Séminaire IRISA du 30 mai 2002

Web dynamique : des pages aux applications

Pascal AUBRY

IFSIC - Université de Rennes 1

http://perso.ifsic.univ-rennes1.fr/aubry/presentations/jres2001

UNIVERSITE DE RENNES 1

IRISA
institut de recherche en informatique et systèmes aléatoires

IFSIC
INSTITUT DE FORMATION SUPÉRIEURE en INFORMATIQUE et COMMUNICATION

Introduction

Le protocole HTTP

Des pages statiques aux pages dynamiques

UNIVERSITE DE RENNES 1

IRISA
institut de recherche en informatique
et systèmes aléatoires

IFSIC
INSTITUT DE
FORMATION
SUPÉRIEURE
en INFORMATIQUE et COMMUNICATION

Le World Wide Web

- Système d'information hyper-média réparti
  - texte, images, images animées, son, vidéo …
- Informations stockées sur les serveurs
  - www.jres.org, www.ifsic.univ-rennes1.fr, www.ulaval.ca
- Interrogation des serveurs
  - tels que Apache, IIS, Netscape, Roxen, AOL, iPlanet, Lotus Domino, NCSA httpd, goAhead
  - par clients tels que Netscape, Mosaic, Internet Explorer, Lynx, Opera, Amaya

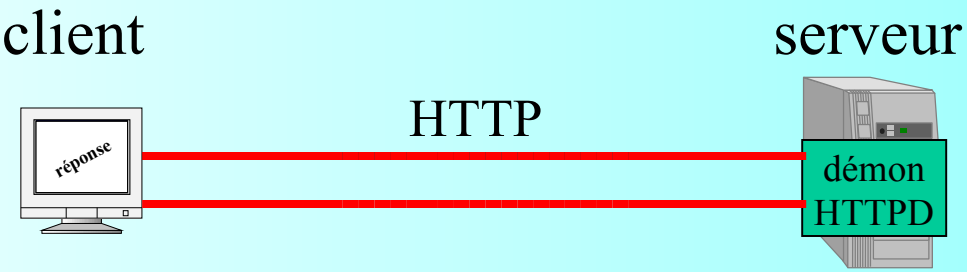
Les principes du Web

- Modèle client-serveur
- Le client envoie des requêtes au serveur
  - transfert de fichiers
  - exécution de programmes sur le serveur
  - mise à jour de fichiers
  - …
- Objets manipulés repérés par leur URL
- Utilisation du protocole HTTP

Le protocole HTTP

- Définit le langage utilisé pour les échanges entre client et serveur Web
  - version 0.9
    - simple protocole de transfert de données (GET et réponse)
  - version 1.0
    - restée un Internet Draft (RFC 1945)
  - actuellement version 1.1
    - RFC 2616 (juin 1999)
- Pas de session permanente entre client/serveur

Déroulement d'une requête HTTP



- Demande de connexion
- Attente de la réponse du serveur
- Établissement de la connexion
- Envoi d'une requête (URL)
- Réponse du serveur
- Affichage de la réponse
- Fermeture de la connexion

Déroulement d'une requête HTTP

client

protocole sans état

serveur

- Demande de connexion
- Attente de la réponse du serveur
- Établissement de la connexion
- Envoi d'une requête (URL)
- Réponse du serveur
- Affichage de la réponse
- Fermeture de la connexion

Exemple de transaction HTTP

- **`telnet www.ifsic.univ-rennes1.fr 80`** ..... *connexion au serveur web*

Exemple de transaction HTTP

```
% telnet www.ifsic.univ-rennes1.fr 80 ....................connexion au serveur web
Trying 148.60.4.30...
Connected to apollon.univ-rennes1.fr.
Escape character is '^]'.
```

Exemple de transaction HTTP

```
% telnet www.ifsic.univ-rennes1.fr 80 ....................connexion au serveur web
Trying 148.60.4.30...
Connected to apollon.univ-rennes1.fr.
Escape character is '^]'.
GET /index.html HTTP/1.1 ..................................demande de transfert
Host: apollon.univ-rennes1.fr ............................nom du serveur
From: pa@ifsic.univ-rennes1.fr ...........................adresse demandeur (optionnelle)
```

(ligne blanche = fin de l'entête HTTP de la requête)

Exemple de transaction HTTP

```
% telnet www.ifsic.univ-rennes1.fr 80 ..................connexion au serveur web
Trying 148.60.4.30...
Connected to apollon.univ-rennes1.fr.
Escape character is '^]'.
GET /index.html HTTP/1.1 ..................................demande de transfert
Host: apollon.univ-rennes1.fr ............................nom du serveur
From: pa@ifsic.univ-rennes1.fr ...........................adresse demandeur (optionnelle)
                                        (ligne blanche = fin de l'entête HTTP de la requête)
HTTP/1.1 200 OK ...........................................réponse du serveur
Date: Tue, 02 Jun 2001 14:11:17 GMT
Server: Apache/1.3b6
Last-Modified: Mon, 07 Apr 2001 10:39:08 GMT ............informations sur la ressource
ETag: "b3dd-524-33b78ccc"
Content-Length: 1316 ......................................taille de la ressource
Accept-Ranges: bytes
Content-Type: text/html ...................................type MIME
                                        (ligne blanche = fin de l'entête HTTP de la réponse)
<DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">          (contenu)
<HTML>
….
</HTML>
```

Exemple de transaction HTTP

```
% telnet www.ifsic.univ-rennes1.fr 80 ..................connexion au serveur web
Trying 148.60.4.30...
Connected to apollon.univ-rennes1.fr.
Escape character is '^]'.
GET /index.html HTTP/1.1 ..................................demande de transfert
Host: apollon.univ-rennes1.fr ............................nom du serveur
From: pa@ifsic.univ-rennes1.fr ...........................adresse demandeur (optionnelle)
                                        (ligne blanche = fin de l'entête HTTP de la requête)
HTTP/1.1 200 OK ...........................................réponse du serveur
Date: Tue, 02 Jun 2001 14:11:17 GMT
Server: Apache/1.3b6
Last-Modified: Mon, 07 Apr 2001 10:39:08 GMT ............informations sur la ressource
ETag: "b3dd-524-33b78ccc"
Content-Length: 1316 ......................................taille de la ressource
Accept-Ranges: bytes
Content-Type: text/html ...................................type MIME
                                        (ligne blanche = fin de l'entête HTTP de la réponse)
<DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">          (contenu)
<HTML>
….
</HTML>
Connection closed by foreign host.                          fermeture de la connexion
```

Les nouveautés de HTTP 1.1

- Identification obligatoire du *hostname* par le client
  - « *virtual hosting* »
- Négociation du contenu
  - documents français/anglais, HTML/PDF
- Connexions persistantes
  - amélioration temps de chargement de pages avec images
- Prise en charge des mandataires (*proxies*)
- Nouveaux codes réponse
- Nouvelles méthodes

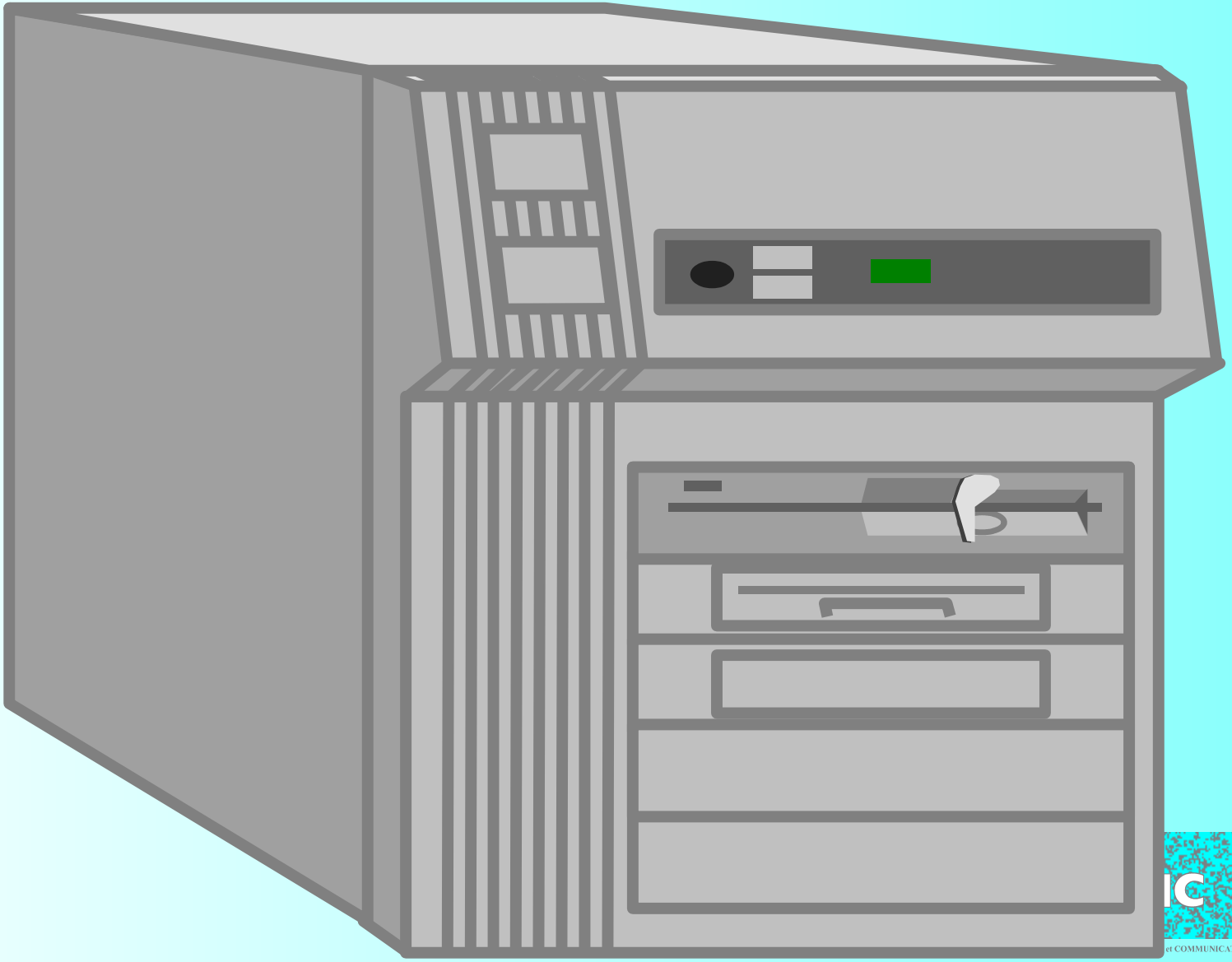
Les méthodes HTTP

- Demander une ressource
  - **GET** : rapatrier une ressource
  - **POST** : envoyer des données et rapatrier une ressource
- Avoir de l'information sur une ressource
  - **HEAD** : connaître ses caractéristiques
  - **OPTIONS** : connaître les options qui lui sont applicables
- Mettre à jour une ressource à distance
  - **PUT** : la créer ou remplacer son contenu
  - **DELETE** : la détruire
- Débogguer
  - **TRACE** : tracer les mandataires (*proxies*)

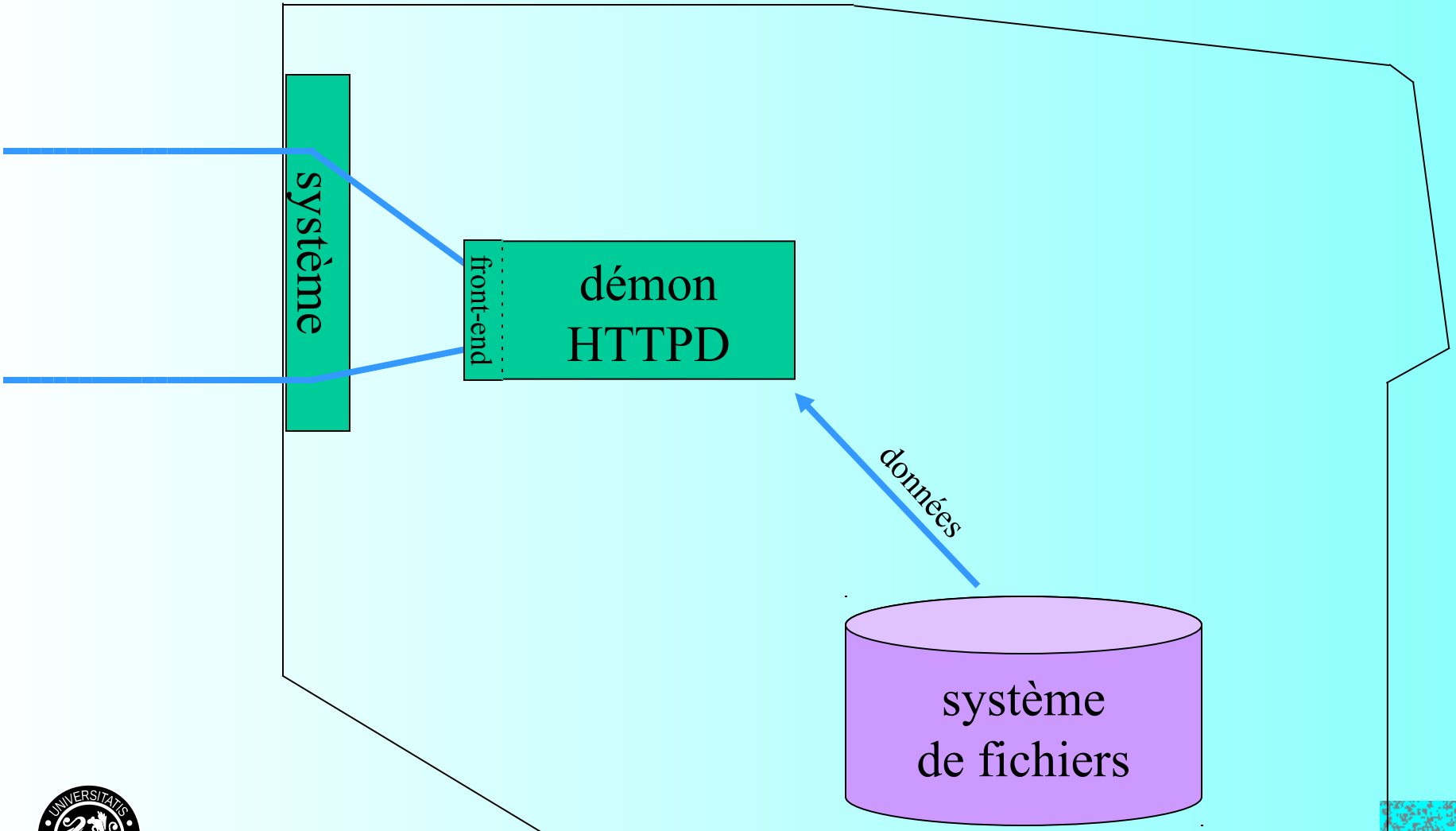
Les codes de réponse sous HTTP

- 10x : information
- 20x : succès — 200 OK
- 30x : redirection — 301 Moved permanently
- 40x : erreur du client — 401 Unauthorized
  402 Payment required
  403 Forbidden
  404 Not found
- 50x : erreur du serveur — 500 Internal server error

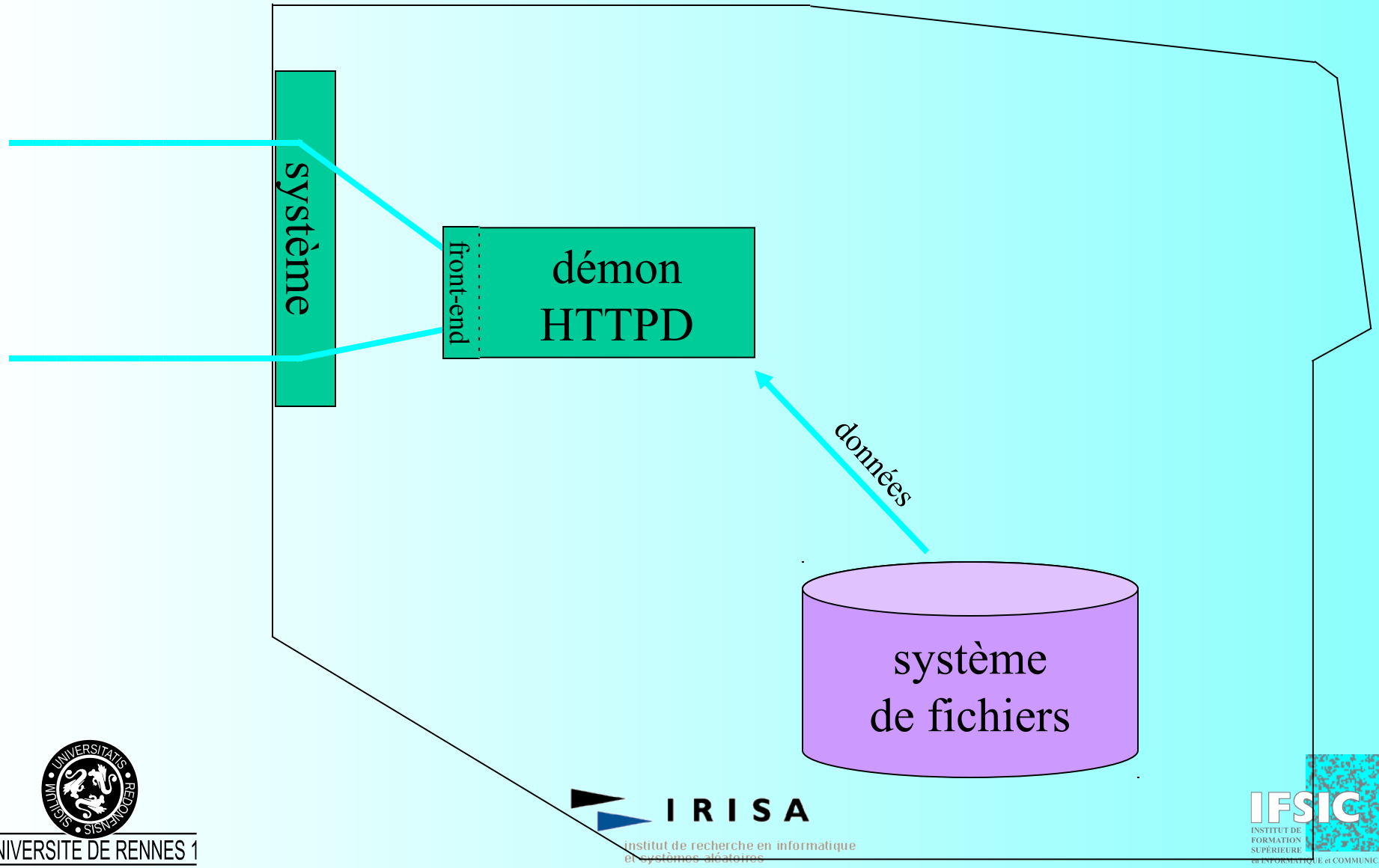
Délivrer un document statique



Délivrer un document statique



Délivrer un document statique



Le web à l'IFSIC

- Un réseau hétérogène
  - DOS, Windows, Solaris, Irix, Linux
- Des besoins applicatifs
  - configuration des comptes utilisateurs
  - changement des mots de passe
  - interrogations de quotas (disque, impression)
  - rendus des TPs aux enseignants
  - gestion administrative
    - édition de l'offre de formation
    - gestion des services et des heures complémentaires
- Un client « universel »

Pourquoi développer sur le web ?

- Universalité du protocole HTTP
  - indépendance vis-à-vis des clients
  - pérennité des applications
- Simplicité de l'interface
- Parce que ça fait moderne ;-)

Web dynamique : qui fait quoi ?

- Le serveur exécute, le client reçoit
  - SSI, XSSI, CGI, FastCGI, PHP, ASP, JSP
  - indépendance vis-à-vis du client (navigateur)
  - interactivité limitée
- Le serveur envoie, le client exécute
  - JavaScript embarqué (DHTML), Applet Java
  - dépendance vis-à-vis du client
  - plus d'interactivité

Pages dynamiques : solutions côté serveur

Le protocole CGI

FastCGI, Perl, PHP, Servlets, JSP, ASP

UNIVERSITE DE RENNES 1

IRISA
institut de recherche en informatique
et systèmes aléatoires

IFSIC
INSTITUT DE
FORMATION
SUPÉRIEURE
en INFORMATIQUE et COMMUNICATION

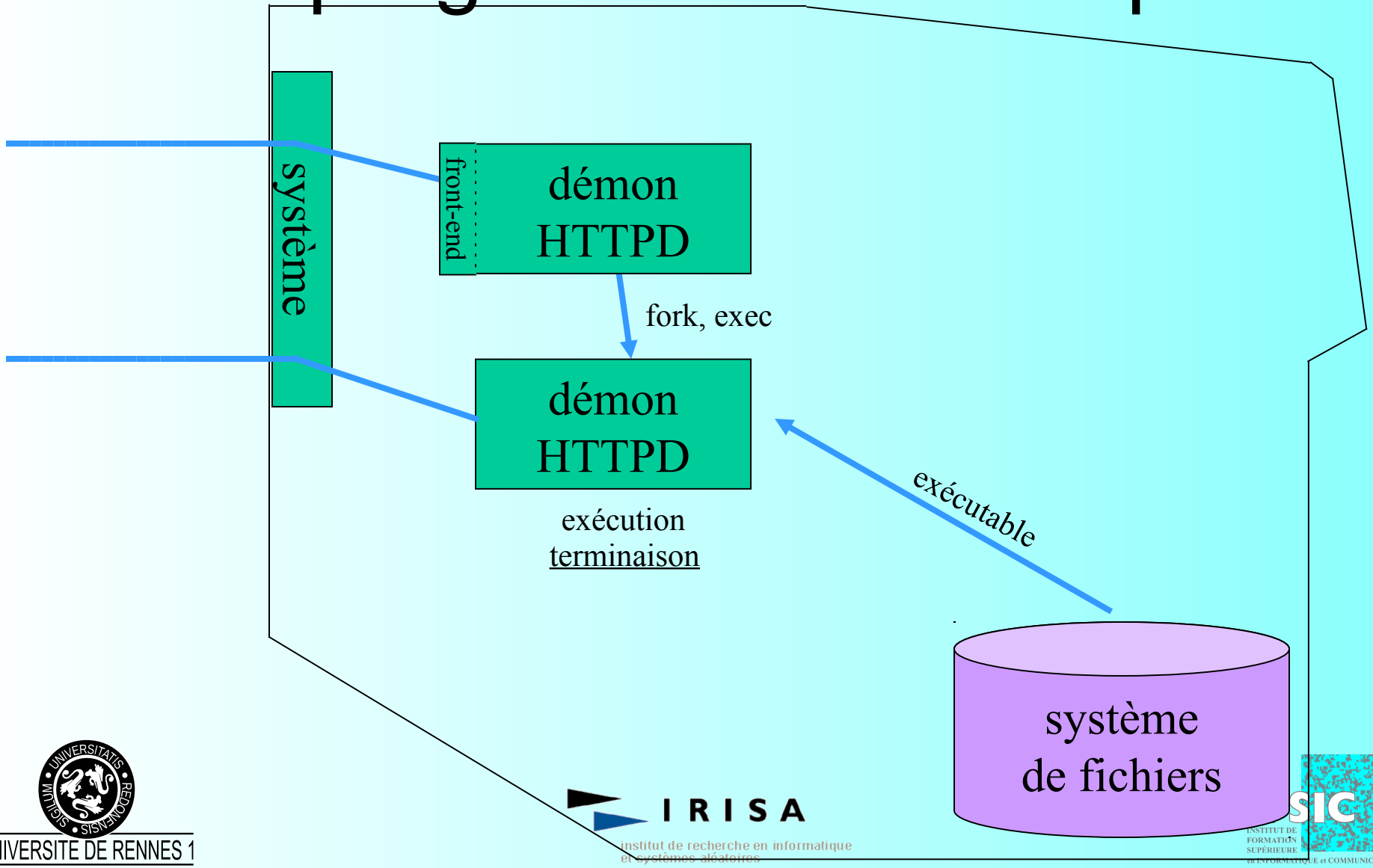
SSI, XSSI :
le premier pas vers la dynamique

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>
<head><title>Exemple XSSI</title></head>
<body bgcolor="#FFFFFF">
<!--#config errmsg="erreur de syntaxe dans un (x)SSI" -->
<!--#set var="x" value="test"--><!--#echo var="x"--><br>
Fichier modifié le : <!--#echo var="LAST_MODIFIED" --> <br>
Nom du serveur : <!--#echo var="SERVER_NAME" --> <br>
<!--#if expr="$HTTP_USER_AGENT = 'Mozilla/2.0 (compatible;
MSIE 3.01; Windows 95)'" -->
 Il est temps de se mettre à jour !
<!--#endif -->
</body>
</html>
```

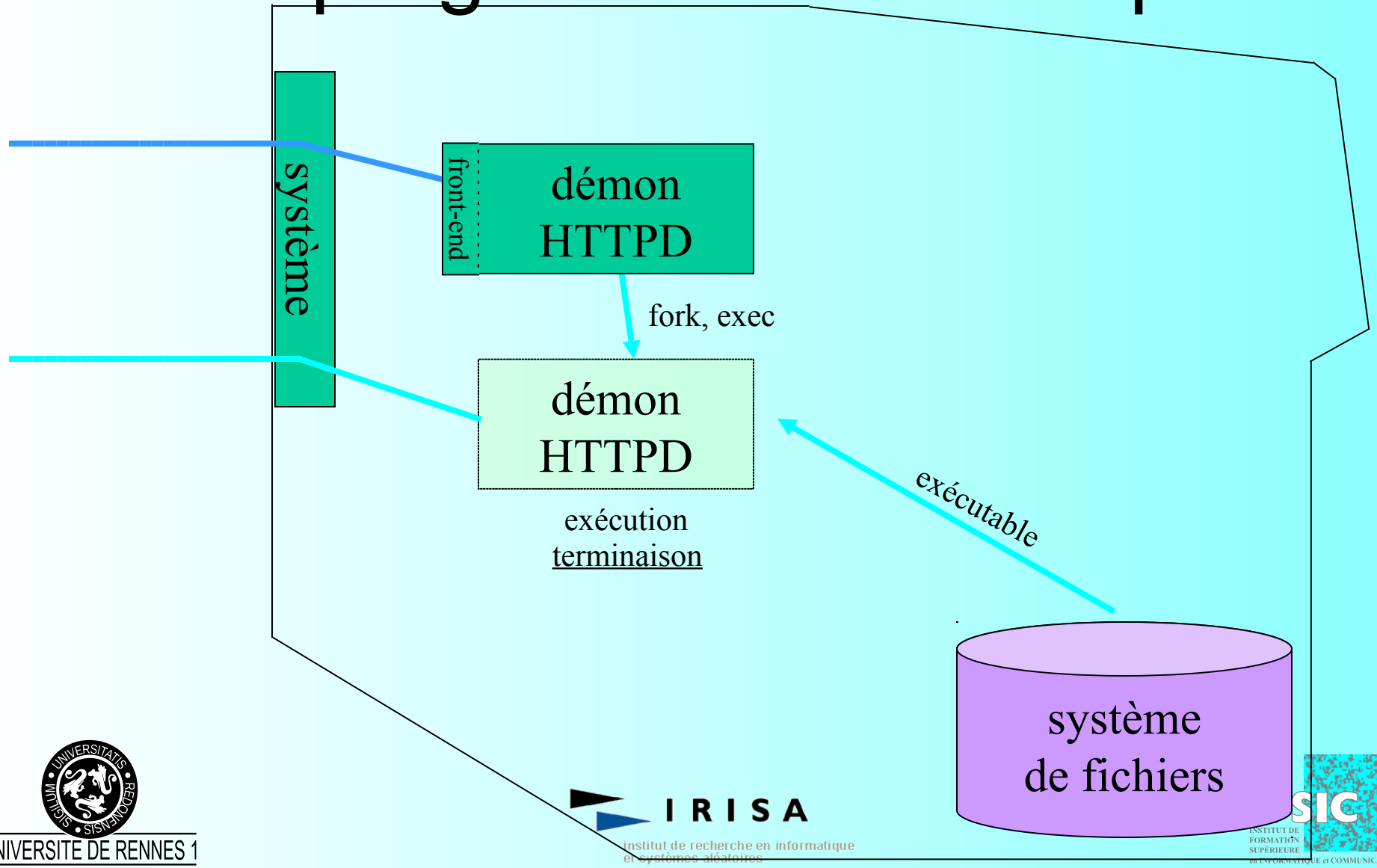
CGI : Common Gateway Interface

- Un standard pour l'interface entre applications et serveurs d'informations
- Permet de passer des paramètres aux requêtes
  - dans l'URL avec la méthode GET
    http://serv.dom.org/cgi-bin/script?arg1=val1&arg2=val2
  - comme des données avec la méthode POST
- Exécution d'un programme sur le serveur
  - Les informations renvoyées au client sont statiques
  - Des requêtes successives permettent le dynamisme

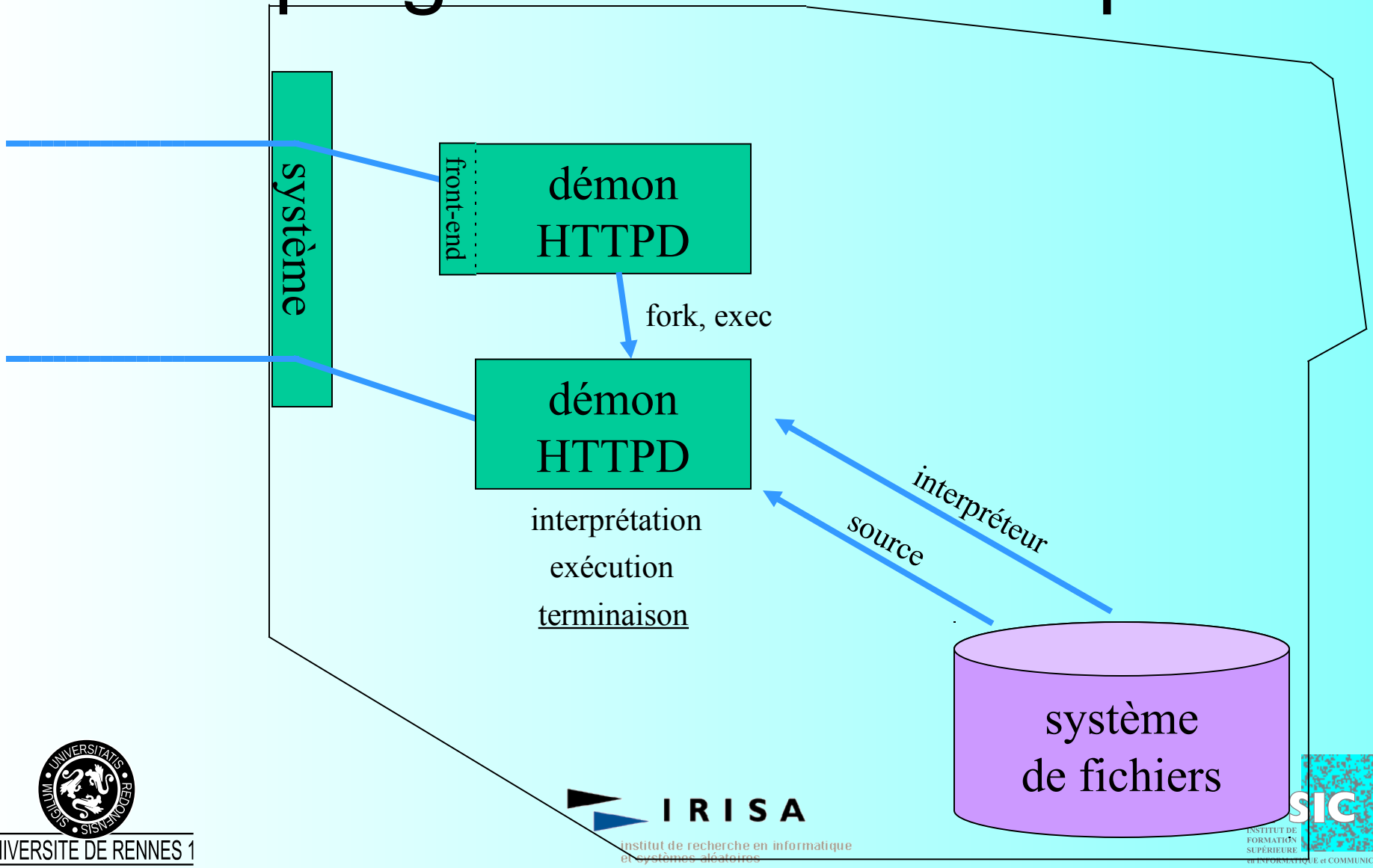
Exécution via HTTP d'un programme CGI compilé



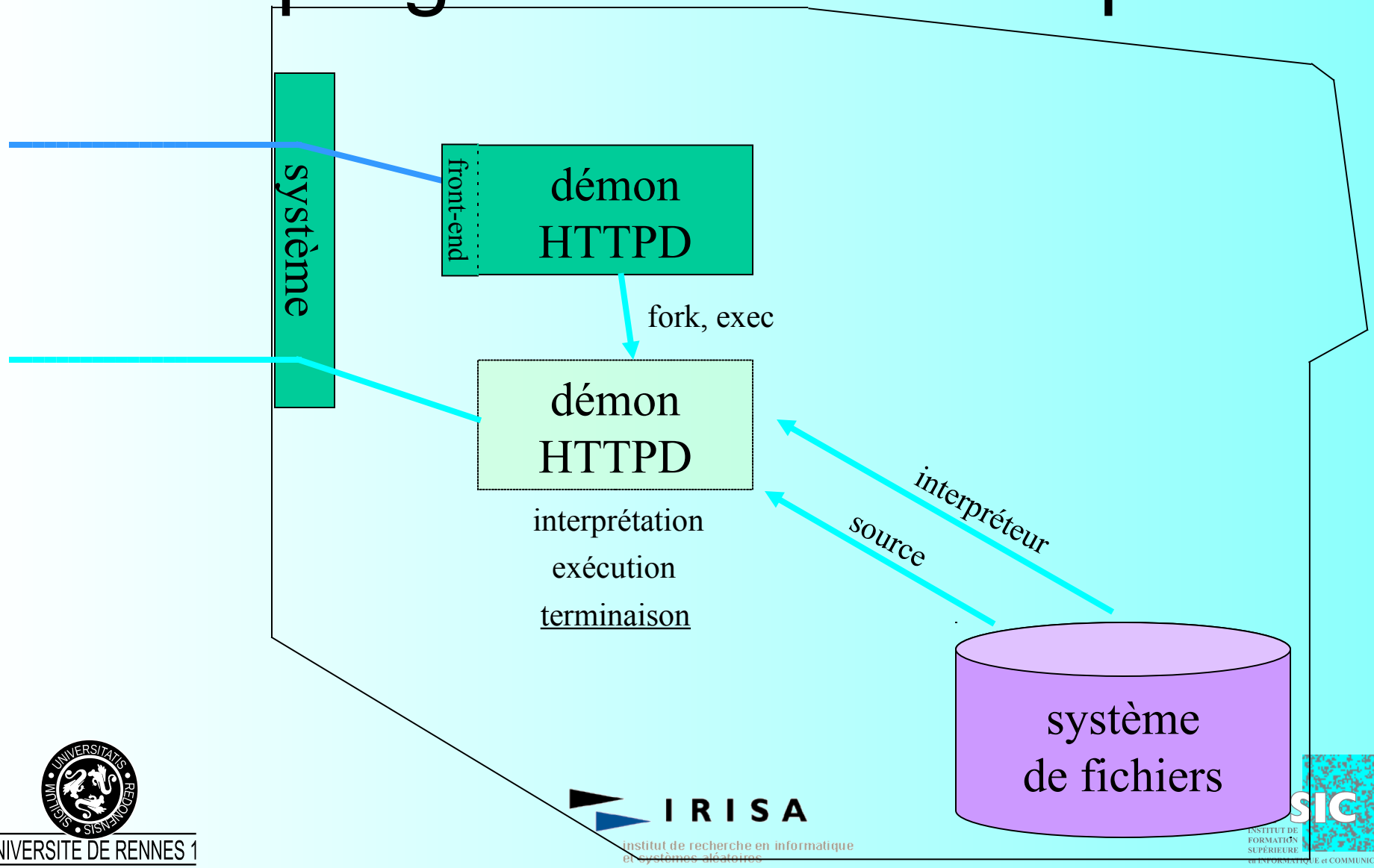
Exécution via HTTP d'un programme CGI compilé



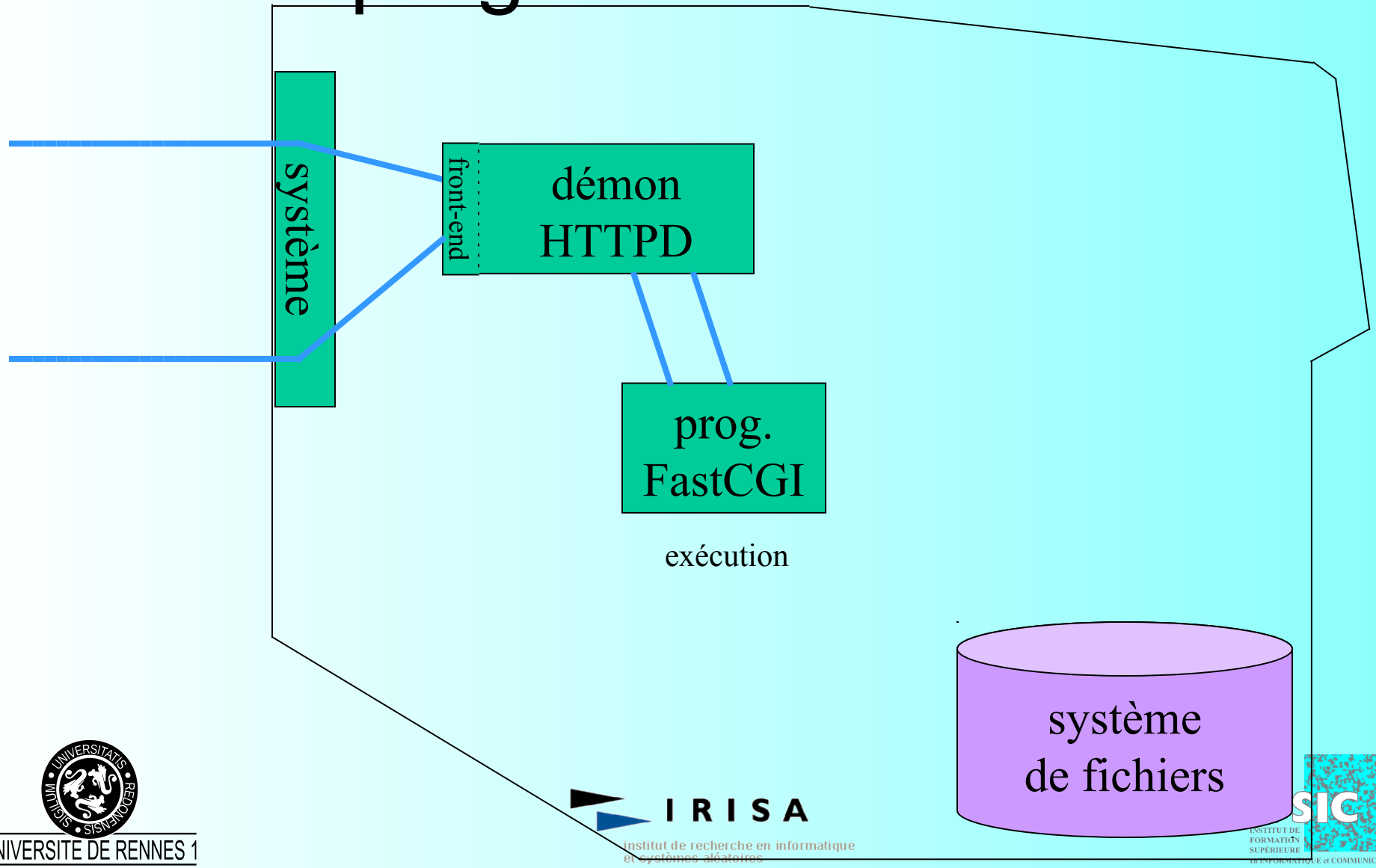
Exécution via HTTP d'un programme CGI interprété



Exécution via HTTP d'un programme CGI interprété



FastCGI : une solution au coût des programmes externes



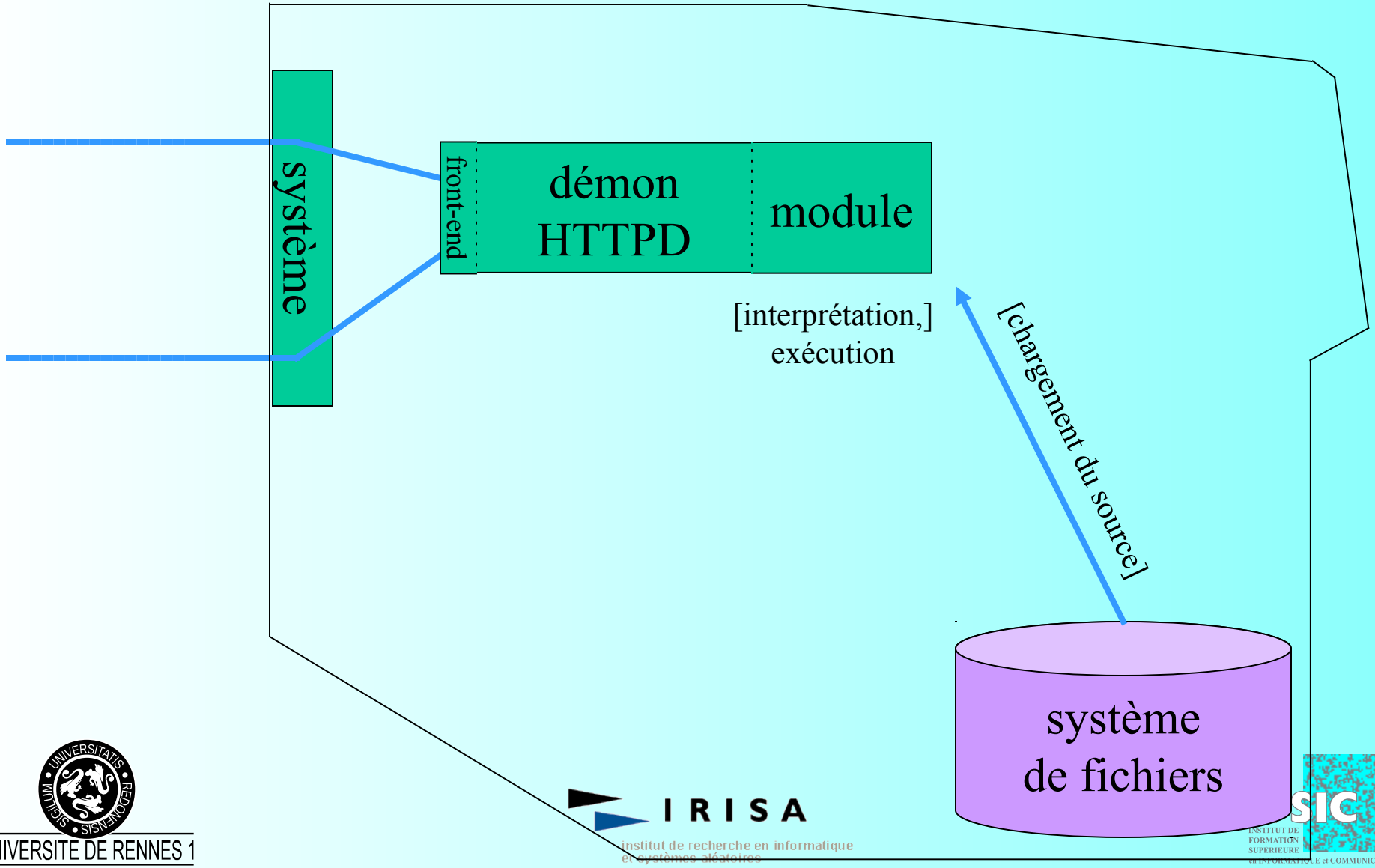
FastCGI

- Performance, simplicité (comme CGI)
- Migration facile (de l'existant CGI)
- Isolation du démon httpd
  - sécurité en cas de crash
  - Indépendance de l'architecture du serveur
- Distribution possible de la charge

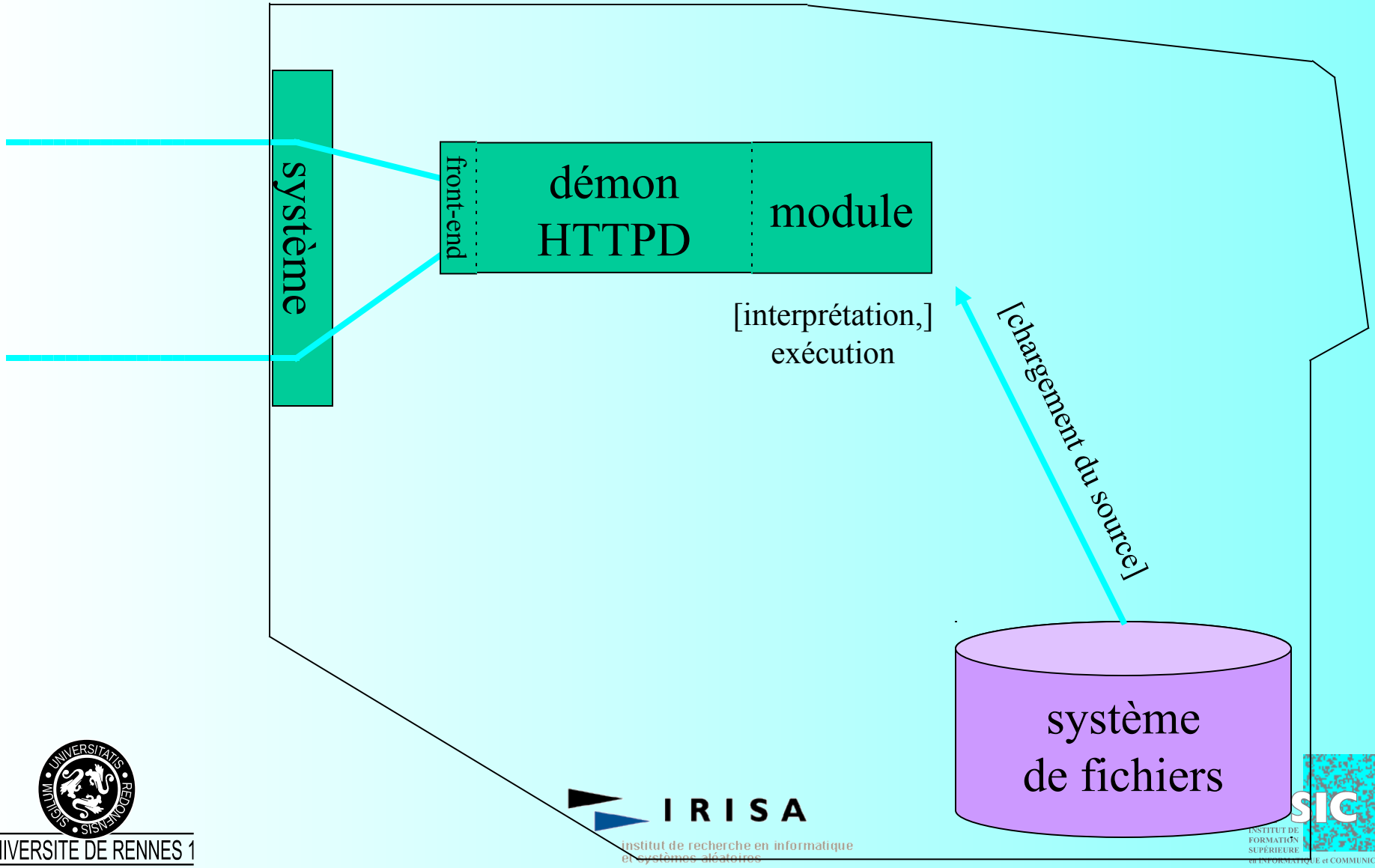
Scripts côté serveur : les quatre leaders du marché

- Perl
- PHP (Php Hypertext Preprocessor)
- JSP (Java Server Pages)
- ASP (Active Server Pages)

L'approche modulaire (PHP, Perl)



L'approche modulaire (PHP, Perl)



Perl

- Practical Extraction and Report Language
- Ressources illimitées
- Module d'Apache (mod_perl) ou stand-alone
- Multi-plateformes
- On aime ou on n'aime pas ;-)

PHP

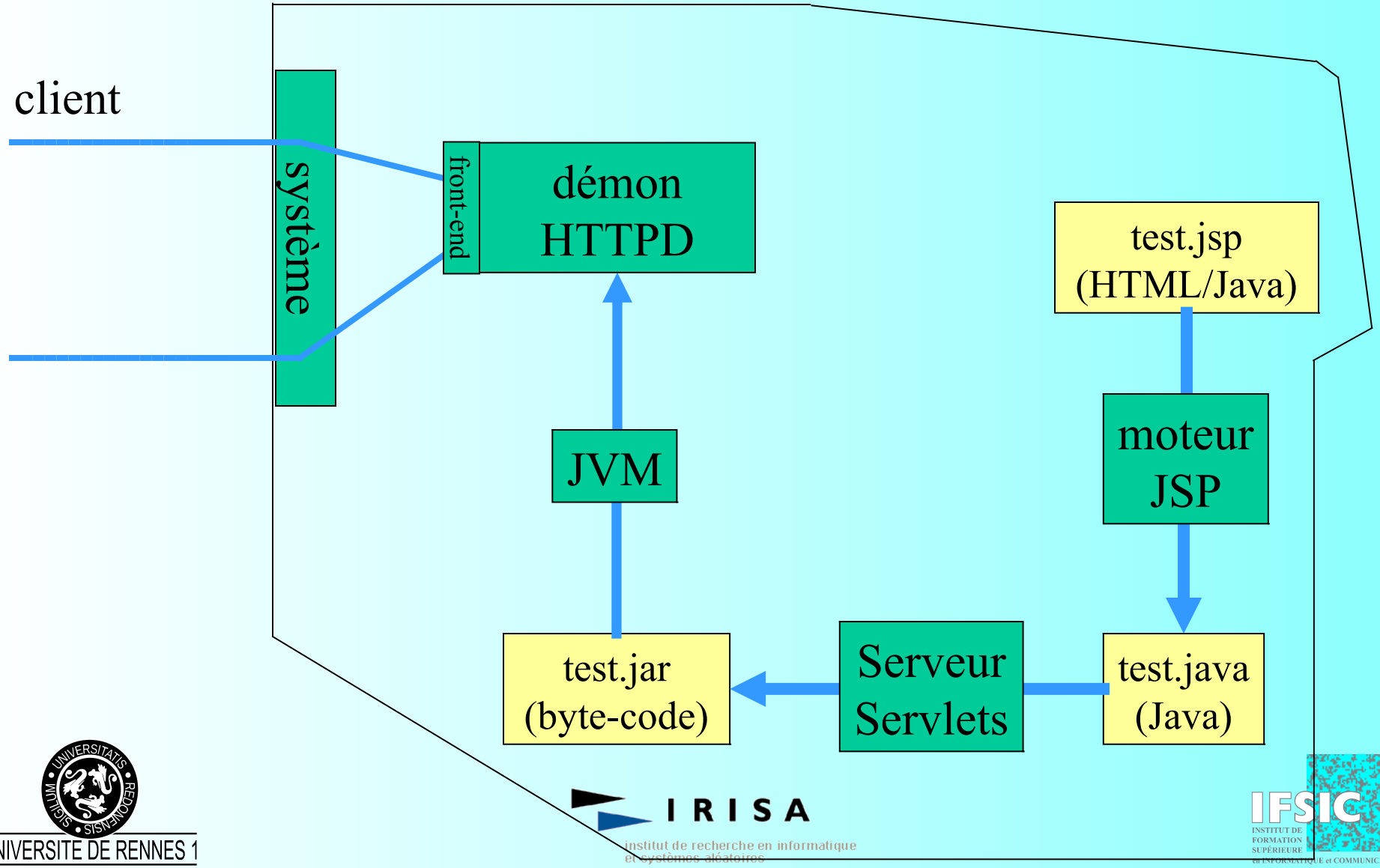
- Multi-plateformes
- Le plus simple
- Un langage non généraliste (comme Java et Perl) mais néanmoins très riche (bibliothèques)
- Un produit libre

Java

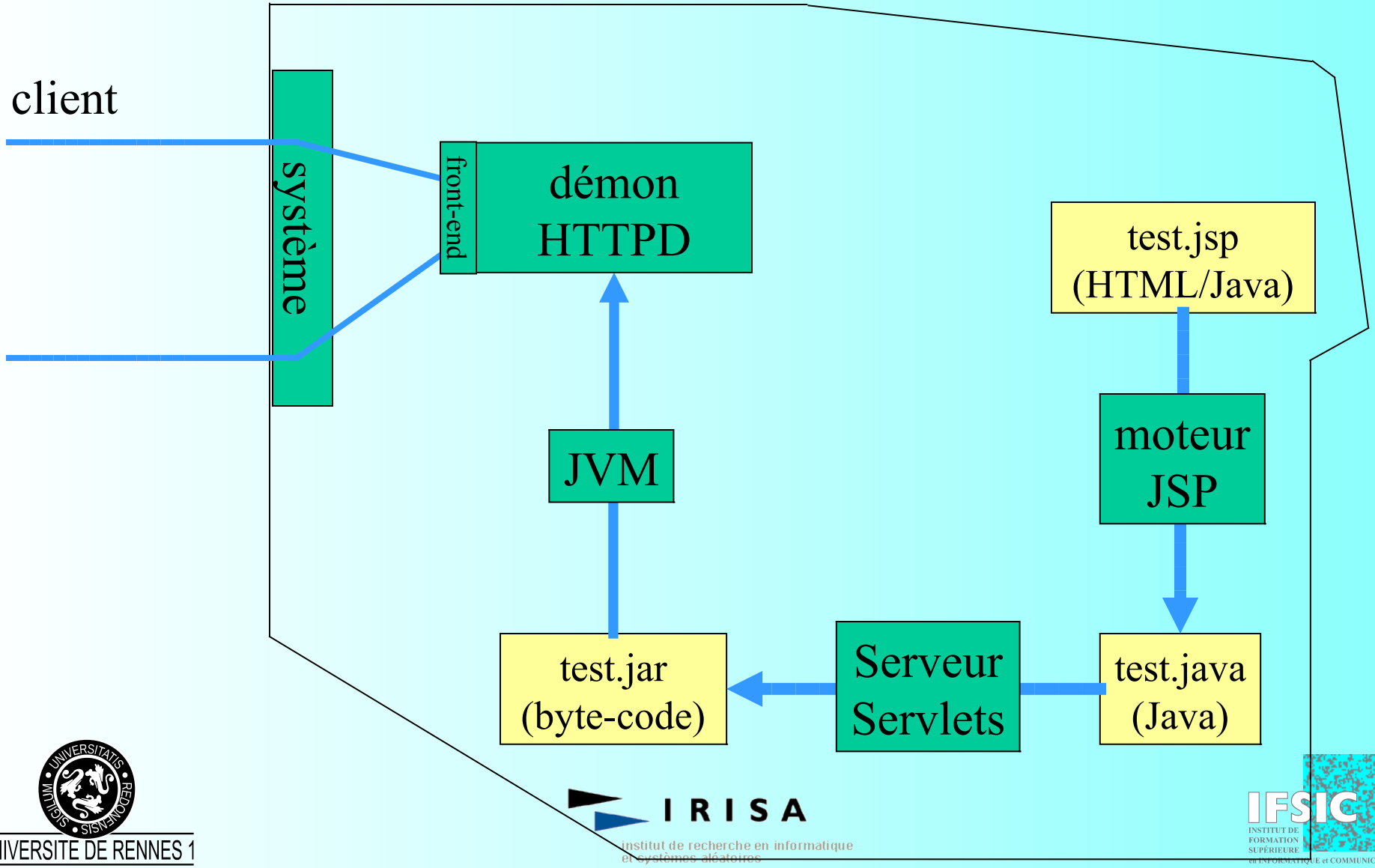
- Les servlets
  - Programme Java exécuté côté serveur
  - Transformation en byte-code avant exécution
  - Servies par un serveur dédié (ex : Tomcat)

- JSP (Java Server Pages)
  - Pages HTML avec Java embarqué
  - Transformation en Servlets

Le mécanisme JSP



Le mécanisme JSP



ASP

- Tourne nativement sur IIS (MicroSoft)
- JavaScript ou VBScript
- ASP+ :
  - support langages compilés (VB, C++, C#)
  - compilation intermédiaire JS et VBS
  - mécanisme de cache des objets compilés
  - configuration format XML
  - contrôle accru des formulaires
- portabilité limitée (www.chilisoft.com)

Les langages de CGI

- Les shells
  - sh, tcsh, bash
- Les langages compilés
  - C, C++, Pascal, ...
- Perl
- Java (JSP, Servlets)
- PHP
- Python
- VBScript, JavaScript (ASP)

Un site web : des responsabilités, des métiers

- Technique (ingénieur)
  - fonctionnement du service
  - mise à disposition d'outils
  - développement
- Éditoriale (rédacteur)
  - mise à jour du contenu
- Graphique (infographiste)
  - présentation du site
- Pénale (juriste)

Pourquoi un environnement de développement ?

- Pour gérer l'accès concurrent
  - les droits sur les systèmes de fichiers sont insuffisants
  - on utilise souvent une base de données
    et/ou une interface web
- Pour séparer les tâches
  - graphisme, interfaçage Homme/Machine
  - développement
  - mise à jour du contenu
- Pour intégrer plusieurs technologies
  - COM, CORBA, EJB, C/C++, Java, XML

Les environnements de développement

- BEA WebLogic (Java)
- Lutris Enhydra (Java, open source)
- Netscape Application Server
- Jbuilder (Java)
- Zope (Python)
- Midgard (PHP, open source)
- Allaire/Macromedia ColdFusion 5 (CFML)
- Vignette Content Suite V6
- ...

Une comparaison rapide…

	[Java|VB]Script (+ASP)	Java (+JSP)	Perl (+mod_perl)	PHP	Python (+Zope)
Installation	★★★★★	★★	★★★★★	★★★★★	★★★★★
Apprentissage	★★★	★★	★	★★★★★	★★★★
Puissance	★★★	★★★★★	★★★	★★★	★★★
Portabilité	★	★★★★★	★★★★★	★★★	★★★
Outils	★★★★	★★★★	★★	★★★	★★★
Ressources	★★★	★★★★	★★★★★	★★★★★	★★★

…et subjective !

Comment choisir ?

- 1. Évaluer le ou les projets à mettre en œuvre
- 2. Évaluer ses moyens
  - humains
  - financiers
- 3. Tenir compte de l'existant
  - les habitudes
  - les compétences
  - les préférences
- 4. Se faire soi-même une idée des produits

Les problèmes de la programmation dynamique sur le web à travers PHP

UNIVERSITE DE RENNES 1

IRISA
institut de recherche en informatique
et systèmes aléatoires

IFSIC
INSTITUT DE
FORMATION
SUPÉRIEURE
en INFORMATIQUE et COMMUNICATION

PHP en quelques mots

- Un langage
  - qui comprend CGI
  - complet
  - simple
  - hérité de Perl, C et sh
- Un module d'Apache
  - performant
  - existe aussi en stand-alone
  - en pleine évolution
    - au départ pour les pages personnelles ;-)
    - aujourd'hui pour les applications sur le web

Un programme PHP, c'est…

- Un fichier HTML…

- dans lequel on trouve des tags PHP :

```
<html>
<head><title>TEST</title></head>
<body>
   <p>
      Il est <?echo date("H:i");?>.
   </p>
</body>
</html>
```

Dans la pratique… (surtout pour les gros projets)

- Un programme PHP est un ensemble d'instructions PHP qui affichent :
  - du code HTML
  - ou autre chose :
    - ascii
    - PostScript, PDF
    - GIF, JPEG, PNG, …

- L'approche « pages » est désormais supplantée par l'approche « composants »

Une page HTML avec des instructions PHP

```
<html>
  <head>
    <title>
      Bonjour !
    </title>
  </head>
  <body>
    <p>
      sur le serveur, il est exactement
      <? echo date("H:i:s") ; ?>
    </p>
  </body>
</html>
```

Un programme PHP

```
class sortie {
  function sortie($titre)  // constructeur
    { $this->titre = $titre ; }
  function debut()
    { echo "<html><head>$this->titre</head><body>" ; }
  function fin()
    { echo "</body></html>" ; }
}

    $s = new sortie("Bonjour !") ;
    $s->debut() ;
    echo "<p>sur le serveur, il est exactement "
        .date("H:i:s")
        ."</p>" ;
    $s->fin() ;
```

Les paramètres CGI

```
<form action="test_cgi.php" method="GET">
  Texte :<br>
  <input type="text" name="texte_court" value="blabla"><hr>
  Sélection simple :<br>
  <select name="sel_simple">
    <option value="1">Choix 1</option>
    <option value="2">Choix 2</option>
    <option value="3">Choix 3</option>
  </select><hr>
  Sélection multiple :<br>
  <select multiple name="sel_multiple[]">
    <option value="1">Choix 1</option>
    <option value="2">Choix 2</option>
    <option value="3">Choix 3</option>
  </select><hr>
  <input type="reset" name="bouton_reset" value="Annuler">
  <input type="submit" name="bouton_submit" value="Valider">
</form>
```

Les paramètres CGI

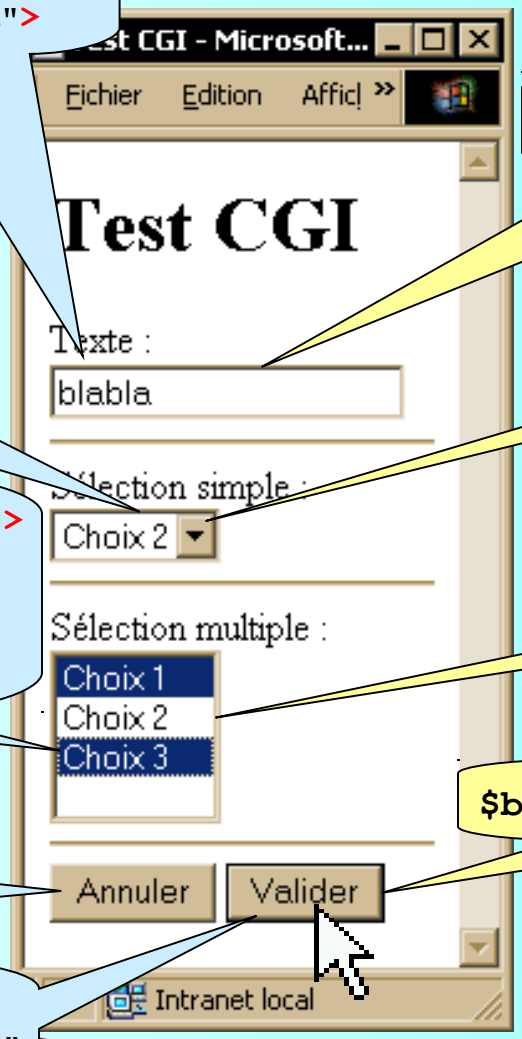
```
<input type="text"
       name="texte_court"
       value="blabla">
```

```
<select name="sel_simple">
  <option value="1">Choix 1</option>
  <option value="2">Choix 2</option>
  <option value="3">Choix 3</option>
</select>
```

```
<select multiple name="sel_multiple[]">
  <option value="1">Choix 1</option>
  <option value="2">Choix 2</option>
  <option value="3">Choix 3</option>
</select>
```

```
<input type="reset"
       name="bouton_reset"
       value="Annuler">
```

```
<input type="submit"
       name="bouton_submit"
       value="Valider">
```




```
$texte_court : "blabla"
```

```
$sel_simple : "2"
```

```
$sel_multiple[0] : "1"
$sel_multiple[1] : "3"
```

```
$bouton_submit : "Valider"
```

PHP et les bases de données

- Un support très large
  - Adabas D, dBase, filePro, Hyperwave, Informix, InterBase, mSQL, MS SQL Server, MySQL, Oracle 7, Oracle 8, SyBase, PostgreSQL, Solid, ODBC
- Utiliser la bibliothèque de son SGBD
- En l'absence de bibliothèque, utiliser ODBC
  - ou développer sa propre bibliothèque ;-)
- Il existe des interface génériques
  - phpDB

Connexions persistantes (aux bases de données)

- La connexion à une base de données est souvent l'opération la plus coûteuse d'une requête
- Identification d'une connexion :
  (*host*, *database*, *user*, *password*)

- Au lieu d'ouvrir et fermer une connexion à une base de données à chaque requête, on peut garder les connexions ouvertes et les mémoriser pour pouvoir les réutiliser par la suite

Accès aux bases de données

```
Require("DB.php") ;

$conn = DB::Connect("mysql://user:passwd@host/db" ) ;

$res = $conn->Query(" . . . ") ;

while ( $row = $res->FetchRow() )
    { . . . }

$conn->Close() ;
```

Sécurité et bases de données

```
<form ...>
    ...
    <input type="text"
           name="cle">
    <input type="submit"
           name="go"
           value="chercher">
</form>
```

Sécurité et bases de données

```
<form ...>
    ...
    <input type="text"
           name="cle">
    <input type="submit"
           name="go"
           value="chercher">
</form>
```

```
$req_sql =
    "SELECT DesChamps
     FROM UneTable
     WHERE cle = '$cle' ";
```

J'essaierai bien la clé suivante :

```
4' ;
SELECT * FROM … ;
DROP DATABASE … ;
SELECT '1
```

Sécurité et bases de données

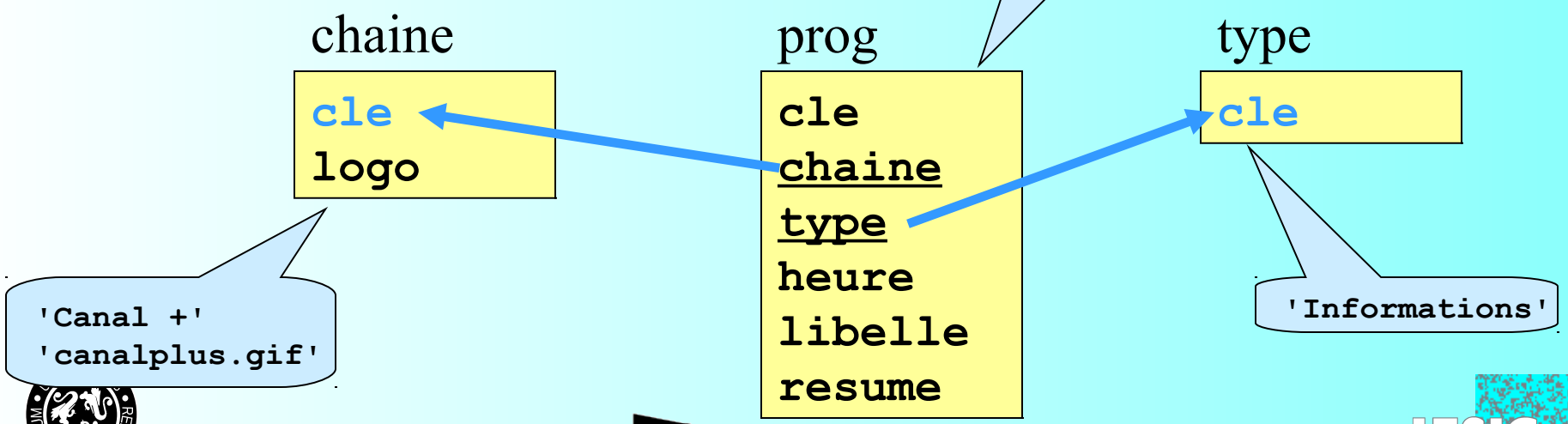
- Tout ce qui provient de l'utilisateur doit être considéré comme suspect :
  - paramètres CGI, fichiers téléchargés, ...

```
$req_sql = "
    SELECT DesChamps FROM UneTable
    WHERE cle = '". Addslashes($cle) ."'" ;
```

Zappeur.org

- Objectif : faire un site web d'interrogation des programmes de télévision
- Moyens :
  - un serveur Web (RedHat Linux 7.1, Apache 1.3.19, P[illegible].0.4pl1)
  - une base de données MySQL :

```
1458
'Canal +'
'Informations'
'2001-12-09 12:40:00'
'Le vrai journal'
'Présenté par Karl Zéro...'
```



Récupération des types d'émission

```
// importation de la bibliothèque DB
require_once( 'DB.php' );

// informations de connexion
$info = "mysql://zappeur:pass@localhost/zappeur" ;

// connexion à la base de données
$db = DB::connect($info) ;

// interrogation de la base de données
$sql = "SELECT cle FROM type ORDER BY cle" ;
$result = $db->query($sql) ;

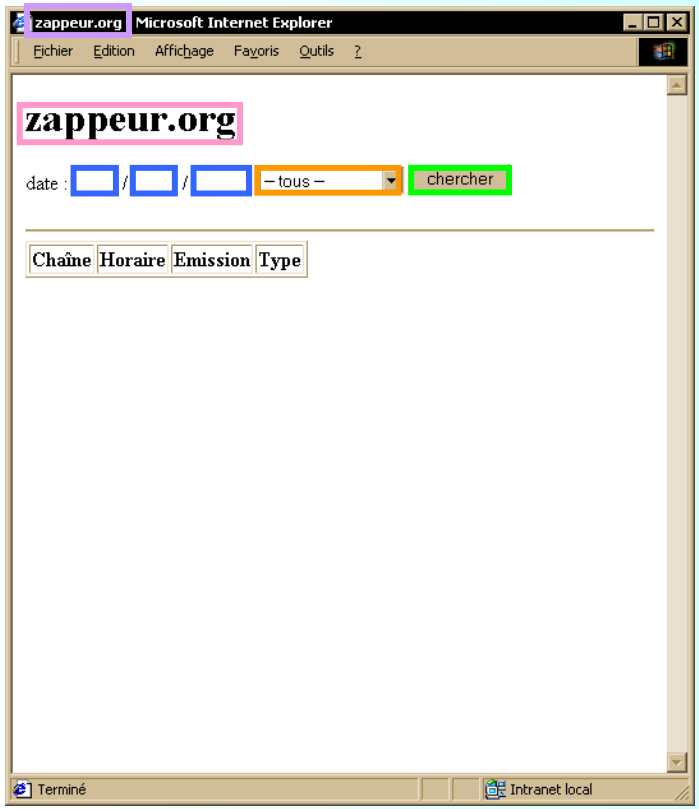
// parcours du résultat
while ($row = $result->fetchRow(DB_FETCHMODE_ASSOC))
  { // $row["cle"] contient le type }

  // libération des ressources
  $result->free() ;
```

Récupération des programmes

```
$user_date = addslashes($annee-$mois-$jour) ;
$sql = "
SELECT cle, chaine, libelle, type,
       HOUR(heure) as h, MINUTE(heure) as m
    FROM programme
    WHERE heure >= '$user_date'
      AND heure < DATE_ADD('$user_date', INTERVAL 1 DAY)" ;
if ( !empty($type) )
  $sql .= " AND type = '".addslashes($type)."'" ;
$sql .= " ORDER BY heure,chaine" ;
$result = $db->query($sql) ;
while ($row = $result->fetchRow(DB_FETCHMODE_ASSOC))
  {
    // $row["chaine"], $row["h"], $row["m"],
    // $row["libelle"], $row["type"]
  }
$result->free() ;
```

Première ébauche




```
<?
  require_once( 'DB.php' );
  $titre ="zappeur.org" ;
  $db = DB::connect( "mysql://zappeur:pass@localhost/zappeur" );
?>

<html>
  <head><title><? echo $titre ; ?></title></head>
  <body>
    <h1><? echo $titre; ?></h1>
    <form action=<? echo $PHP_SELF ;?> method="POST">
      date : <input type="text" name="jour" size="2">
      / <input type="text" name="mois" size="2">
      / <input type="text" name="annee" size="4">
      <select name="type">
        <option value="">-- tous --</option>
        <?
          $result = $db->query("SELECT cle FROM type ORDER BY cle") ;
          while ($row = $result->fetchRow(DB_FETCHMODE_ASSOC))
            echo "<option value=\"".$row["cle"]."\">".$row["cle"]."</option>" ;
          $result->free() ;
        ?>
      </select>
      <input type="submit" name="go" value="chercher">
    </form>
    <hr>
    <table border="1">
      <tr><th>Chaîne</th><th>Horaire</th><th>Emission</th><th>Type</th></tr>
      <?
        $user_date = addslashes($annee)."-".addslashes($mois)."-".addslashes($jour) ;
        $sql = "SELECT cle, chaine, libelle, type,
                HOUR(heure) as h, MINUTE(heure) as m
                FROM programme
                WHERE heure >= '$user_date'
                  AND heure < DATE_ADD('$user_date', INTERVAL 1 DAY)" ;
        if ( !empty($type) ) $sql .= " AND type = '".addslashes($type)."'" ;
        $sql .= " ORDER BY heure,chaine" ;
        $result = $db->query($sql) ;
        while ($row = $result->fetchRow(DB_FETCHMODE_ASSOC))
            echo "<tr><td>".$row["chaine"]."</td>
                      <td>".$row["h"]."h".$row["m"]."</td>
                      <td>".$row["libelle"]."</td>
                      <td>".$row["type"]."</td></tr>" ;
      ?>
    </table>
  </body>
</html>

<? $db->disconnect() ; ?>
```

Première ébauche

```
<?
  require_once( 'DB.php' );
  $titre ="zappeur.org" ;
  $db = DB::connect( "mysql://zappeur:pass@localhost/zappeur" );
?>

<html>
  <head><title><? echo $titre ; ?></title></head>
  <body>
    <h1><? echo $titre; ?></h1>
    <form action=<? echo $PHP_SELF ;?> method="POST">
      date : <input type="text" name="jour" size="2">
      / <input type="text" name="mois" size="2">
      / <input type="text" name="annee" size="4">
      <select name="type">
        <option value="">-- tous --</option>
        <?
          $result = $db->query("SELECT cle FROM type ORDER BY cle") ;
          while ($row = $result->fetchRow(DB_FETCHMODE_ASSOC))
            echo "<option value=\"".$row["cle"]."\">".$row["cle"]."</option>" ;
          $result->free() ;
        ?>
      </select>
      <input type="submit" name="go" value="chercher">
    </form>
    <hr>
    <table border="1">
      <tr><th>Chaîne</th><th>Horaire</th><th>Emission</th><th>Type</th></tr>
      <?
        $user_date = addslashes($annee)."-".addslashes($mois)."-".addslashes($jour) ;
        $sql = "SELECT cle, chaine, libelle, type,
                HOUR(heure) as h, MINUTE(heure) as m
                FROM programme
                WHERE heure >= '$user_date'
                  AND heure < DATE_ADD('$user_date', INTERVAL 1 DAY)" ;
        if ( !empty($type) ) $sql .= " AND type = '".addslashes($type)."'" ;
        $sql .= " ORDER BY heure,chaine" ;
        $result = $db->query($sql) ;
        while ($row = $result->fetchRow(DB_FETCHMODE_ASSOC))
            echo "<tr><td>".$row["chaine"]."</td>
                      <td>".$row["h"]."h".$row["m"]."</td>
                      <td>".$row["libelle"]."</td>
                      <td>".$row["type"]."</td></tr>" ;
      ?>
    </table>
  </body>
</html>

<? $db->disconnect() ; ?>
```

Mémorisation de la date

- Pour que l'utilisateur n'ait pas à entrer la date lors de sa prochaine requête, on peut pré-remplir le formulaire de réponse avec les paramètres fournis

```
date : <input type="text" name="jour" size="2"
             value="<?echo $jour?>">
/      <input type="text" name="mois" size="2"
             value="<?echo $mois?>">
/      <input type="text" name="annee" size="4"
             value="<?echo $annee?>">
```

Mémorisation du type d'émission

- De la même manière on peut mémoriser le type d'émission souhaité par l'utilisateur

```
$result = $db->query("SELECT cle FROM type ORDER BY cle") ;
while ($row = $result->fetchRow(DB_FETCHMODE_ASSOC))
    echo "<option value=\"".$row["cle"]."\""
        .( ($row["cle"]==$type) ? " selected" : "" )
        .">".$row["cle"]."</option>" ;
$result->free() ;
```

Contrôle des paramètres

- La date fournie par l'utilisateur étant par nature non fiable, il faut :
  - vérifier qu'elle correspond bien à quelque chose
  - éventuellement l'initialiser

```
$jour = intval($jour) ;
$mois = intval($mois) ;
$annee = intval($annee) ;
if ( !checkdate($mois,$jour,$annee) )
    list($jour,$mois,$annee) = array( date("d"),
                                      date("m"),
                                      date("Y")) ;
```

La forme et le fond

- On veut séparer le contenant du contenu
  - pour pouvoir changer le « look » de l'application sans modifier l'application elle-même
  - pour faire travailler différentes personnes (développeurs, infographistes) en même temps sur un projet
- Les solutions
  - utiliser des feuilles de style (CSS, pas toujours suffisant)
  - adopter un environnement de développement qui offre cette possibilité (Zope, Midgard, …)
  - se servir de « templates »

Création d'un template

```
<html>
  <head><title>__TITRE__</title></head>
  <body>
    <h1>__TITRE__</h1>
    <form action=__ACTION__ method="POST">
      date : <input type="text" name="jour" size="2" value="__JOUR__">
      / <input type="text" name="mois" size="2" value="__MOIS__">
      / <input type="text" name="annee" size="4" value="__ANNEE__">
      <select name="type">
        <option value="">-- tous --</option>
        __TYPES__
      </select>
      <input type="submit" name="go" value="chercher">
    </form>
    <hr>
    <table border="1">
     <tr><th>Chaîne</th><th>Horaire</th><th>Émission</th><th>Type</th></tr>
      __RESULTATS__
    </table>
  </body>
</html>
```

Utilisation d'un template

```
function affichage($titre,$jour,$mois,$annee,$types,$resultats)
{
  // lecture du template dans un tableau
  foreach ( file("modeles/modele4.html") as $ligne )
    {
      // replacement des tokens du template par les valeurs à afficher
      $ligne = str_replace("__TITRE__",$titre,$ligne) ;
      $ligne = str_replace("__ACTION__",$GLOBALS["PHP_SELF"],$ligne) ;

      $ligne = str_replace("__JOUR__",$jour,$ligne) ;
      $ligne = str_replace("__MOIS__",$mois,$ligne) ;
      $ligne = str_replace("__ANNEE__",$annee,$ligne) ;
      $ligne = str_replace("__TYPES__",$types,$ligne) ;
      $ligne = str_replace("__RESULTATS__",$resultats,$ligne) ;
      echo $ligne ;
    }
}
```

Place aux infographistes...

- Une image cliquable pour soumettre les requêtes :

```
<input type="image"
        src="images/submit.png"
```

- Un fond d'écran créé avec Gimp
- Des polices et des couleurs agréables

Plus d’interactivité : JavaScript

- Code embarqué dans les pages HTML...
- …donc exécuté par le client
  - attention à la portabilité !

- Programmation objet événementielle
  - onLoad, onClick, onFocus, onSubmit, …

- Dynamic HTML

JavaScript : exemple 1

```
<html>
  <head>
    <title>Identifiez-vous !</title>
    <script language="JavaScript">
      function TestNom()
      { if (document.ident.nom.value != "")
          return true ;
        alert("Merci de vous identifier avant de continuer.") ;
        return false ;
      }
    </script>
  </head>
  <body>
    <h1>Identifiez-vous !</h1>
    <form action="ident.php"
          name="ident"
          onSubmit="return TestNom()">
      <input type="text" name="nom">
      <input type="submit" name="go" value="continuer">
    </form>
  </body>
</html>
```

JavaScript : exemple 2

```
<html>
  <head>
    <title>Aide</title>
    <script language="JavaScript">
      function Bouton()
      { var popup =window.open("aide.php",
                               "aide_popup",
                               "width=300,height=300");
        popup.Focus() ; }
    </script>
  </head>
  <body>
    <h1>Aide : cliquez sur le bouton</h1>
    <form>
      <input type="button" name="go" value="aide"
             onClick="Bouton() ; return false">
    </form>
  </body>
</html>
```

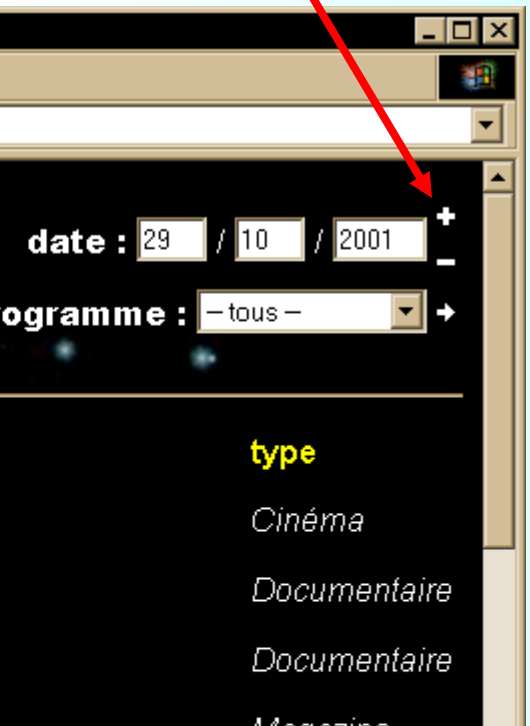
Régler la date facilement

- Une image cliquable :

```
<a href="#" onClick="date_plus();return false">
  <img src="images/up.png" border="0">
</a>
```

- Une fonction Javascript :

```
function date_plus()
{
  var jour = window.document.criteres.jour.value ;
  var mois = window.document.criteres.mois.value ;
  var annee = window.document.criteres.annee.value ;
  jour ++ ;
  if ( jour == 32 )
    { jour = 1 ; mois ++ ;
      if ( mois == 13 )
        { mois = 1 ; annee ++ ; } }
  window.document.criteres.jour.value = jour ;
  window.document.criteres.mois.value = mois ;
  window.document.criteres.annee.value = annee ;
}
```



Rafraîchir la page à chaque réglage

```
<select name="type"

  onChange="window.document.criteres.submit()">



function date_plus()
  {
    /* . . . */
    window.document.criteres.submit() ;
  }
```

Utiliser deux cadres différents

```
<head>
  <frameset rows="150,*">

    <frame name="haut"
           src="__ACTION__?cadre=haut"
           scrolling="no"
           noresize>

    <frame name="bas"
           src="__ACTION__?cadre=bas">

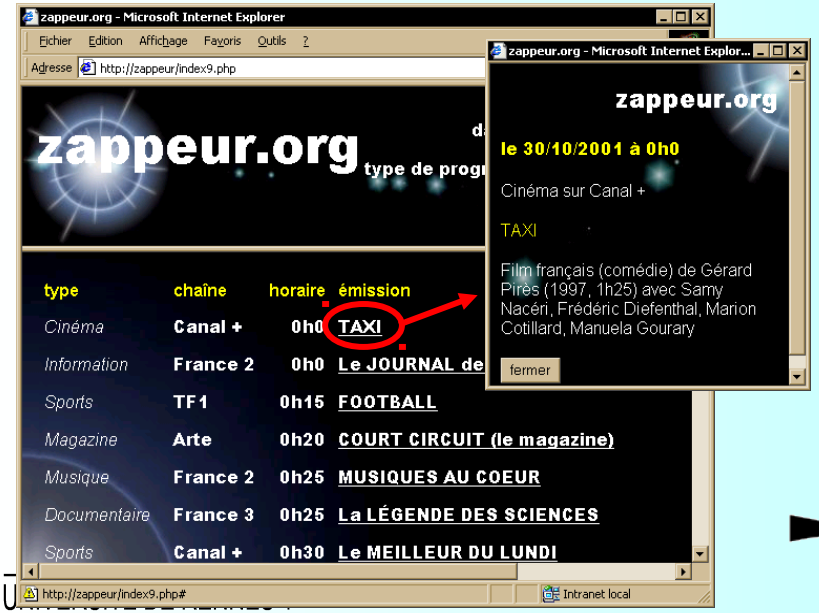
    <noframes>
      Pas de frame, pas de programme !
    </noframes>
  </frameset>
</head>
```

`top.haut.document.criteres.jour.value`

Afficher plus d'informations

- Une fonction JavaScript...

```
function popup(prog)
{ var popup_win =  window.open(
                        '__ACTION__?cadre=popup&prog='+prog,
                        'zappeur_popup',
                        'location=no,toolbar=no') ;
  popup_win.focus() ; }
```



- … déclenchée par un lien :

```
<a href="#"
   onClick="popup('$cle');
            return false">
   $libelle
</a>
```

La consécration : zappeur.com

un nom qui accroche

des réglages qui tuent

un look d'enfer !

un truc qui ne sert à rien mais qui mange du CPU

un sponsor

zappeur.com - Microsoft Internet Explorer

Fichier Edition Affichage Favoris Outils

Adresse http://zappeur.com

zappeur.com

le 1 / 11 / 2001

à partir de 22 h

type de programme : Cinéma

type	chaîne	horaire	émission
Cinéma	Canal +	22h25	L'AFFAIRE MARCORELLE
Cinéma	TF1	00h30	L'AMOUR PROPRE ne le reste
Cinéma	Arte	00h50	Le JOUR SE LÈVE (R)
Cinéma	Canal +	01h00	La NUIT AMÉRICAINE

IRES2001

Terminé

Intranet local

La consécration : zappeur.com

un nom qui accroche

des réglages qui tuent

un look d'enfer !

Hi John !
Got an idea for a new start-up...

un truc qui ne sert à rien mais qui mange du CPU

un sponsor

L’authentification sur le web

- Transport des informations d’authentification
  - protocole HTTP (royaume)
  - cookies
  - SSL (Secure Socket Layout)
- Vérification
  - par le démon HTTP ou par l’application
- Référence
  - fichiers, base de données, codage « en dur »

Authentification sous HTTP

```
GET /prive/index.html HTTP/1.1
Host: zappeur.com
```

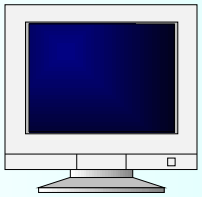
```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Basic realm="private zone"

Vous n'êtes pas autorisé...
```

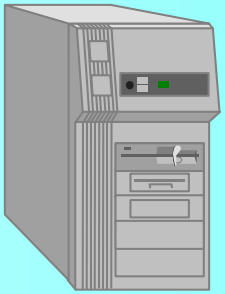
requête

réponse :
erreur 401

client



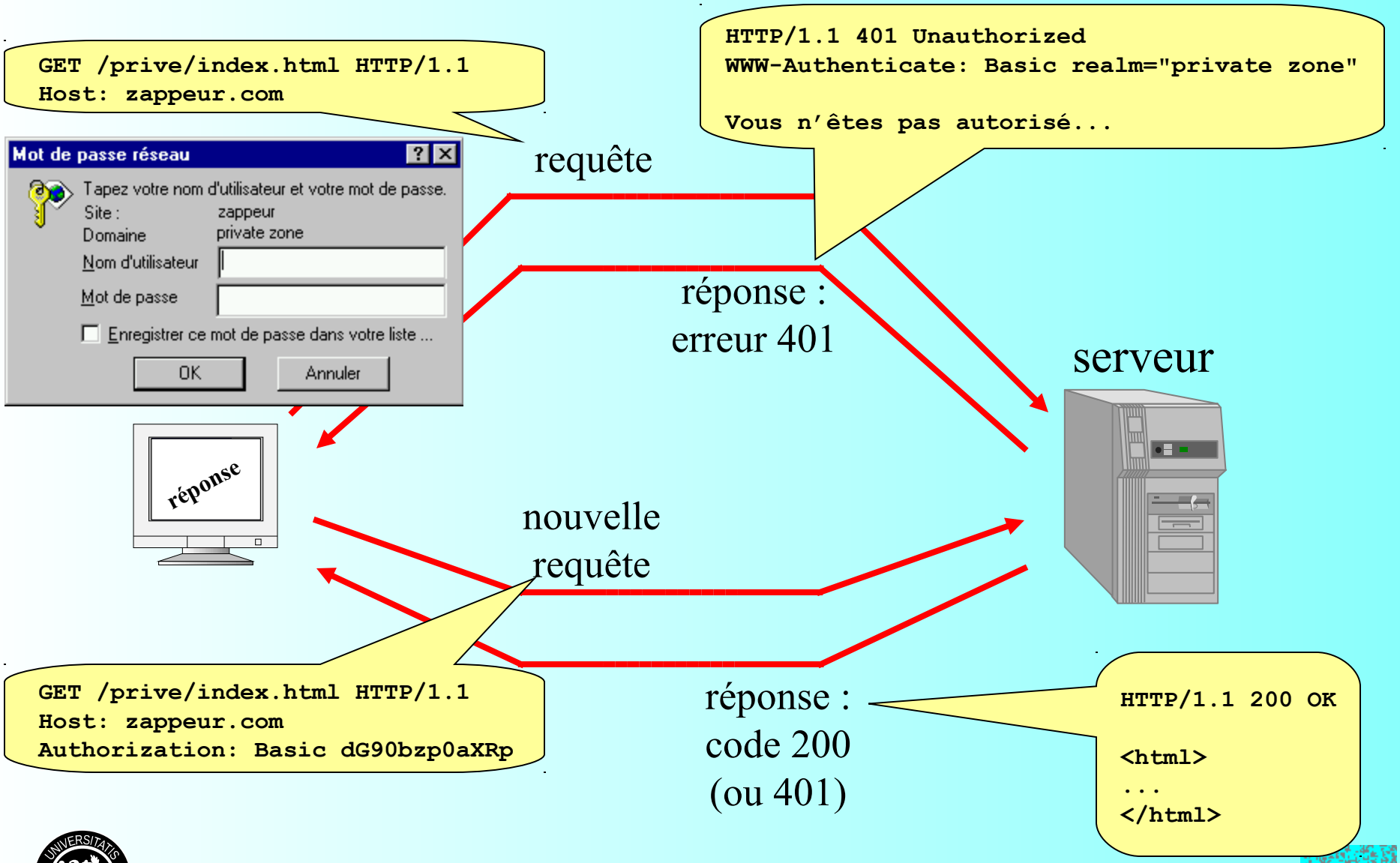
serveur



httpd.conf

```
<Directory "/var/www/html/prive">
  AuthType Basic
  AuthUserFile /etc/prive.users
  AuthName "private zone"
  require valid-user
</Directory>
```

Authentification sous HTTP



Vérification de l'authentification

- Par le démon httpd
  - authentification HTTP simple
    - sur un fichier de mots de passe (htpasswd)
    - sur une base de données (mod_auth_mysql)
    - sur un annuaire (auth_ldap)
  - authentification SSL
    - sur des certificats (X509)
- Par applicatif
  - sur n'importe quoi !

Protéger le site avec HTTP

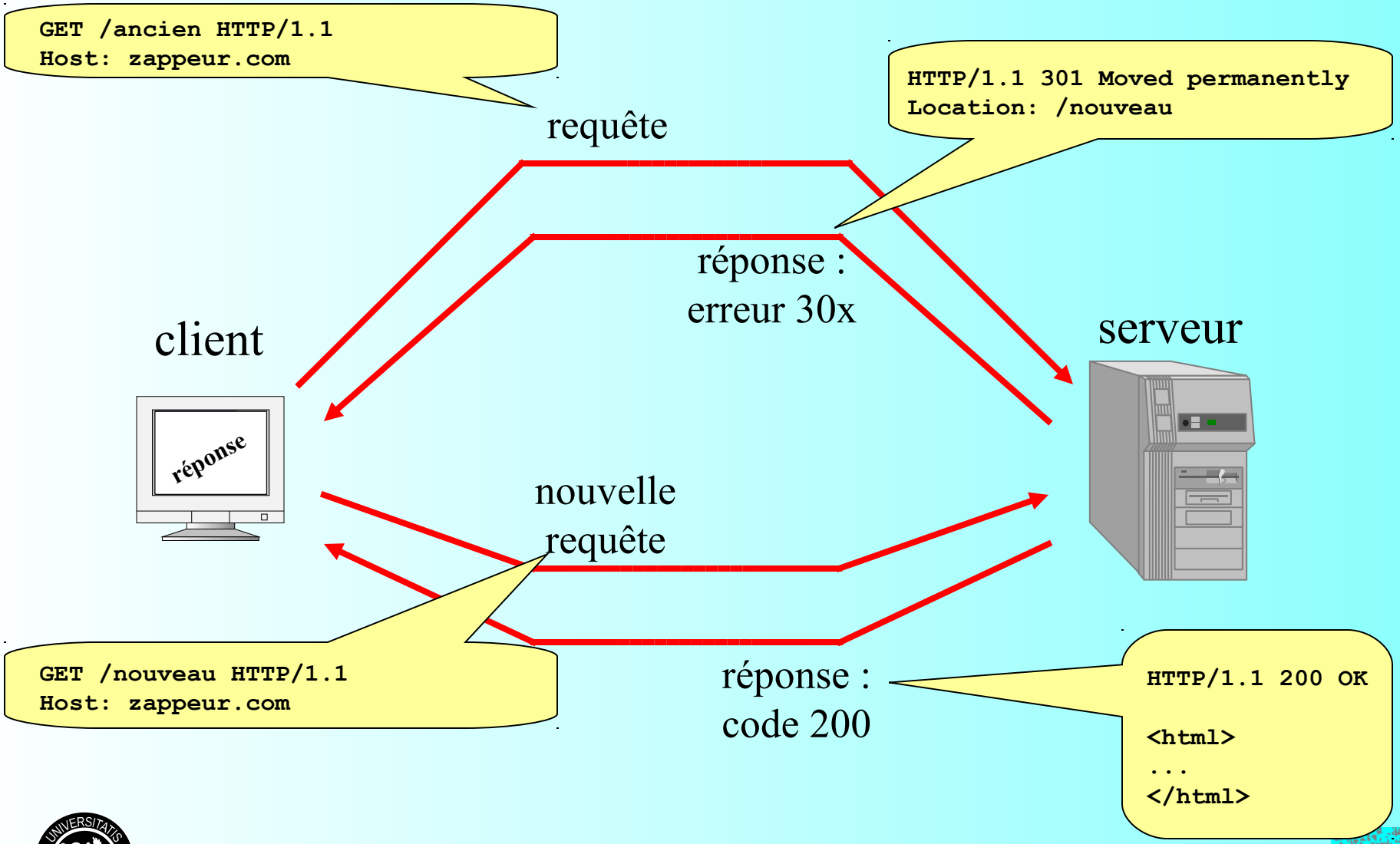
```
if (     ($PHP_AUTH_USER != "titi")
     || ($PHP_AUTH_PW != "toto") )
{
  header('WWW-Authenticate: Basic realm="zappeur"');
  header("HTTP/1.0 401 Unauthorized");
  echo "Vous n'êtes pas autorisé...\n" ;
  exit ;
}

// à partir d'ici, l'utilisateur est authentifié
```

Protéger le site avec HTTPS

- Par le démon httpd
  - vérification du certificat client suivant configuration
- Par applicatif
  - protocole
    - `$SLL_PROTOCOL, $SSL_CIPHER, $SSL_CIPHER_ALGKEYSIZE,`
  - vérification du certificat client
    - `$SSL_CLIENT_CERT, $SLL_CLIENT_CERT_CHAINx, $SSL_CLIENT_VERIFY`
  - informations sur les certificats client et serveur
    - `$SSL_XXX_I_DN_C, $SSL_XXX_I_DN_CN, $SSL_XXX_I_DN_Email, $SSL_XXX_I_DN_L, $SSL_XXX_I_DN_O, $SSL_XXX_I_DN_OU, $SSL_XXX_I_DN_ST, $SSL_XXX_M_SERIAL, $SSL_XXX_M_VERSION, $SSL_XXX_S_DN, $SSL_XXX_S_DN_C, $SSL_XXX_S_DN_CN, $SSL_XXX_S_DN_Email,`

Redirection sous HTTP



Forcer le protocole HTTPS (redirection immédiate)

```
If ( $HTTPS != "on")
{
  header("Location: https://$SERVER_NAME:443$REQUEST_URI");
  exit ;
}

// à partir d'ici, on sait que l'on est sous HTTPS
```

Forcer le protocole HTTPS (redirection différée)

```
If ( $HTTPS != "on")
  {
    $new_url = "https://$SERVER_NAME:443$REQUEST_URI" ;
    header("Refresh: 2;url=$new_url");
    echo "<h2>zappeur.com utilise maintenant
          le protocole HTTPS</h2>
          Vous allez être redirigé vers l'URL
          suivante dans quelques instants :
          <a href=\"$new_url\">$new_url</a>." ;
    exit ;
  }
```

Les cookies

- Ce sont des lignes particulières de l'entête HTTP qui permettent de véhiculer des informations entre client et serveur

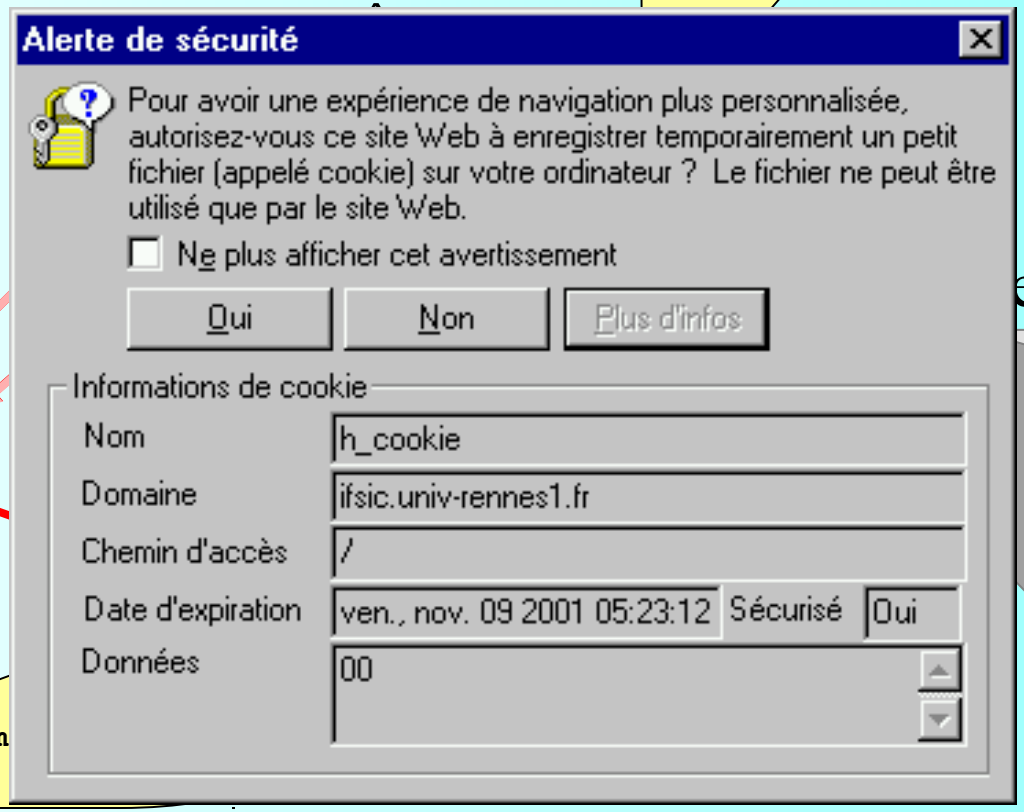
- Caractéristiques :
  - nom (**`UTILISATEUR`**)
  - valeur (**`Pascal.Aubry`**)
  - date de validité (**`Sunday, 04-Nov-01 23:12:40 GMT`**)
  - domaine/serveur (**`www.zappeur.com`**)
  - chemin (**`/public`**)

Les cookies

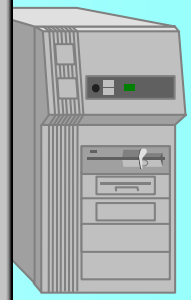
```
HTTP/1.1 200 OK
Set-Cookie: h_cookie=00; \
      path=/rep/; \
      expires=Friday, 09-Nov-01 05:23:12 GMT
```

```
GET /rep/fichier.html HTTP/1.1
Host: zappeur.com
```

client



erveur



```
GET /rep/fichier2.htm
Cookie: h_cookie=00
```

Les cookies (limitations)

- La taille
  - 4Ko par cookie
- Le nombre
  - 300 cookies par navigateur
  - 20 cookies par serveur (pour un même client)
- Les clients peuvent « refuser » des cookies

Mémoriser l'horaire préféré d'un utilisateur entre ses visites

- On envoie un cookie qui va (peut-être) être stocké sur le client et renvoyé à chaque requête

```
// on prend l'ancienne valeur si $h est vide
if ( empty($h) && !empty($h_cookie) )
  { $h = $h_cookie ; }
/* ... */
// on renvoie la nouvelle valeur si elle a changé
if ( $h != $h_cookie )
  SetCookie(
      "h_cookie",                      // nom
      $h,                             // valeur
      time()+604800,                  // validité
      "/",                            // chemin
      ".ifsic.univ-rennes1.fr") ;     // domaine
      1) ;                            // HTTPS
```

Transmettre des variables d'état

- Objectif : propager des variables entre les requêtes
  - identité du visiteur
  - informations de connexion
  - ...
- Le problème
  - HTTP est un protocole sans état
- Les solutions
  - paramètres CGI cachés
  - cookies
  - sessions

Solution 1 : paramètres CGI cachés

- Dans chaque formulaire et chaque lien hypertexte, on ajoute un (des) paramètre(s) caché(s) :

```
<form ...>
   <input type="hidden" name="ident" value="durand">
   <input type="hidden" name="pass" value="xx98yy">
   ...
</form>
```

```
<a href="action.php?ident=durand&pass=xx98yy">texte</a>
```

- Problèmes :
  - Les liens hypertextes et les grosses variables d'état
  - on transmet toutes les variables d'état à chaque requête
  - ça marche, mais ça va bien un moment… ;-)

Solution 2 : cookies

- À chaque changement d'une variable d'état, on envoie un cookie au client qui pourra nous le renvoyer lors de la requête suivante
- Intérêts :
  - on ne transmet les variables que lorsqu'elles changent
  - on ne fait pas de ré-écriture des formulaires et des liens
- Problèmes :
  - nombre et longueur des cookies
  - il faut encore le faire « à la main »
  - certains ne supportent pas les cookies...

Solution 3 : les sessions

- Affectation d'un ID unique
  - pour chaque visiteur non connu (sans ID)
  - de forme aléatoire
- Liaison (identifiant - données sur le serveur)
  - stockage mémoire, disque, base de données, ...
  - format propriétaire, XML (WDDX), ...
- Transmission de l'identifiant
  - par cookie, ré-écriture (automatique)

Transmission des IDs de session

- Ré-écriture automatique :
  - ```
    <form …>
      <input type="hidden"
             name="SESS_ID"
             value="5kj81112yhs3">
      ...
    </form>
    ```
  - `<a href="…?…&SESS_ID=5kj81112yhs3">`
  - `<img src="…?…&SESS_ID=5kj81112yhs3">`
  - `<frame src="…?…&SESS_ID=5kj81112yhs3">`
  - ...
    ```

Tracer la connexion des utilisateurs

```
// on active le mécanisme de session
session_start() ;
// on déclare les variables de session
session_register("nb_req") ;
session_register("debut") ;

$nb_req++ ;
if ( empty($debut) ) $debut = time() ;
// on calcule le temps de connexion
$temps = time() - $debut ;
```

Télécharger des fichiers sur le serveur (uploading)

- Il est parfois intéressant de transmettre les données sous forme de fichiers
  - données de taille importante
  - données déjà stockées sous forme de fichier sur le client
- Un encodage particulier du formulaire
- Une prise en charge par le serveur

Formulaire de téléchargement

```
<form enctype="multipart/form-data"
      action="upload.php"
      method="post">
  <input type="hidden"
         name="MAX_FILE_SIZE"
         value="50000">
  Envoyer ce fichier :
  <input name="fichier" type="file">
  <br>
  <input type="submit"
         value="Envoyer">
</form>
```

Accepter un téléchargement

```
// test pour vérifier que l'utilisateur
// a bien transmis un fichier
if ( $HTTP_POST_FILES["fichier"]["tmp_name"] == "none" )
  return ;

// sauvegarde sous le nom original
// (nom du fichier sur le client)
if ( !move_uploaded_file(
                $tmp_name,
                "repert/". $HTTP_POST_FILES["fichier"]["name"]
                ))
  return ;

// taille du fichier : $HTTP_POST_FILES["fichier"]["size"]
...
```

Y'a pas que HTML sur le web !

```
GET fichier.pdf HTTP/1.1
Host: zappeur.com
```

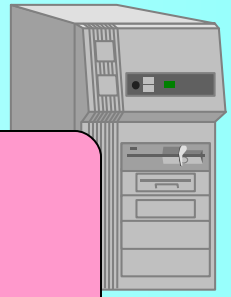
```
HTTP/1.1 200 OK
Content-Type: application/pdf
```

requête

réponse

client

serveur



Lancement du plug-in ou du lecteur externe (Acrobat Reader)

mime.types

application/pdf	pdf
application/vnd.ms-powerpoint	ppt
image/tiff	tiff tif
model/vrml	wrl vrml
text/css	css
video/quicktime	qt mov
...	

type

équence

Créer des images dynamiques

- avec la librairie GD

```
// ouverture de l'image
if (!($im = ImageCreateFromPNG("logos/img")) )
  return ;
// allocation de la couleur d'écriture
$fg = ImageColorAllocate ($im, 255, 255, 255);
// affichage d'un texte
ImageTTFText($im,16,0,10,20,$fg,"comicbd.ttf","texte") ;
// envoi sur la sortie standard
header("Content-Type: image/jpeg") ;
ImageJpeg($im) ;
ImageDestroy($im) ;
```

Renvoyer les programmes au format XML

```
Header("Content-Type: text/xml") ;

echo '<?xml version="1.0" encoding="ISO-8859-1"?>' ;
echo '<racine>' ;
/* … */
echo '</racine>' ;
```

- En bref, on peut renvoyer n'importe quoi !

Une connexion dure...

- … au moins un certain temps !
- Largement le temps pour qu'elle soit interrompue
  - par le client (arrêt du chargement en cours)
  - par le serveur (défaillance quelconque)
  - par le média (panne sur le réseau)
- Il faut s'y préparer
  - transactions avec des bases de données
  - …

Alors, qu’attendez-vous ?

http://perso.ifsic.univ-rennes1.fr/aubry/presentations/jres2001