annee  | NOM   |Prenom |Nais   |
+-----+-----+-----+-----+-----+
|Alien   |1979   | Scott|Ridley | 1943  |
+-----+-----+-----+-----+-----+
9 rows in set (0.00 sec)
```

- Résultat tronqué pour la présentation

Interroger la table

```
mysql> SELECT Titre, Annee FROM  
FilmSimple;
```

```
+-----+-----+
```

```
|Titre    |Annee| Résultat tronqué!
```

```
+-----+-----+
```

```
|Alien    |1979 |
```

```
+-----+-----+
```

```
9 rows in set (0.00 sec)
```

- On peut sélectionner les colonnes de son choix

Interroger la table

```
mysql> SELECT Titre, Annee  
       FROM FilmSimple  
       WHERE Titre='Vertigo'  
       OR Prenom='Alfred'  
       OR (Annee >= 1980  
          AND Annee < 2000);
```

- La clause "WHERE" permet de définir un ensemble de conditions qui doivent être vérifiées par les lignes retenues.

Conditions Clause WHERE

- = égalité
- <> différence
- < inférieur, <= inférieur ou égal
- > supérieur, >= supérieur ou égal
- BETWEEN a AND b
- LIKE modèle
- On peut spécifier le contraire avec NOT
- On peut combiner les clauses avec AND / OR

Interroger la table: ORDER BY

```
mysql>SELECT Titre, Annee FROM FilmSimple
WHERE Titre BETWEEN 'Psychose' AND
'Titanic' ORDER BY Titre;
```

Titre	Annee
Pulp Fiction	1995
Psychose	1960
Sacrifice	1986
Titanic	1997

Supprimer des données

```
mysql> DELETE FROM FilmSimple WHERE  
Annee <= 1960;
```

- Détruit tous les films dont l'année est inférieure ou égale à 1960.
- Fonctionnement par défaut de MySQL:
 - Données perdues
- Si mode transactionnel (optionnel) supporté alors possibilité de "rollback" (voir cours suivants)

Modifier des données

```
mysql> UPDATE FilmSimple  
      SET Nom='Wu', Prenom='Yusen'  
      WHERE Nom='Woo';
```

- Met à jour les champs 'Nom' et 'Prenom' de toutes les lignes sélectionnées par la clause 'WHERE'.
- Pour vérifier les lignes affectées par un ordre 'UPDATE' ou 'DELETE', faire préalablement un ordre 'SELECT' avec la même clause 'WHERE'.

Création d'une table

```
mysql> CREATE TABLE CompteBancaire
```

```
    -> (Nom          VARCHAR(30) ,
```

```
    ->   Avoir       INTEGER ,
```

```
    ->   Quand       DATE
```

```
    -> ) ;
```

```
Query OK, 0 rows affected (0.05 sec)
```

Modifier des Données

UPDATE Compte_Bancaire

SET Avoir=Avoir-50,

Quand=CURDATE()

WHERE Nom='Jules';

Quand='AAAA-MM-JJ'

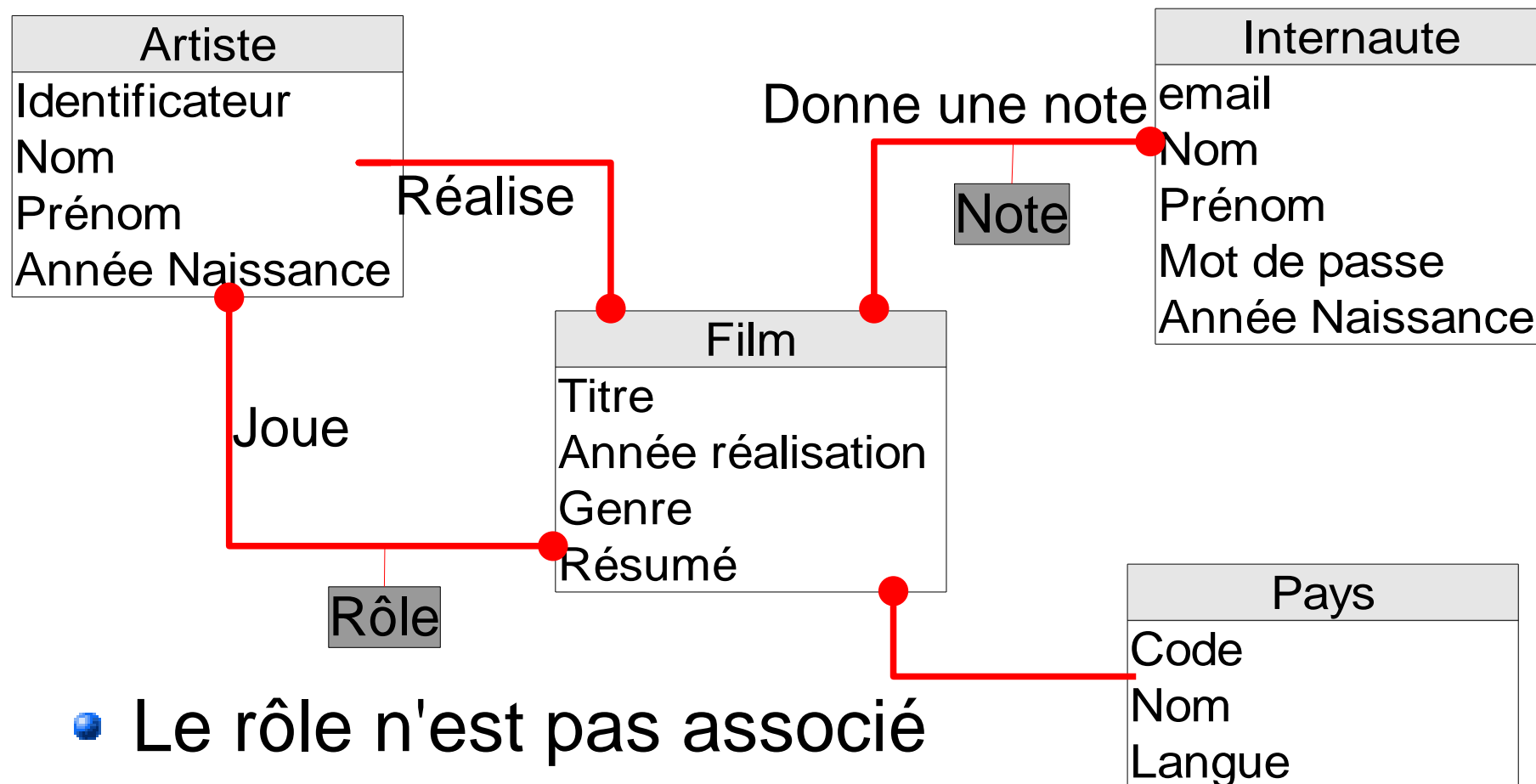
Quand=ADDDATE(CURDATE(), 30)

Agréger des résultats: GROUP BY

```
mysql> SELECT Client, SUM(Depense) FROM  
EnCours GROUP BY Client;
```

```
+-----+-----+  
| Client | Depense |  
+-----+-----+  
| Jules  |    752  |  
| Albert |   1603  |  
| Gaston |    321  |  
+-----+-----+
```

Schéma "complexe"



- Le rôle n'est pas associé
 - au film ou à l'acteur, mais à la participation de l'acteur dans un film.

Contraintes d'intégrité

- **Contrainte d'intégrité** : règle que l'on demande au système de garantir
 - Un attribut doit toujours avoir une valeur (NOT NULL)
 - Un (ensemble d') attribut(s) constitue(nt) la clé de la table
 - Un attribut d'une table est lié à la clé primaire d'une autre table
 - La valeur d'un attribut est unique au sein d'une table
 - Un attribut ne peut prendre qu'une des valeurs prédéfinies d'un ensemble.

Clés Primaires

```
CREATE TABLE Internaute (  
  email VARCHAR(50) NOT NULL,  
  Nom VARCHAR(20) NOT NULL,  
  Prenom VARCHAR(20),  
  Mot_De_Passe VARCHAR(60) NOT NULL,  
  Naissance INTEGER,  
  PRIMARY KEY(email));
```

- Toute table devrait avoir une clé primaire
- Difficile de changer de clé primaire à postériori.

Clés Primaires à plusieurs attributs

CREATE TABLE Notation (

Titre **VARCHAR(50) NOT NULL**,

email **VARCHAR(50) NOT NULL**,

Note **INTEGER DEFAULT 0**,

PRIMARY KEY (Titre, email));

- Une clé primaire peut-être définie par plusieurs attributs.
- Les attributs utilisés doivent être "NOT NULL".

Clés Secondaires

- ```
CREATE TABLE Artiste (
 Ident INTEGER NOT NULL AUTO INCREMENT ,
 Nom VARCHAR(50) NOT NULL,
 Prenom VARCHAR(50) NOT NULL,
 Naissance INTEGER,
 PRIMARY KEY (Ident),
 UNIQUE (Nom, Prenom));
```
- Nom, Prenom: clé secondaire
  - MySQL offre **AUTO\_INCREMENT**, pas SQL...

# Clés Étrangères

```
CREATE TABLE Films (
 Titre VARCHAR(50) NOT NULL,
 Annee INTEGER NOT NULL,
 Id_Real INTEGER,
 Code_Pays INTEGER,
 PRIMARY KEY (Titre),
 FOREIGN KEY (Id_Real) REFERENCES Artiste,
 FOREIGN KEY (Code_Pays) REFERENCES Pays);
```

# Accéder à plusieurs tables

```
mysql> SELECT Titre, Nom FROM Films, Artiste
 -> WHERE Films.Id_Real=Artiste.Ident;
```

```
+-----+-----+
| Titre | Nom |
+-----+-----+
| Alien | Scott |
| Sacrifice | Tarkovski|
+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql>
```

- Accès à des informations contenues dans plusieurs tables: JOINTURE.

# Exemple

- Trouver les clients qui ont commandé le produit dont la description est "Machin Vert"

Table: Produits

|            |            |
|------------|------------|
| Produit_Id | Prod_Descr |
|------------|------------|

Table: Cmd\_Unitaires

|         |            |          |      |      |
|---------|------------|----------|------|------|
| Cmd_Num | Produit_Id | Quantité | Coût | Prix |
|---------|------------|----------|------|------|

Table: Cmdes

|         |           |               |
|---------|-----------|---------------|
| Cmd_Num | Client_Id | Date_Commande |
|---------|-----------|---------------|

Table: Clients

|           |     |        |
|-----------|-----|--------|
| Client_Id | Nom | Prénom |
|-----------|-----|--------|

# Exemple

- Trouver le produit qui s'appelle Machin Vert:
- **SELECT** Produit\_Id **FROM** Produits **WHERE** Prod\_Descr='Machin Vert';
- Ensuite, parcourir le chaînage entre les tables jusqu'à ce que l'on trouve le "Produit" avec l'identificateur correspondant.

# Exemple

```
SELECT Clients.Nom
FROM Clients, Cmdes, Cmd_Unitaires, Produits
WHERE
 Clients.Client_Id = Cmdes.Client_Id
AND Cmdes.Cmd_Num = Cmd_Unitaires.Cmd_Num
AND Cmd_Unitaires.Produit_Id =
 (SELECT Produit_Id FROM Produits
 WHERE Prod_Descr = 'Machin Vert')
ORDER BY Clients.Nom;
```

# Exemple (Cnx.php)

```
<?php
function Connexion($nom, $mdp, $base, $srv)
{
 $connexion = mysql_pconnect($srv, $nom, $mdp)
 if (!$connexion) {
 echo "Connexion au serveur $srv impossible\n";
 exit;
 }
}
```

# Exemple (Cnx.php)

```
if (!mysql_select_db($base, $connexion)) {
 echo "Accès à la base $base impossible\n";
 echo "
Erreur: ";
 echo mysql_error($connexion);
 exit;
}

return $connexion
} // Fin de fonction Connexion
```

# Exemple (Requete.php)

```
<?php
function Requete($req, $cnx)
{
 $resultat = mysql_query($req, $cnx);
 if ($resultat) {
 return $resultat;
 } else {
 echo "Erreur" . mysql_error() . "
";
 }
}
```

# Exemple (Requete.php)

```
function LigneSuivante ($res)
{
return mysql_fetch_array($res);
}
?>
```

# Exemple (AfficheFilms.php)

```
<?php
function AfficheFilms($cnx)
{
 $res = Requete("SELECT * FROM FilmSimple",
 $cnx);
 echo "<table><caption> Liste des Films
 </caption>";
 echo "<tr><th>Titre</th> <th>Annee</th>
 <th>Real</th> <th>Ne en</th> <th>Action</th>
 </tr>";
```

# Exemple (AfficheFilms.php)

```
while ($film=LigneSuivante($res)) {
 $titre_url=urlEncode($film['Titre']);
 echo "<tr><td>$film['Titre']</td>
 <td>$film['Prenom'] $film['Nom']</td>
 <td>$film['Naissance']</td>
 <td><a href='Film.php?mode=MAJ
 &titre=$titre_url'>
 Modifier ce film</td>
 </tr>";
}
```

# Exemple (AfficheFilms.php)

```
echo "</table>";
```

```
}
```

```
?>
```

# Révisions: Plan

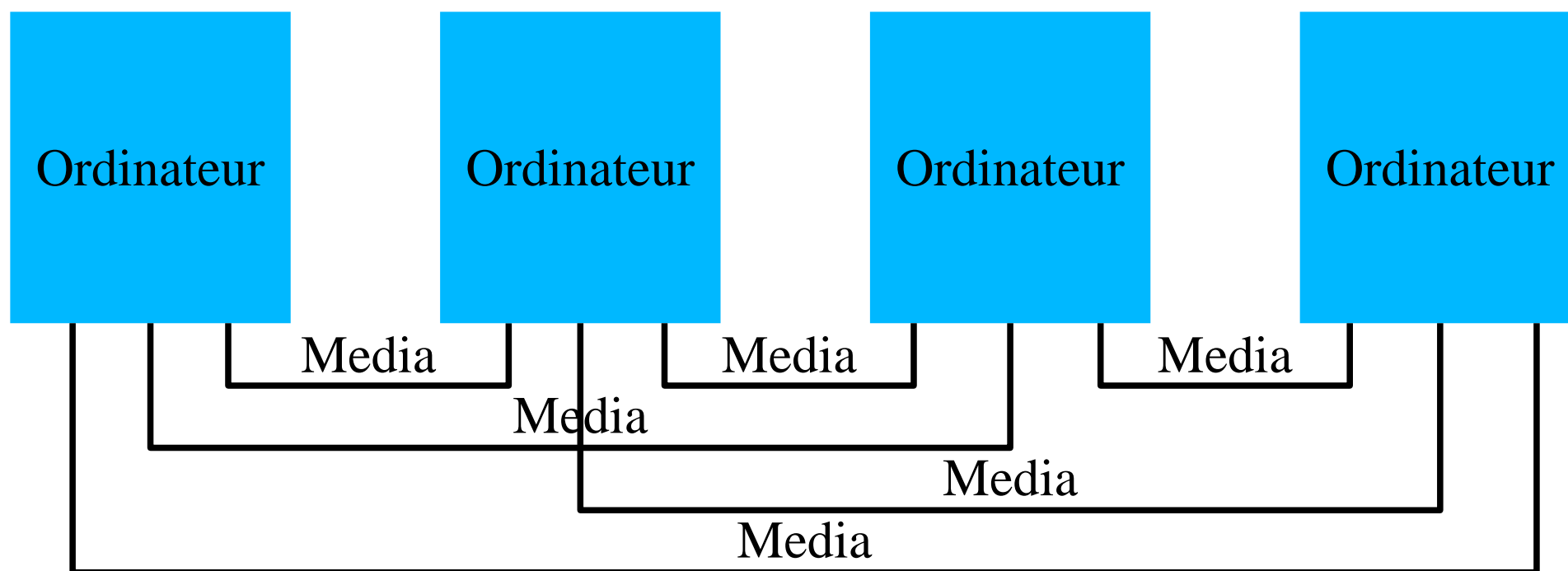
- [X]HTML
- CSS
- PHP
- SQL
- Réseaux

# Problèmes à résoudre

- **Connecter les machines**
  - Support physique,
  - Topologie (schéma de câblage)
- Tour de Babel:
  - Machines hétérogènes
- Partager les connexions entre plusieurs utilisations simultanées

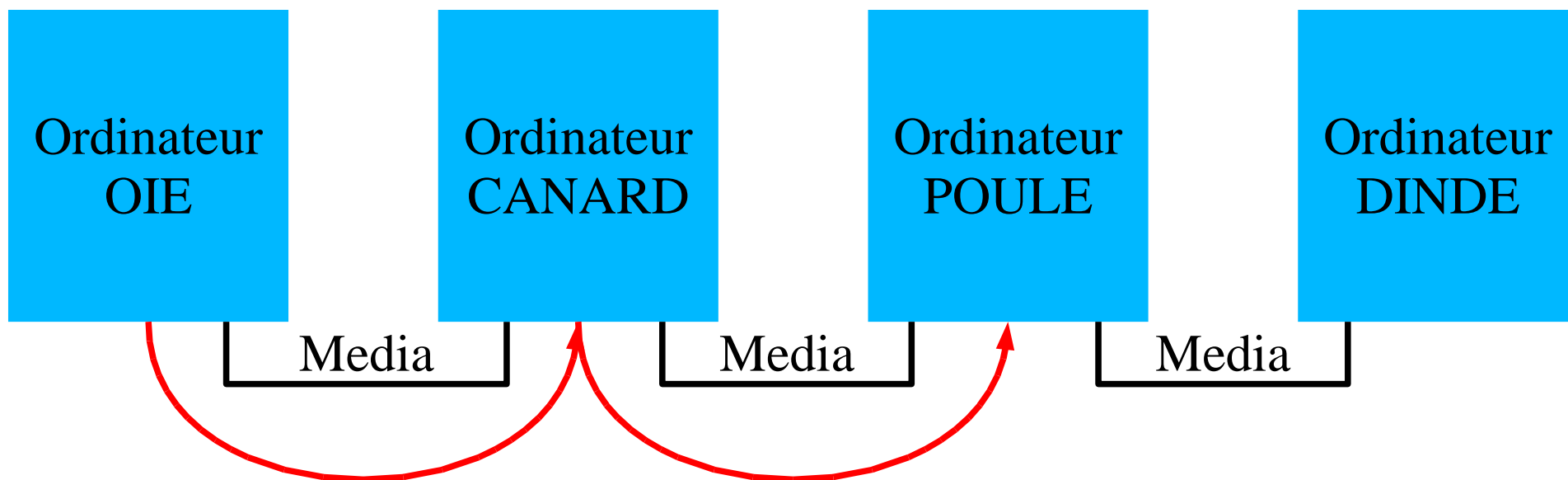
# Liaison et Topologie

- Communication point à point entre plusieurs machines
  - Nombreuses liaisons, difficulté de câblage
  - Impossible si le nombre de machines est grand



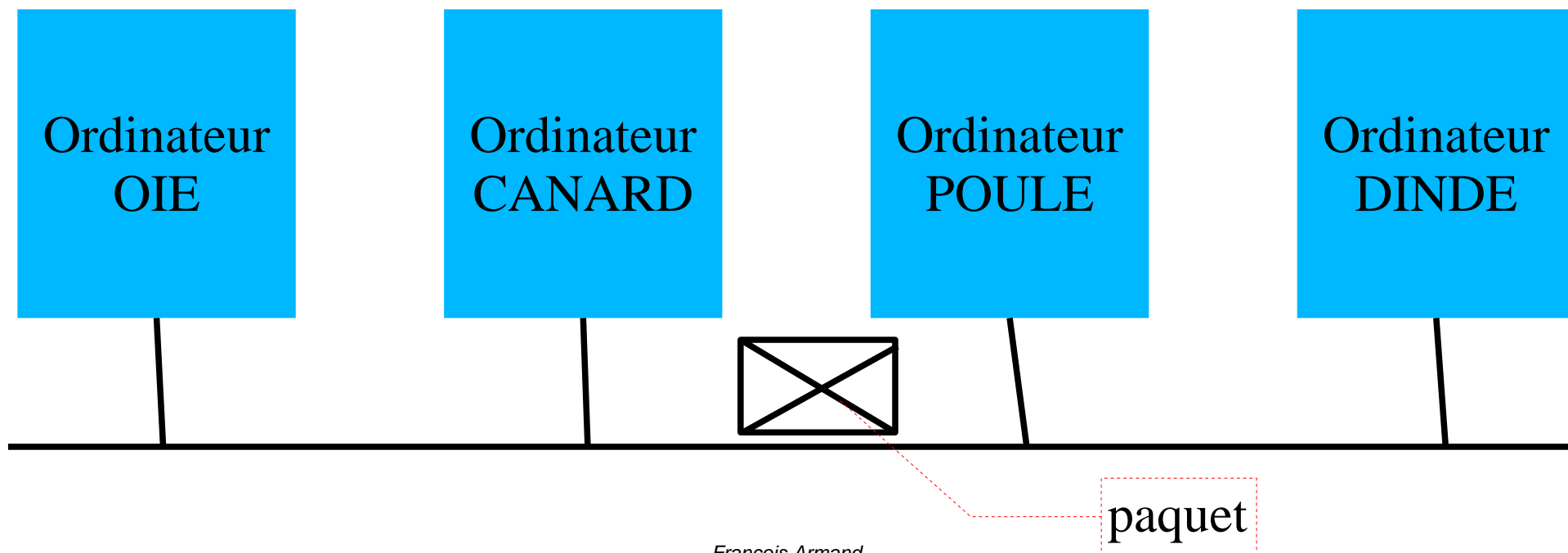
# Liaison et Topologie

- Ne pas inter-connecter toutes les machines directement:
  - Pour envoyer une information de OIE à POULE, il faut la faire transiter par CANARD
  - Trouver le chemin entre émetteur et destinataire:  
**routage**

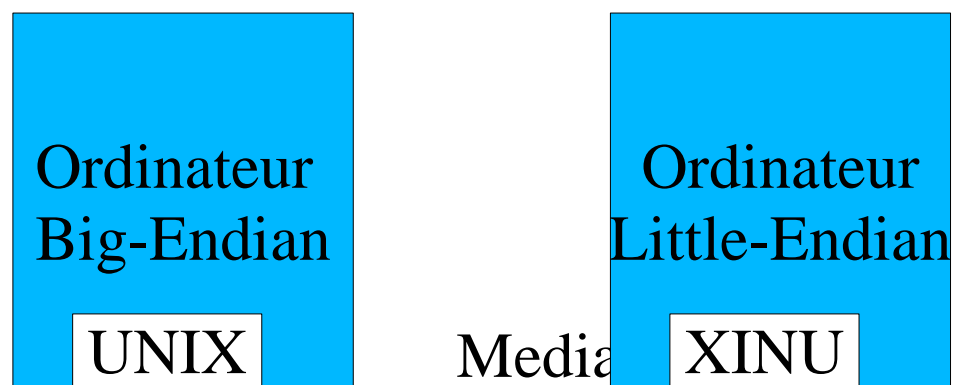


# Liaison et Topologie

- Communication entre plusieurs machines
  - Réseau à diffusion
  - Câblage simple
  - Envoi direct d'une machine à toutes les autres



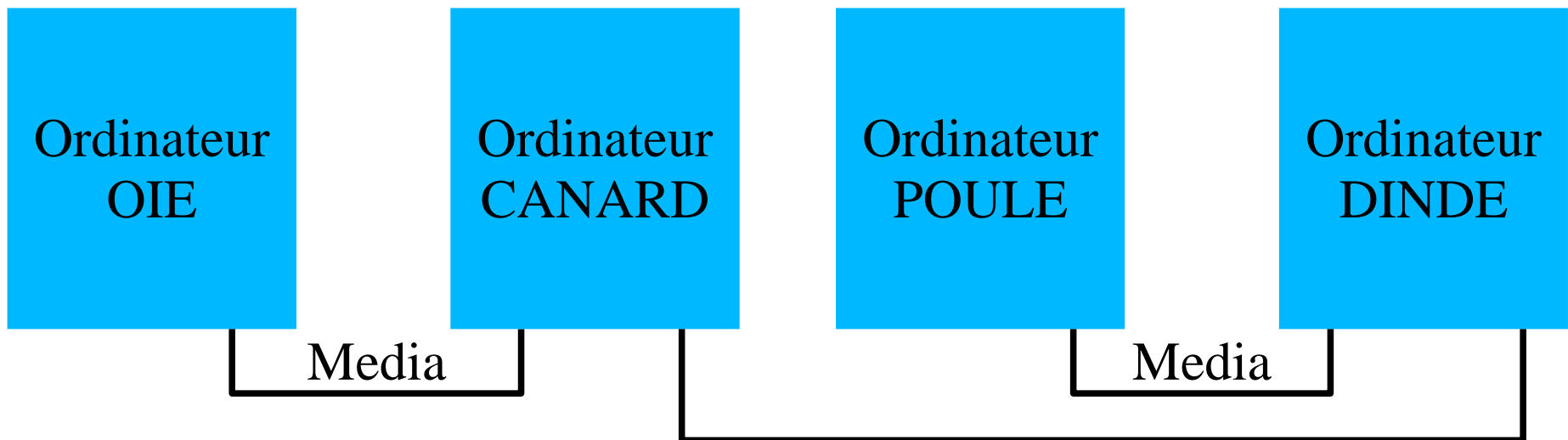
# Échanges entre machines



- Données:
  - La chaîne UNIX sera vue comme XINU sur l'autre machine.
- Programmes:
  - Les jeux d'instructions diffèrent, impossible de transmettre un programme exécutable (sauf interprété: Java)

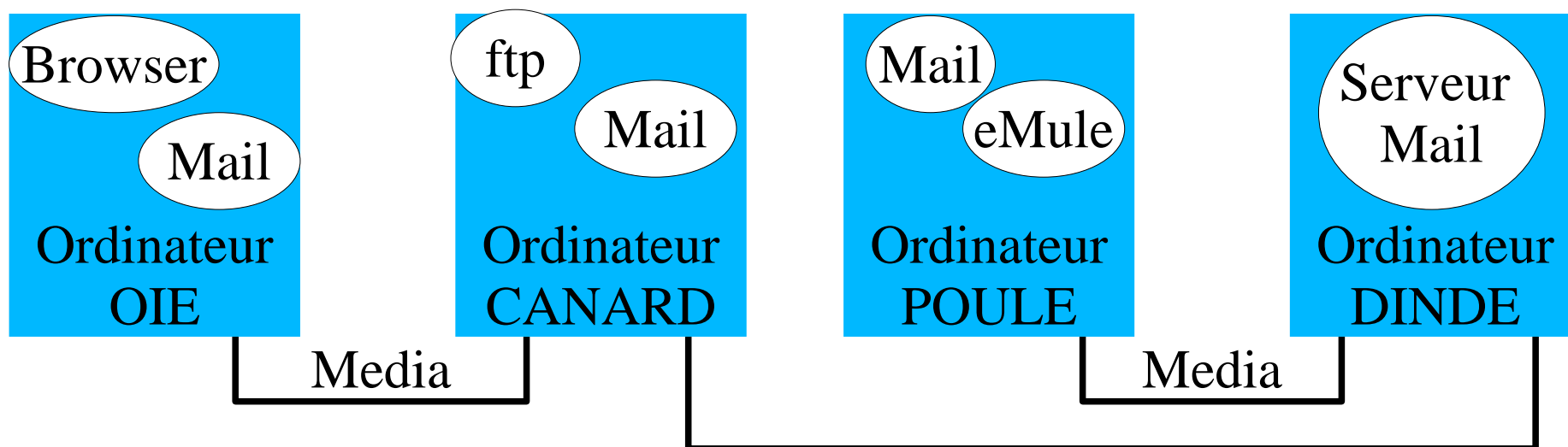
# Concurrence entre machines

- DINDE peut recevoir et avoir à traiter des requêtes en provenance simultanée de CANARD (OIE) et POULE.



# Concurrence entre applications

- Plusieurs applications accèdent au réseau simultanément (Ex: Mail, EMule sur POULE)
- Le serveur de Mail doit fournir son service à plusieurs clients simultanément



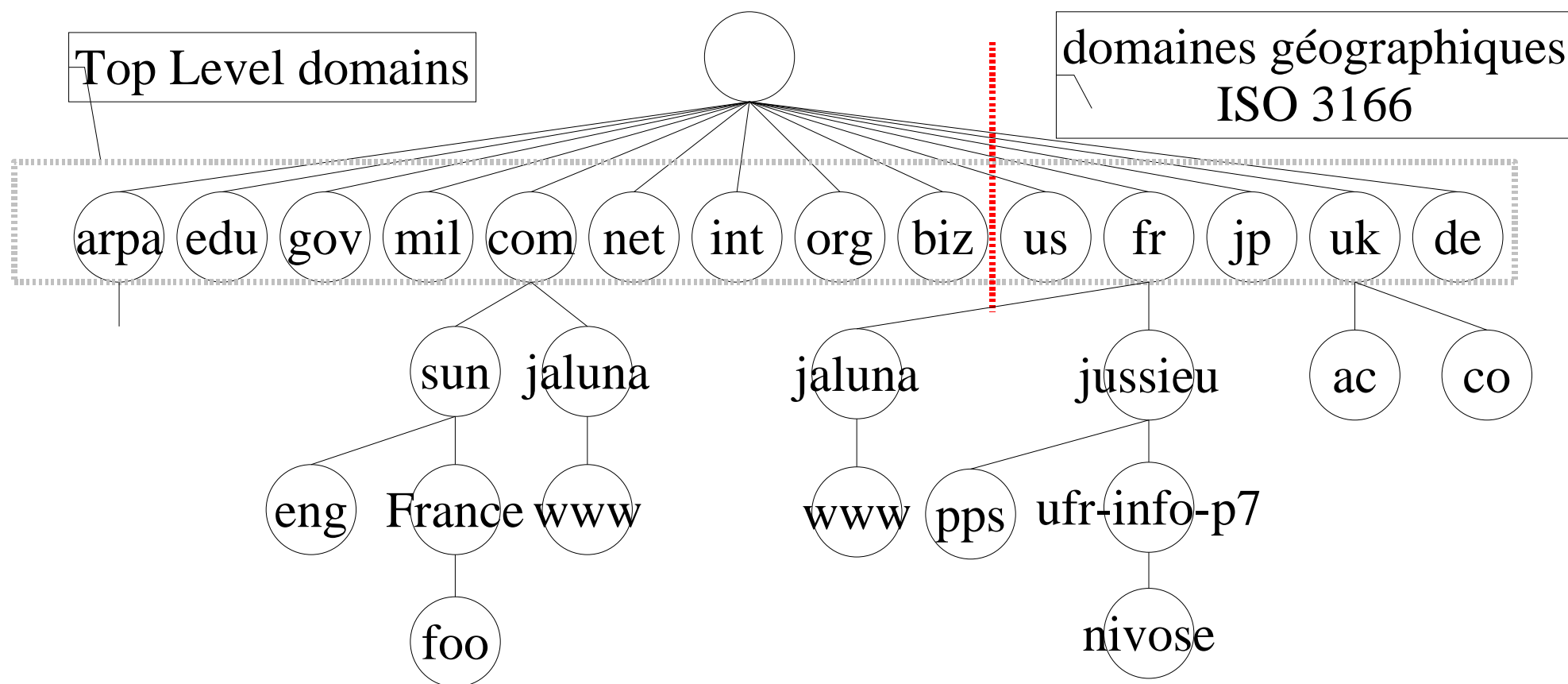
# Adresses et Numéros de Port

- Chaque machine a en principe:
  - une adresse unique (dite adresse IP)
    - Ex (IPv4) : 192.168.1.1 , 127.0.0.1
  - ou plusieurs
- Sur une même machine, chaque application attendant des requêtes du réseau est identifiée par un numéro, dit numéro de port.
  - ftp : 21; ssh : 22; telnet : 23; http : 80
  - tftp : 69; time: 37

# Noms symboliques

- Nommer les machines par un nom plutôt que par son adresse IP
  - Chaîne de caractères
  - Plus "naturel"
- Espace de noms hiérarchique plutôt que "plat"
  - Limiter les conflits
  - Décentraliser sa gestion, délégation d'autorité
- Résolution de nom en adresse IP

# Espace de noms DNS



- Pas de correspondance avec la topologie physique du réseau
- DNS: *Domain Name System*