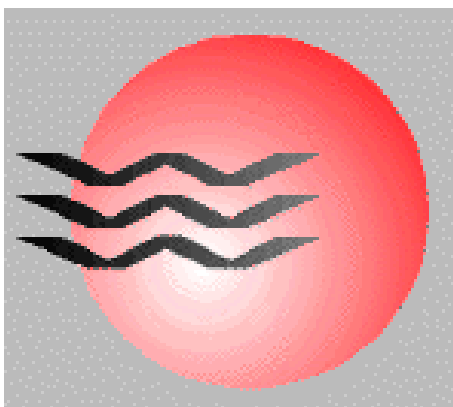




Installation d'un serveur WWW



François Dagorn - CRI Université Rennes I
et
Claude Gross - U.R.E.C. / C.N.R.S.

JRES95 - 23, 24, 25 novembre 1995



Plan

1 - Introduction	3
2 - HTML	4
3 - URIs	42
4 - HTTP	56
5 - CERN httpd	69
6 - NCSA httpd	94
7 - APACHE	109
8 - CGI	114
9 - Harvest	132
10 - Conclusion	138
11 - Conclusion	139



Introduction

⇒ Installation d'un serveur W3

- pourquoi faire?
 - a-t-on quelque chose à dire?
- pour qui?
 - pour quels utilisateurs?
- comment?
 - matériels, logiciels, ...
 - organisation...

HTML

Hypertext Markup Language

HTML

⇒ **HyperText Markup Language**

⇒ **Actuellement version 2**

`http://www.w3.org/pub/WWW/MarkUp/MarkUp.html`

`http://www.sandia.gov/sci_compute/html_ref.html`

HTML

⇒ **HTML et SGML**

HTML est une classe de document *SGML*
(*Standard Generalized Markup Language*)

SGML permet de définir des types de documents structurés

Les principes de bases de SGML sont les suivants :

- une méthode normalisée pour représenter l'information contenue dans un document;
- indépendant des systèmes de saisies et de traitements;
- indépendants de la forme physique finale;
- un principe de balisage logique généralisée;
- ne définit pas de langage de balisage;
- un méta-langage pour construire des langages de balisage

HTML

SGML permet de définir l'ensemble des balises pour identifier les éléments d'un document et les règles formelles qui décrivent sa structure : c'est le rôle de la **DTD** (Document Type Definition)

HTML est une **DTD** SGML

Les versions de HTML :

- HTML version 1;
- HTML version 2 : version actuelle
- HTML version 3 : en cours

HTML

La version 2 de HTML introduit la notion de niveau de conformance :

niveau 0 :

indique le niveau minimal de conformance

niveau 1 :

inclut le niveau 0 + quelques additions comme les images

niveau 2 :

inclut le niveau 0, le niveau 1 + les formulaires

niveau 3 :

tableaux, figures, etc...

niveau 4 :

pour utilisation future

HTML

⇒ HTML et MIME

Le transfert par HTTP d'un document HTML se fait dans un body-part MIME.
HTML a été proposé comme un type MIME

Content-type: text/HTML

Ce type peut avoir 3 paramètres :

Level :

le niveau de conformance

Version :

le numéro de version HTML

Character sets :

pour utilisation future

HTML

HTML est composé d'un certain nombre de balises de la forme :

<balise [liste d'attributs]>...texte...</balise>

Chaque élément HTML est constitué par une balise de début, qui donne le nom de l'élément et ses attributs éventuels, suivi par son contenu et terminé par une balise de fin.

Les balises de début sont délimitées par < et >

Les balises de fin sont délimitées par </ et >

Certains éléments sont vides => pas de contenu et pas de balise de fin
Certains éléments peuvent avoir des attributs

Les règles d'inclusions des balises entre elles sont définies dans la DTD

HTML

⇒ Structure d'un document HTML

Un document HTML est délimité par

`<HTML>` et `</HTML>`

Il est composé d'une entête délimitée par

`<HEAD>` et `</HEAD>`

et d'un corps délimité par

`<BODY>` et `</BODY>`

L'entête donne des informations au sujet du document

Le corps contient l'information utile

HTML

⇒ Exemple :

```
<HTML>
<HEAD>
<TITLE>Exemple de document HTML</TITLE>
</HEAD>
<BODY>
<P>
<B>Ceci est une un document HTML</B>
</P>
</BODY>
</HTML>
```

HTML

⇒ Structure générale

```
<HTML>
  <HEAD>
    <TITLE> titre </TITLE>
    [<BASE>]
    [<LINK>]
  </HEAD>
  <BODY>
    corps du document
  </BODY>
</HTML>
```

HTML

⇒ Éléments de structure du document (obligatoires)

<HTML>...</HTML>
indique que le document contient des éléments HTML
=> head | body

<HEAD>...</HEAD>
contient des éléments HTML décrivant lme document : titre, relations avec d'autres documents...
=> base | isindex | link | meta | nextid | title

<BODY>...</BODY>
contient le texte du document ainsi que ses éléments HTML associés
=> h1 | h2 | h3 | h4 | h5 | h6 | a | img | br | em | strong | code | samp | kbd | var |
cite | tt | b | i | p | ul | ol | dir | menu | dl | pre | xmp | listing | blockquote |
form | isindex | hr | address

HTML

⇒ Éléments de l'entête :

<TITLE> ... </TITLE>
indique le titre du document

<ISINDEX>
indique que le document est indexé

<BASE>
donne l'URL de base du document
ATTRIBUTS :
HREF = "URL"

<LINK>
indique des relations avec d'autres documents

<META>
donne des informations sur le document

HTML

⇒ Éléments de formatage de blocs

<ADDRESS>...</ADDRESS>
utilisé pour donner par exemple le nom et l'adresse de l'auteur d'un document
=> a | img | br | em | strong | code | samp | kbd | var | cite | tt | b | i | p

<BLOCKQUOTE>...</BLOCKQUOTE>
met en évidence un paragraphe
=> h1 | h2 | h3 | h4 | h5 | h6 | a | img | br | em | strong | code | samp | kbd | var |
cite | tt | b | i | p | ul | ol | dir | menu | dl | pre | xmp | listing | blockquote |
form | isindex | hr | address

<H1>...</H1>, ..., <H6>...</H6>
niveaux de sections de 1 à 6
=> | a | img | br | em | strong | code | samp | kbd | var | cite | tt | b | i

HTML

⇒ Éléments de séparations

retour à la ligne

<P>...</P>
paragraphe
| a | img | br | em | strong | code | samp | kbd | var | cite | tt | b | i

<HR>
trait horizontal

HTML

⇒ Éléments de liens

<A>...
permet de définir une ancre origine ou destination
=> h1 | h2 | h3 | h4 | h5 | h6 | a | img | br | em | strong | code | samp |
kbd | var | cite | tt | b | i
Attributs :
HREF = "URL"
NAME = "string"
URN = "URN"
TITLE = "string"
METHODS = "method"

Exemples :

l'UREC

label

HTML

⇒ Éléments de formatage de caractères

...

caractères gras

=> a | img | br | em | strong | code | samp | kbd | var | cite | tt | b | i

<CITE>...</CITE>

pour les citations (italique)

=> a | img | br | em | strong | code | samp | kbd | var | cite | tt | b | i

<CODE>...</CODE>

pour les lignes de codes (fontes à espacement fixe)

=> a | img | br | em | strong | code | samp | kbd | var | cite | tt | b | i

...

mise en évidence du texte (italique)

=> a | img | br | em | strong | code | samp | kbd | var | cite | tt | b | i

HTML

⇒ Éléments de formatage de caractères

<I>...</I>

caractères italiques

=> a | img | br | em | strong | code | samp | kbd | var | cite | tt | b | i

<KBD>...</KBD>

entrée clavier (fontes à espacement fixe)

=> a | img | br | em | strong | code | samp | kbd | var | cite | tt | b | i

<SAMP>...</SAMP>

pour les exemples (fontes à espacement fixe)

=> a | img | br | em | strong | code | samp | kbd | var | cite | tt | b | i

...

mise en évidence du texte (caractères gras)

=> a | img | br | em | strong | code | samp | kbd | var | cite | tt | b | i

HTML

⇒ Éléments de formatage de caractères

`<TT>...</TT>`

fontes à caractères de largeur fixe

=> `a | img | br | em | strong | code | samp | kbd | var | cite | tt | b | i`

`<VAR>...</VAR>`

pour les noms de variables (italique)

=> `a | img | br | em | strong | code | samp | kbd | var | cite | tt | b | i`

HTML

⇒ Éléments de listes

`<DL>... <DT>term<DD>definition...</DL>`

liste descriptive

=> `dt | dd`

`<DIR> ... List item... </DIR>`

liste de répertoire

=> `li`

`<MENU> ... List item... </MENU>`

liste de menu

=> `li`

` ... List item... `

liste numérotée

=> `li`

HTML

⇒ Éléments de listes

` ... List item... `
liste non numérotée
=> `li`

``
item de liste
=> `a | img | br | em | strong | code | samp | kbd | var | cite | tt | b | i | p | ul | ol | dir | menu | dl | pre | xmp | listing | blockquote | form | isindex`

`<DT>...</DT>`
terme à définir
=> `| a | img | br | em | strong | code | samp | kbd | var | cite | tt | b | i`

`<DD>...</DD>`
définition du terme
=> `a | img | br | em | strong | code | samp | kbd | var | cite | tt | b | i | p | ul | ol | dir | menu | dl | pre | xmp | listing | blockquote | form | isindex`

HTML

⇒ Éléments pour les images

``
images intégrées dans le texte
Attributs :
`ALIGN = top | middle | bottom`
`ALT = "string"`
`ISMAP`
`SRC = "URL"`

Exemple :

```
<IMG SRC="http://foo.org.com/pics/pic.gif" ALT=" ">
```

HTML

⇒ Exemple

```
<HTML>
<HEAD>
<TITLE>Exemple de fichier HTML</TITLE>
</HEAD>
<BODY>
<P>
<H1>Exemple de fichier HTML</H1>
</H1>
<P> Ceci est un <i>exemple</i> de fichier <A HREF="http://www.urec.fr/docs/WWW/WWW.html">HTML</A>
</P>
Un fichier HTML contient :
<UL>
<LI> le texte destiné à être lu;
<LI> des instructions de formattage :
<OL>
<LI> niveaux de sections,
<LI> caractères gras, italiques
<LI> listes ...
</OL>
<LI> des liens <A HREF="/home/gross/html/essai.html">hypertexte</A>;
<LI> des incrustations d'images <IMG SRC="http://www.urec.fr/images/www.gif" ALIGN=middle>
</UL>
</BODY>
</HTML>
```

HTML

Exemple de fichier HTML

Ceci est un *exemple* de fichier [HTML](#)

Un fichier HTML contient :

- le texte destiné à être lu;
- des instructions de formattage :
 1. niveaux de sections,
 2. caractères gras, italiques
 3. listes ...
- des liens [hypertexte](#);

- des incrustations d'images



HTML

⇒ Les caractères spéciaux

→ caractères accentués (ISO Latin 1):

- "é" = é
- "à" = à
- ...

`http://www.urec.fr/docs/WWW/ISOlat1.html`

→ caractères particuliers :

- "<" = < ">" = > "&" = &

HTML

⇒ Éléments pour les formulaires

`<FORM>...<INPUT>...<SELECT>...<TEXTAREA>...</FORM>`

formulaire

=> h1|h2|h3|h4|h5|h6|a|img|br|em|strong|code|samp|kbd|var|
cite|tt|b|i|p|ul|ol|dir|menu|dl|pre|xmp|listing|blockquote
|form|isindex|hr|address

Attributs :

ACTION = "URL"

ENCTYPE = MIME type

METHOD = GET|POST

HTML

<INPUT>

champ d'entrée (1 ligne)

Attributs :

ALIGN = top|middle|bottom

CHECKED

MAXLENGTH = integer

NAME = "string"

SIZE = integer

SRC = URL|URN (TYPE = IMAGE)

TYPE = CHECKBOX|HIDDEN|IMAGE|PASSWORD|RADIO|RESET|
SUBMIT|TEXT

VALUE = "string"

SELECT

permet un choix parmi plusieurs valeurs proposées

=> option

Attributs

MULTIPLE

NAME = "string"

SIZE = integer

HTML

⇒ Éléments pour les formulaires

TEXTAREA

zone d'entrée (plusieurs lignes)

=>

Attributs :

COLS = integer

NAME = "string"

ROW = integer

OPTION

permet de définir les valeurs proposées dans SELECT

Attributs :

SELECTED

VALUE = "string"

HTML

⇒ Exemple :

```
<FORM METHOD="post" ACTION="http://www.urec.fr/cgi-bin/post-query">
<P>Votre nom: <INPUT name="name" size="48">
<P>homme <INPUT NAME="sexe" TYPE=radio VALUE=homme>
<P>femme <INPUT NAME="sexe" TYPE=radio VALUE=femme>
<p>
Nombre de personnes dans la famille : <INPUT NAME="family" TYPE=int>
<P>Dans quelles villes avez-vous une r e;sidence ?
<UL>
<LI>Chamb ery <INPUT NAME="chambery" TYPE=checkbox >
<LI>Grenoble <INPUT NAME="grenoble" TYPE=checkbox >
<LI>Autre <br><TEXTAREA NAME="autre" COLS=48 ROWS=4></TEXTAREA>
</UL>
<P ALIGN=CENTER><INPUT TYPE=SUBMIT VALUE=Envoyer> <INPUT TYPE=RESET VA-
LUE=Annule
r>
</FORM>
```

HTML

Exemple de questionnaire

Votre nom: gross claud 

homme femme

Nombre de personnes dans la famille: 4

Dans quelles villes avez-vous une r e;sidence ?

- Chamb ery
- Grenoble
- Autre

Monac 

Envoyer Annuler

HTML3

⇒ En cours de spécification


<http://www.w3.org/pub/WWW/MarkUp/html3-dtd.txt>

<http://www.hpl.hp.co.uk/people/dsr/html/CoverPage.html>

<http://www.ics.uci.edu/pub/ietf/html/index.html>

HTML3

⇒ Tableaux

Address Book			
Name	Telephone	Address	Comments
 Dave Raggett	0454-238122	11 Hunters Mead Hawkesbury Upton nr. Badminton AVON GL9 1BL	Dave lives in a detached house in a 4 year old cul-de-sac in the village of Hawkesbury Upton. The village lies on the uphill edge of the cotswold escarpement and is well known for the nearby monument to a general who fought in the battle of Waterloo

HTML3

⇒ Figures



A pause after a hard spell of fishing © Time-Life 1987

Fill-out forms may include a range of input controls including checkboxes, radio buttons, single and multiline text fields and selection menus. Client side handling of constraints is possible via simple scripts linked to the form. Tables are built up from header or data cells, and you can merge cells for more complex layouts.

This browser is still under development but will be made freely available.



The IETF working group on HTML is

HTML3

⇒ Equations mathématiques

$y = a x^y + b \sin x$	<code><math>y = a x^y + b \sin x</math></code>
$y = a X^y + b \sin x$	<code><math>y = a X^y + b \sin x</math></code>
$\frac{\partial p}{\partial x} = f(x)$	<code><math>∂p∂x = f(x)</math></code>
$y = \int_0^{\infty} \frac{\sin x}{1+x} dx$	<code><math>y = (\int_0^{\infty} (\sin x / (1+x)) dx)</math></code>

HTML3

⇒ **Autres nouveautés :**

- justification (gauche, centre, droite)
- feuilles de style (présentation des documents)
- client-side image map
- ...

HTML

⇒ **Netscaperies, microsofteries,...**

- Extensions propriétaires
- Pressions du marché
- Volonté de monopole

HTML

⇒ Conclusion

- il y a HTML et HTML
- si vous voulez être lu par tous => HTML 2
- utilisez des éditeurs avec vérification syntaxique
- vérifiez vos pages avec des outils de test

HTML

⇒ Les dix commandements :

1. Ayez quelque chose à dire dans vos pages
2. N'essayez pas de créer une nouvelle liste de listes
3. Ne rendez pas la lecture de vos pages dépendante d'un browser particulier
4. N'oubliez pas que certains utilisateurs ont des accès à moins de 28.000 b/s
5. N'utilisez pas le travail des autres sans leur autorisation
6. Reconnaissez le travail fait par d'autres
7. Indiquez les mises-à-jours et les changements sur votre serveur
8. Testez régulièrement la validité des liens dans vos pages
9. Donnez la possibilité à vos lecteurs de vous contacter
10. Même si vous utilisez un éditeur, apprenez HTML

HTML

⇒ Références

→ HTML :

- http://www.sandia.gov/sci_compute/html_ref.html
- <http://www.access.digex.net/~werbach/barebone.html>
- <http://www.kosone.com/people/nelson/nl.htm>
- <http://www.w3.org/pub/WWW/Provider/Style/Overview.html>

→ Editeurs et convertisseurs :

- <http://www.w3.org/pub/WWW/Tools/>
- <http://www.ccs.org/htmledit/index.html>
- <http://union.ncsa.uiuc.edu/HyperNews/get/www/html/editors.html>
- <http://luff.latrobe.edu.au/~medgjw/editors/index.html>



URIs

Uniform Resource Identifier



Les URLs

W3 utilise la technique de l'hypertexte étendue aux réseaux informatiques

Un lien dans un document peut référencer un document sur différents types de serveurs (gopher, ftp...)

L'adresse du document dans un lien doit être non ambiguë

W3 a créé les **Uniform Resource Locator (URL)**

=> schéma de nommage des ressources sur l'Internet

=> extension de la notion de nom de fichier au niveau de l'Internet

Un URL permet d'adresser de façon précise toute ressource accessible sur l'Internet.

`http://info.cern.ch/hypertext/WWW/Addressing/Addressing.html`



Les URLs

Le format d'un URL est :

`<prefixe>:<suffixe>`

Le préfixe indique la méthode d'accès au document => le protocole.

Le format du suffixe dépend du préfixe, en général :

`<protocole>://[user[:password]@]
<machine>[:port][/<path
[#label|?liste paramêtres>]]`

- <protocole> : gopher, ftp, wais...
- <machine> : adresse de la machine
- <port> : numéro de port sur la machine
- <path> : chemin d'accès au document sur la machine



Les URLs

Exemples de préfixe :

- **file** : fichier local sur la machine.
- **ftp** : la ressource est accessible par le protocole **FTP**. Identique à FILE.
- **http** : la ressource est accessible par le protocole **HTTP**. C'est le cas des ressources disponibles sur un serveur W3.
- **telnet** : la ressource est accessible via une session interactive **TELNET**.
- **gopher** : la ressource est accessible par le protocole **GOPHER**.
- **wais** : la ressource est accessible par la version **WAIS** du protocole **Z39.50**.
- **news** : la ressource est accessible par le protocole **NNTP**.
- ...



Les URLs

Exemples :

- file:/usr/local/lib/mailcap
- ftp://ftp.urec.fr/pub
- gopher://gopher.univ-rennes1.fr/
- news:rec.gardening
- http://www.yoyodyne.com:1234/pub/files/foobar.html
- telnet://info.cern.ch
- telnet://www@info.cern.ch
- wais://zenon.inria.fr:210/directory-zenon-inria-fr
- news:fr.comp.infosystemes



Les URLs

Caractères spéciaux dans le <path> :

⇒ **Caractère “%”** :

caractère de contrôle

⇒ **Caractère “#”** :

suivi d'une chaîne de caractères, permet de référencer un "fragment" dans un document. Par exemple :

`http://web.urec.fr/docs/www/web.1.html#HDR 2 8`

permet d'accéder directement à un endroit précis du document .



Les URLs

⇒ **caractère “?”**

suivi d'une chaîne de caractères correspond :

- soit à l'interrogation d'un document indexé,
- soit à une liste de paramètres pour l'exécution d'un programme.

Par exemple :

`wais://quake.think.com:210/directory-of-servers?inria`



Les URLs

⇒ Les URLs relatifs

Un URL relatif consiste soit en un nom complet d'un fichier, soit en un nom relatif au document où l'on se trouve. Il faut entendre par le nom complet d'un fichier, son nom par rapport au répertoire `root` du serveur W3 et non par rapport au répertoire `root` de la machine serveur.

On peut utiliser un URL relatif dans un document pour référencer un autre document localisé sur le même serveur et accessible par le même protocole.

Par exemple, dans le document

```
http://www.yoyodyne.com/pub/afile.html
```

vous pouvez avoir un URL du type

```
/pub/files/bfile.html ou files/bfile.html
```

dont l'équivalent est en absolu :

```
http://www.yoyodyne.com/pub/files/bfile.html
```



Les URIs

⇒ Problèmes :

- beaucoup de protocoles
- beaucoup d'informations
- recherche de documents difficile
- adressage et nom des ressources / machines

⇒ Solution :

- identifiants universelles de ressources

```
http://www.w3.org/pub/WWW/Addressing/Addressing.html
```



URNs

⇒ **Uniform Ressource Name**

⇒ **Identifie une ressource**

⇒ **indépendance par rapport :**

→ protocole

→ machine

⇒ **peut contenir des indications du contenu**

⇒ **attribué par une autorité de nommage :**

→ IBSN

→ ISSN

→ IANA



URNs

⇒ **Syntaxe : quatre champ séparé par ":"**

→ URN:

→ Organisation garant de l'unicité du
nommage

• IANA

• ISBN

• ISSN

→ autorité de nommage

→ chaîne de caractères

⇒ **Exemples :**

<URN:IANA:merit.edu:1929642>

<URN:ISBN:0_201_12:xyzmnopq>

<URN:IANA:12456:1.34.2345>



URNs

- ⇒ Il peut y avoir plusieurs URLs correspondant à un URN

- ⇒ Un URL identifie la localisation d'une instance (un conteneur) d'un ressource identifié par un URN

- ⇒ système proche du DNS



URCs

- ⇒ Uniform Ressource Characteristics
- ⇒ Méthode d'encodage d'informations sur une ressource
- ⇒ suite de couples :

[attribut]:[valeur]

- ⇒ Attributs :

- URN
- URL
- LIFN
- Author
- TTL
- Abstract
- ...



URCs

⇒ Exemple :

```
URN:IANA:626:oit.5674
URL:http://www.gatech.edu/iiir/urc2.paper.html
Content-Type: text/html
Content-Length: 89345
URL:gopher://gopher.gatech.edu:2048/iiir/urc2.paper
Content-Type: text/plain
Content-Length: 4563
```

```
URN:IANA:623:oit:cs:ftp-and-telnet
Title: Intro to FTP and Telnet
Author: Adam Arrowood
URL:file://ftp.gatech.edu/pub/docs/ftp.telnet.ps
Content-Type: text/postscript
Size: 1MB
URL:http://www.gatech.edu/oit/info/ftp.telnet.html
Content-Type: text/html
Size: 600K
Cost: US$0.25
```



HTTP

HyperText Transport Protocol



HTTP

⇒ HyperText Transport Protocol

⇒ Version actuelle : 1.0

⇒ Références :

<http://info.cern.ch/hypertext/WWW/Protocols/Overview.html>

<http://www.ics.uci.edu/pub/ietf/http/>



HTTP

⇒ Principe

```
gross-15 telnet www.univ-rennes1.fr 80
Trying 129.20.254.1 ...
Connected to sir.univ-rennes1.fr.
Escape character is '^]'.
GET /ROR/renater.html
<TITLE> Le r eacute;seau RENATER </TITLE>
<H1> Le r eacute;seau RENATER </H1>

Choisissez un r eseau, vous obtiendrez la carte correspondante.
<P>
<HR>
<A HREF="/cgi-bin/imap/renater"><IMG SRC="/ROR/renater.gif" ISMAP></A> <P>
<HR>

Connection closed by foreign host.
```

- connexion TCP sur le port 80
- requête du document (GET)
- la connexion est coupée



HTTP

⇒ **HTTP version 1** introduit les changements suivants :

- nouvelles méthodes pour obtenir un document
- émission de la version de protocole supportée par le client
- encapsulation du corps du document à transmettre dans des entêtes MIME



HTTP

⇒ **Les messages HTTP**

```
HTTP-message = Simple-Request ; HTTP/0.9 messages
               | Simple-Response
               | Full-Request ; HTTP/1.0 messages
               | Full-Response
```

```
Simple-Request = "GET" URI CRLF
```

```
Simple-Response = [ Entity-Body ]
```



HTTP

⇒ Les requêtes

Full-Request = Method URI HTTP-Version
 *(General-Header ;
 | Request-Header ;
 | Entity-Header) ;
 CRLF
 [Entity-Body]

Full-Response = Status-Line
 *(General-Header
 | Response-Header
 | Entity-Header)
 CRLF
 [Entity-Body]



HTTP

Method = "GET" | "HEAD" | "PUT" | "POST"
 | "DELETE" | "LINK" | "UNLINK"
 | extension-method

HTTP-Version = "HTTP/0.9"|HTTP/1.0

Status-Line = HTTP-Version Status-Code Reason-Phrase CRLF

General-Header = Date
 | Forwarded
 | Message-ID
 | MIME-Version
 | extension-header



HTTP

Request-Header = Accept
| Accept-Charset
| Accept-Encoding
| Accept-Language
| Authorization
| From
| If-Modified-Since
| Pragma
| Referer
| User-Agent
| extension-header

Response-Header = Public
| Retry-After
| Server
| WWW-Authenticate
| extension-header



HTTP

Entity-Header = Allow
| Content-Encoding
| Content-Language
| Content-Length
| Content-Transfer-Encoding
| Content-Type
| Derived-From
| Expires
| Last-Modified
| Link
| Location
| Title
| URI-header
| Version
| extension-header

extension-header = HTTP-header



HTTP

⇒ Les principales méthodes :

GET

pour demander un document
GET conditionnel si le champ *If-Modified-Since* est inclus dans la requête

HEAD

pour obtenir des informations sur un document (titre, dernière mise à jour...)

POST

pour communiquer des données au serveur (=> Entity-Body)

PUT

pour demander que les données (=> Entity-Body) soit stockées dans l'URI indiqué

DELETE

pour demander la destruction du document spécifié par l'URI indiqué



HTTP

```
gross-7 telnet ns.urec.fr 80
Trying 134.157.4.14 ...
Connected to ns.urec.fr.
Escape character is '^]'.

GET /info.html HTTP/1.0
```

```
HTTP/1.0 200 OK
Date: Thursday, 27-Apr-95 13:42:34 GMT
Server: NCSA/1.3
MIME-version: 1.0
Content-type: text/html
Last-modified: Thursday, 27-Apr-95 13:41:56 GMT
Content-length: 459
```

```
<HTML><HEAD><TITLE>L'UREC</TITLE></HEAD>
<BODY>
<p>
<HR>
<H1>L'Unité; Réseaux du CNRS</A>
</H1>
<HR><UL>
<LI> <A HREF="urec.html"><b> Ses missions</b></A>
<LI> <A HREF="acces.html"><b> Son accès</b></A>
<LI> <A HREF="urec.personnel.html"><b> Son personnel</b></A>
<LI> <A HREF="/Ftp/publis_urec"><b> Ses publications</b></A>
</UL><HR>
<ADDRESS><A HREF="/annuaires/gross.html">CG</A></ADDRESS></BODY></HTML>
Connection closed by foreign host.
```



HTTP

```
gross-7 telnet ns.urec.fr 80
Trying 134.157.4.14 ...
Connected to ns.urec.fr.
Escape character is '^]'.

```

```
POST /cgi-bin/Mail HTTP/1.0
Accept: text/plain
Accept: application/x-html
Accept: application/html
Accept: application/postscriptAccept: */*
User-Agent: NCSA Mosaic for the X Window System/2.4
Content-type: application/x-www-form-urlencoded
Content-length: 50
name=gross&from=gross@imag.fr&sub=essai&body=essai

```

```
HTTP/1.0 200 OK
Date: Thursday, 27-Apr-95 14:28:39 GMT
Server: NCSA/1.3
MIME-version: 1.0
Content-type: text/html

```

```
<HTML><HEAD><TITLE>Message</TITLE></HEAD>
<BODY><H1>Message</H1>
<P>Message envoyé; &grave; <B>webmaster@urec.fr</B>:<BR><BR></P>
<PRE>
<B>Sujet</B>:
<B>De</B>: &lt;></PRE>
</BODY></HTML>
Connection closed by foreign host.

```



HTTP

```
gross-7 telnet ns.urec.fr 80
Trying 134.157.4.14 ...
Connected to ns.urec.fr.
Escape character is '^]'.
HEAD /index.html HTTP/1.0

```

```
HTTP/1.0 200 OK
Date: Friday, 28-Apr-95 10:14:42 GMT
Server: NCSA/1.3
MIME-version: 1.0
Content-type: text/html
Last-modified: Thursday, 06-Apr-95 15:16:06 GMT
Content-length: 2062

```

Connection closed by foreign host



CERN httpd

CERN httpd



CERN httpd

⇒ La version courante est *CERN httpd 3.0*.

⇒ Une documentation en ligne est accessible :

<http://www.w3.org/pub/WWW/Daemon/>

⇒ Une livraison *binaire* est disponible pour un grand nombre d'architectures :

Tableau 1 :

Plateforme	Next	Next 386	Sun	Sun	HP	SGI	DEC	DEC	IBM	OSF	PC	SCO
Version	3.2	3.2	4.1.3	2.3	9.0	5.2		4.3	3.2	2.0/3.0	1.1.29	3.0/3.2
Binaire	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
OS	Nextstep	Nextstep	SunOs	Solaris	HP-UX /Snake	IRIX	VMS	Ultrix	Aix	OSF/1	Linux	SCO



CERN httpd :

⇒ Les différentes versions *binaires* sont disponibles par :

`http://www.w3.org/pub/WWW/Library/User/Platform/`

⇒ Il est intéressant de compiler localement *CERN httpd* pour profiter de la fonctionnalité passerelle WAIS.

⇒ Les sources sont disponibles par :

`ftp://ftp.w3.org/pub/httpd/httpd_3.0_src.tar.Z`

⇒ Il est aussi nécessaire d'installer la *Library of Common Code* :

`ftp://ftp.w3.org/pub/libwww/libwww_2.17_src.tar.Z`



CERN httpd :

⇒ Après avoir décompressés et *détarés* les fichiers transférés, il convient de procéder comme suit :

- décommenter les lignes à propos de WAIS dans le fichier

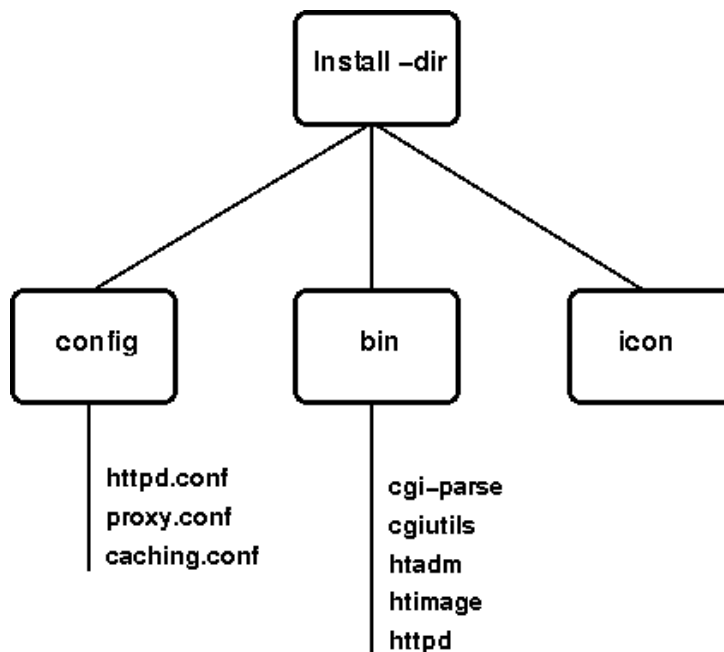
`/src/WWW/CERN/All/<plate-forme>/Makefile.include`

```
#
# WHEN COMPILING WITH DIRECT WAIS SUPPORT:
#
# Uncomment these six lines (and edit them, if necessary).
WAIS = /src/reseau/wais/freeWAIS-0.3
WAISLIB = $(WAIS)/bin/client.a $(WAIS)/bin/wais.a
MATHLIB = -lm
WAISINC = -I$(WAIS)/ir
WAISCFRAGS = -DDIRECT_WAIS
HTWAIS = $(WTMP)/Library/$(WWW_MACH)/HTWAIS.o
```

- exécuter la commande *BUILD* se trouvant dans `/src/WWW/CERN`.

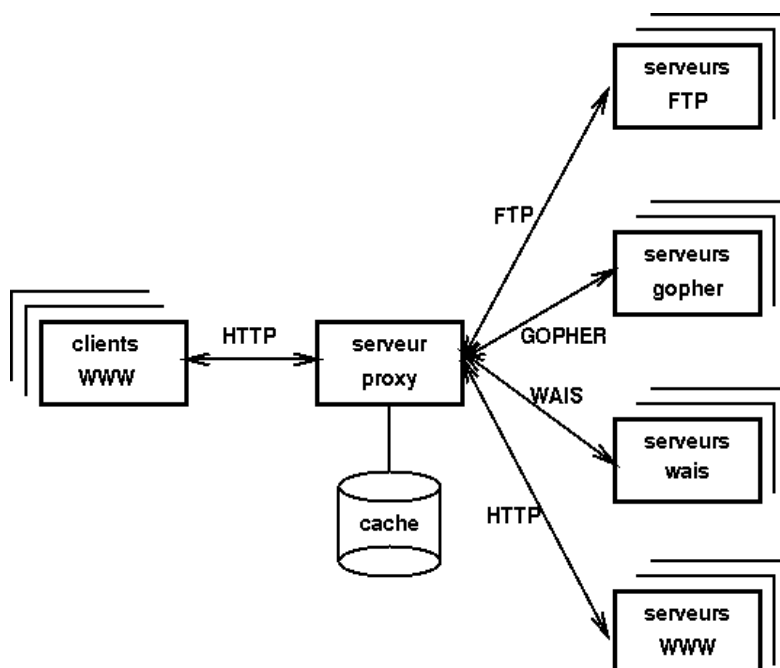
CERN httpd :

⇒ après l'installation



CERN httpd

⇒ Proxy, caching...





CERN httpd :

⇒ Un serveur proxy s'intercale entre des clients W3 et des serveurs d'information utilisant divers protocoles, il sert de relai.

⇒ L'utilisation d'un proxy est intéressante dans les cas suivants :

- pour permettre à certains clients d'accéder à des protocoles qu'ils ne connaissent pas (WAIS et les clients W3 sur micros ...) ;
- installé sur une machine pare-feu un proxy procure l'accès à l'Internet aux clients situés derrière lui ;
- l'ensemble des documents distants accédés par un proxy peuvent être stockés dans un cache. Avant d'effectuer une requête, le proxy s'assure que le document demandé n'a pas déjà été accédé et stocké dans le cache. Le partage entre plusieurs clients W3 d'un serveur proxy peut diminuer considérablement le temps d'accès aux documents fortement demandés.



CERN httpd :

⇒ Il est en général très simple d'indiquer à un client W3 qu'il doit utiliser les services d'un *proxy* :

- pour les clients fonctionnant dans un environnement Unix (Mosaic pour X, Lynx, ...) :

```
setenv http_proxy "http://www.univ-rennes1.fr:8080/"  
setenv ftp_proxy "http://www.univ-rennes1.fr:8080/"  
setenv gopher_proxy "http://www.univ-rennes1.fr:8080/"  
setenv wais_proxy "http://www.univ-rennes1.fr:8080/"
```

- concernant les clients de la famille *Netscape*, il convient d'utiliser le menu *preferences* et l'option *Configure Proxies* ;
- les versions pour Mac et Pc de *Mosaic* ne connaissent pas du tout la configuration de *proxies*.



CERN httpd :

⇒ Dans le cas où un serveur *proxy* fonctionne en complément d'un serveur *classique*, il est possible d'indiquer aux clients de ne solliciter le *proxy* que pour les accès extérieurs :

```
setenv no_proxy "univ-rennes1.fr"
```

⇒ Dans ce cas la fonctionnalité *proxy* est invalidée pour tous les protocoles. Cette solution est mise en oeuvre au CRI. *no_proxy* n'est pas compris par *xmosaic 2.x*, est par contre intégré dans toutes les versions de *Netscape*.



CERN httpd :

⇒ Au démarrage, le serveur *http* consulte le fichier */etc/httpd.conf*. S'il n'existe pas, il est possible de le préciser au lancement :

```
httpd -r /votre/propres/httpd.conf
```

⇒ Des exemples de configuration sont installés dans le répertoire *InstallDir/config*. Le fichier *caching.conf* est une base de départ permettant de mettre en route un *proxy* serveur avec utilisation d'un cache des documents accédés.



CERN httpd :

⇒ Dans le fichier de configuration de *httpd*, on peut préciser par des directives, les paramètres suivants :

- configuration générale (*ServerRoot*, *\dots*) ;
- traitement des URL (accès, nommage, ...);
- les suffixes des fichiers servis ;
- les logs ;
- proxy, caching ;
- vision *icônique* des répertoires.

⇒ Un seul fichier permet de personnaliser le fonctionnement de *httpd*.



CERN httpd :

```
ServerRoot /usr/local/cern-http
Port 8080
Userid nobody
Groupid nogroup
#
AccessLog /var/log/cern-log
ErrorLog /var/log/cern-errors
LogFormat Common
LogTime LocalTime
#
Pass http:*
Pass ftp:*
Pass gopher:*
Pass wais:*
Pass /* /var/www/*
#
Caching On
CacheRoot /var/www/proxy-cache
CacheSize 50
CacheClean * 2 months
#
CacheUnused http:* 2 weeks
CacheUnused ftp:* 1 week
CacheUnused gopher:* 1 week
CacheDefaultExpiry ftp:* 10 days
CacheDefaultExpiry gopher:* 2 days
#
Gc On
GcDailyGc 3:00
```



CERN httpd :

⇒ Par défaut, seules les méthodes *GET*, *HEAD* et *POST* sont autorisées. Les directives *Enable* et *Disable* permettent de personnaliser :

`Enable PUT`

⇒ La méthode *PUT* permet de mettre à jour des URL (cf. HTTP), elle n'est pas mise en oeuvre par le serveur du CERN. Des solutions *locales* sont possibles :

`PutScript /un/code/qui/fait/le/travail`

⇒ Il est possible de configurer *httpd* de façon à donner accès aux répertoires utilisateurs par le biais d'URL `/\~{utilisateur}` :

`UserDir repertoire`



CERN httpd :

⇒ L'accès aux URL est réalisé par les directives *MAP*, *PASS*, *FAIL*, *Exec* et *Protect*.

⇒ La directive *Map* permet d'effectuer des traitements sur les chaînes de caractères constituant un URL :

- `Map file:* ftp:*` => les URL débutant par *file:* seront traités comme s'il s'agissait d'URL *ftp* ;
- `Map */ici/* */ailleurs/*` => la chaîne *ici* est remplacée par la chaîne *ailleurs*.



CERN httpd :

⇒ La directive **Pass** permet de préciser quels sont les URL servis par *httpd* :

- Pour configurer un serveur *proxy* pour un protocole donné :

Pass wais:* (tous les URL débutant par *wais:* seront traités ;

Pass http:*

- Fixer quels sont les répertoires locaux *traités* par *httpd* :

Pass /* /var/www/*

⇒ L'ordre dans lequel sont présentées les directives *Map* et *Pass* est important ...



CERN httpd :

⇒ Le fichier de configuration de *httpd* peut contenir des directives pour contrôler l'accès aux répertoires. L'accès peut être contrôlé par :

- utilisateur ;
- adresse IP.

⇒ les protections par utilisateur impliquent la création d'utilisateurs et groupes spécifiques (rien à voir avec Unix ...) ;

⇒ on définit des modèles de protection, on applique le modèle aux répertoires ciblés.



CERN httpd :

```
Protection POTES {
    Mask @(*.potes.fr)
}
Protect /labos/* POTES
#
#     ici seules les machines du domaine potes.fr sont autorisées à consulter le répertoire /labos.
#
Protection PROXY-PROT {
    Mask @(*.univ-rennes1.fr, ...)
}
Protect http:* PROXY-PROT
Protect ftp:* PROXY-PROT
Protect gopher:* PROXY-PROT
#
```



CERN httpd :

⇒ Un serveur *proxy* peut être configuré pour cacher les documents accédés. Le cache est mis en route en utilisant les directives suivantes :

```
Caching On
CacheRoot /Un/repertoire/pour/le/cache
```

⇒ Il est possible de stipuler des URL qui ne seront jamais cachés (cacher tout sauf) ou d'indiquer des URL qui le seront (ne rien cacher sauf) :

```
Nocaching http://tres.mauvais.fr/*
CacheOnly http://super.bien.fr/*
```



CERN httpd :

⇒ La taille du cache est exprimée en mégas octets (5 par défaut), la directive **CacheSize** permet de fixer sa taille :

CacheSize 100 M

⇒ La durée de vie des documents dans le cache est paramétrée par la directive **CacheClean** :

CacheClean http:* 1 month

CacheClean ftp:* 15 days

CacheClean gopher:* 4 days 3 hours Un processus de nettoyage (*garbage collector*) est activé à intervalles réguliers (GcDaillyGc 3:00).

⇒ Les documents qui n'ont pas été accédés depuis ... peuvent être enlevés du cache par le processus de nettoyage :

CacheUnused http:* 7 days

CacheUnused ftp://ftp.univ-rennes1.fr/* 15 days



CERN httpd

⇒ Les serveurs *http/1.0* peuvent fournir l'en-tête HTTP *Last-Modified* (l'en-tête *Expires* est très rarement délivrée), MAIS :

- des documents ne sont pas servis par *httpd* et donc pas de *Last-Modified* ;
- tous les serveurs *http/1.0* ne fournissent pas *Last-Modified* ;
- les *cgi scripts* ne fournissent pas *Last-Modified*.

⇒ Une valeur par défaut **CacheDefaultExpiry** permet de contourner ces anomalies.

CacheDefaultExpiry ftp:* 1 month

CacheDefaultExpiry gopher:* 10 days



CERN httpd:

⇒ Compte tenu des *cgi scripts*, il est préférable de laisser pour *http*, la valeur par défaut à zéro. Donc tous les documents servis par *http* qui ne fournissent pas *last-modified* sont systématiquement accédés depuis leur URL d'origine.



CERN httpd :

⇒ Pour déterminer si un document doit être enlevé du cache, un facteur est appliqué à partir de la date de dernière modification :

`CacheLastModifiedFactor : 0.5`

⇒ Dans ce cas un document présent dans le cache, dont la date de dernière modification remonte à 20 jours, sera effacé au bout de 10 jours de présence. La valeur par défaut est 0.1.

⇒ Un *log* des accès au cache peut être géré séparément :

`CacheAccessLog /un/repert/pour/le/log/du/cache`



CERN httpd : config

⇒ La gestion du cache est opérée de la façon suivante :

- un *garbage collector* fait le ménage toutes les nuits ;
- la durée maximale de vie d'un document dans le cache est bornée par `Cache-Clean` ;
- les documents non accédés depuis `CacheUnused` sont enlevés du cache ;
- une date d'expiration est estimée par la conjonction de `Last-Modified`, `CacheDefaultExpiry` et de `CacheLastModifiedFactor`.



CERN httpd : les cgi scripts

⇒ L'endroit où sont situés les Scripts exécutables est précisé par la directive *Exec* dans le fichier de config :

```
Exec /cgi-bin/* /usr/local/cern-http/bin
```

⇒ Attention à la localisation des directives *Exec*, les placer avant la directive *Pass* /*

⇒ La commande *cgiparse* est fournie avec le serveur, elle permet de traiter les paramètres ... mais *cgi-lib.pl* fonctionne également.



CERN httpd : les images réactives

⇒ *htimage* est un programme livré dans le répertoire *bin*. Il reçoit en paramètre la localisation d'un fichier donnant le découpage d'une image :

```
<A HREF="/cgi-bin/htimage/maconfig.conf"> <IMG SRC="/images/Image.gif" ISMAP></A>
```

⇒ *maconfig.conf* situé (dans cet exemple) dans le répertoire *bin* autorise la définition de cercles, rectangles et polygones :

```
default /images/toto.html  
circle(10,20) 20 /images/tutu.html  
rectangle (50,100) (100,200) ftp://tata.truc.fr/qqc
```



NCSA httpd

NCSA
HTTPd



NCSA httpd

⇒ La version courante est NCSA httpd 1.4.

⇒ Une documentation en ligne est accessible :

`http://hoohoo.ncsa.uiuc.edu/docs/Overview.html`

⇒ Une livraison binaire est disponible pour un grand nombre d'architectures :

Tableau 2 :

Plateforme	Sun	Sun	HP	SGI	DEC	DEC	IBM	PC
OS	SunOS	Solaris	HP-UX/ Snake	IRIX	Ultrix	OSF/1	AIX	Linux
Version	4.1.3	2.3/2.4	9.05	4.0.5/5.2	4.0	3.0	3.2.5	1.2.8
Binaire	OUI	OUI	OUI	OUI	OUI	OUI	OUI	OUI



NCSA httpd

⇒ **Compilation :**

→ Récupérer les sources :

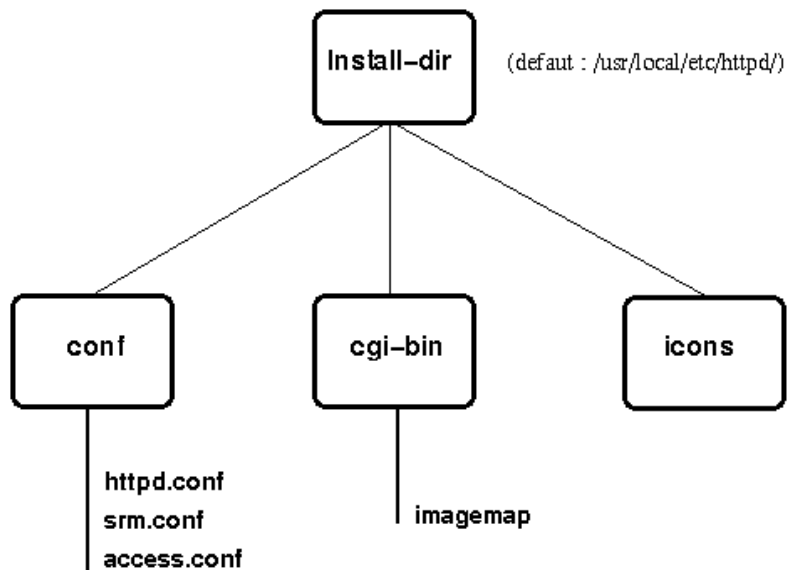
`ftp://ftp.ncsa.uiuc.edu/Web/httpd/Unix/ncsa_httpd/current/`
`ftp://ftp.urec.fr/pub/reseaux/services_infos/WWW/ncsa/httpd/Unix/ncsa_httpd/current/`

→ Modifier les fichiers Makefiles et src/httpd.h



NCSA httpd

⇒ Installation :



NCSA httpd

⇒ httpd.conf :

- User
- Group
- ServerAdmin
- ServerRoot
- ServerName
- StartServers
- MaxServers
- TimeOut
- ErrorLog
- TransferLog
- AgentLog
- RefererLog
- RefererIgnore
- PidFile
- AccessConfig
- ResourceConfig
- TypesConfig
- IdentityCheck



NCSA httpd

⇒ Exemple :

```
ServerType standalone
Port 80
StartServers 3
MaxServers 5
User nobody
Group nobody
ServerAdmin webmaster@imag.fr
ServerRoot /usr/local/etc/httpd
ErrorLog logs/error_log
TransferLog logs/access_log
AgentLog logs/agent_log
RefererLog logs/referer_log
PidFile logs/httpd.pid
ServerName www.imag.fr
```



NCSA httpd

⇒ srm.conf :

- DocumentRoot
- UserDir
- DirectoryIndex
- AccessFileName
- AddType
- AddEncoding
- DefaultType
- Redirect
- Alias
- ScriptAlias
- OldScriptAlias
- FancyIndexing
- DefaultIcon
- ReadmeName
- HeaderName
- AddDescription
- AddIcon
- AddIconByType
- AddIconByEncoding
- IndexIgnore
- IndexOptions
- ErrorDocument



NCSA httpd

⇒ Exemple :

```
DocumentRoot /usr/local/etc/httpd/htdocs
UserDir public_html
DirectoryIndex index.html
FancyIndexing on
AddIcon /icons/binary.gif .bin .exe
AddIcon /icons/binhex.gif .hqx
AddIcon /icons/tar.gif .tar
...
AddIcon /icons/back.gif ..
AddIcon /icons/hand.right.gif README
AddIcon /icons/folder.gif ^^DIRECTORY^^
AddIcon /icons/blank.gif ^^BLANKICON^^
AddIconByType (TXT,/icons/text.gif) text/*
...
AddIconByType (MOV,/icons/movie.gif) video/*
DefaultType text/plain
DefaultIcon /icons/unknown.gif
HeaderName README
IndexIgnore */.* *~ *# */HEADER* */README*
AccessFileName .htaccess
AddEncoding x-compress Z
AddEncoding x-gzip gz
Alias /icons/ /usr/local/etc/httpd/icons/
ScriptAlias /cgi-bin/ /usr/local/etc/httpd/cgi-bin/
AddType text/x-server-parsed-html .shtml
AddType application/x-httpd-cgi .cgi
ErrorDocument 302 /cgi-bin/redirect.cgi
ErrorDocument 500 /errors/server.html
ErrorDocument 403 /errors/forbidden.html
```



NCSA httpd

⇒ access.conf :

- Directory
- Options
- AllowOverride
- AddType
- DefaultType
- AddEncoding
- AddDescription
- AddIcon
- IndexIgnore
- DefaultIcon
- ReadmeName
- AuthName
- AuthType
- AuthUserFile
- AuthGroupFile
- Limit



NCSA httpd

⇒ Exemple :

```
<Directory /usr/local/etc/httpd/htdocs>
Options Indexes SymLinksIfOwnerMatch Includes ExecCGI
AllowOverride All
<Limit GET>
order allow,deny
allow from all
</Limit>
</Directory>
<Directory /usr/local/etc/httpd/htdocs/imag>
<Limit GET>
order deny,allow
deny from all
allow from .imag.fr
</Limit>
</Directory>
<Directory /usr/local/etc/httpd/htdocs/privé>
AuthType Basic
AuthUserFile /usr/local/etc/httpd/conf/.htpasswd
AuthGroupFile /usr/local/etc/httpd/conf/.htgroup
AuthName Groupe Systeme
<Limit GET>
require group systeme
</Limit>
</Directory>
```



NCSA httpd

⇒ Sécurité :

- accès par domaines

- accès par utilisateurs : login + mot de passe

- configuration dans :
 - dans fichier access.conf
 - dans fichier .htaccess

⇒ Tutorial : <http://hoohoo.ncsa.uiuc.edu/docs/tutorials/user.html>



NCSA httpd

⇒ Server Side Include : directives dans les fichiers HTML

→ Pour activer cette fonctionnalité :

- option `includes` pour le repertoire dans le fichier `access.conf`
- définir dans le fichier `httpd.conf` le suffixe associé au type MIME
text/x-server-parsed-html

→ Exemples :

```
<!--#include virtual="/divers/intro.html" -->
<!--#include file="intro.html" -->
<!--#exec cmd="/usr/bin/date" -->
<!--#exec cgi="/cgi-bin/date" -->
<!--#echo var="LAST_MODIFIED" -->
```

⇒ Tutorial : <http://hoohoo.ncsa.uiuc.edu/docs/tutorials/includes.html>



NCSA httpd

⇒ Images cliquables :

- avoir le programme `imagemap` compilé
- avoir une image au format GIF, par exemple *essai.gif* dans le repertoire `/test`
- créer le fichier `essai.map`

Syntaxe :

type URL coordonnées

Avec :

- type = [default|rect|poly|circle|point]
- coordonnées : dépend du type



NCSA httpd

⇒ Images cliquables

Exemple de fichier de configuration

```
default /examples/none.html
circle /examples/ow.html 313,28 313,44
circle /examples/ow.html 382,193 383,200
poly /examples/fish.html 298,93 251,26 300,0 453,0 511,48 511,101 474,65 420,55
358,88
poly /examples/fish.html 349,196 350,233 406,221 444,195 455,214 470,181 418,150
poly /examples/plant.html 117,96 116,267 172,283 192,299 247,254 242,101
poly http://hoohoo.ncsa.uiuc.edu/examples/pillar.html 11,0 26,225 18,261 83,270
109,264 110,97 105,0
poly /examples/floor.html 0,383 82,383 82,271 2,267
rect /examples/post.html 83,284 180,383
circle /examples/post.html 202,87 202,71
poly /examples/rail.html 175,320 227,383 347,268 345,166
poly /examples/stairs.html 223,383 371,261 511,341 511,383
poly /examples/floor.html 389,270 511,336 511,124 439,136 411,258
rect /examples/post.html 336,119 436,261
rect /examples/post.html 397,89 436,265
poly /examples/gold.html 307,199 245,177 239,115 511,91 511,121 292,144 346,166
```



NCSA httpd

⇒ Images cliquables

→ créer un fichier HTML avec :

```
<A HREF="/cgi-bin/imagemap/test/essai.map"><IMG="/test/essai.gif"
ISMAMP></A>
```

⇒ Tutorial : <http://hoohoo.ncsa.uiuc.edu/docs/tutorials/imagemapping.html>

⇒ A récupérer : [mapedit](http://sunsite.unc.edu/boutell/mapedit/mapedit.html)

(<http://sunsite.unc.edu/boutell/mapedit/mapedit.html>)

APACHE



APACHE

- ⇒ Apache (A PAtCHy server) est un projet de volontaires qui se donnent pour but de produire un serveur HTTP du domaine public ;
- ⇒ Utilise la même syntaxe que NCSA 1.3 pour les fichiers de configuration (*plug-in replacement*) ;
- ⇒ corrige les nombreux trous de sécurité introduit par NCSA1.2 et NCSA 1.3 ;
- ⇒ Apporte des fonctionnalités supplémentaires.



APACHE

⇒ Documentation accessible par <http://www.apache.org/>.

⇒ Parmi les fonctionnalités supplémentaires :

- existence d'une interface de programmation permettant d'ajouter des fonctionnalités *maison* au serveur ;
- personnalisation des messages d'erreurs HTTP ;
- gestion automatique du multi-langage en traitant les en-têtes HTTP *Content-Language* (serveur) et *Accept-Language* (client) ;



APACHE :

⇒ config multi-langage dans le fichier *srm.conf* :

```
AddLanguage en .en  
AddLanguage fr .fr  
LanguagePriority fr en
```

⇒ Par défaut le serveur émettra les fichiers suffixés par *.fr*.

⇒ Pour recevoir prioritairement un fichier suffixé par *.en* un client doit émettre l'en-tête HTTP *Accept-Language: en*. Mais quels sont les clients qui émettent *Accept-Language* ?

APACHE :

⇒ erreurs HTTP dans le fichier *srm.conf* :

ErrorDocument 401 /erreurs/401.html
ErrorDocument 404 /erreurs/404.html
ErrorDocument 500 /erreurs/500.html



Damned !

Désolé, le document que vous avez demandé n'existe pas sur ce serveur !

Sorry, the requested document does not exist on this server !



CGI

Common Gateway Interface



CGI

⇒ Les formulaires électroniques

Les clients WWW sont en général utilisés pour accéder à des documents diffusés par des serveurs d'informations.

En sélectionnant une ancre dans un texte, l'utilisateur déclenche le transfert de données dans le sens serveur vers client ...

Les clients WWW peuvent également servir à communiquer des informations dans l'autre direction et être utilisés pour transmettre des paramètres à un serveur HTTP.

L'émission de formulaires électroniques ou la présentation d'images réactives utilisent ces propriétés.



CGI

Depuis l'émission d'un formulaire jusqu'à l'accusé de réception du traitement par un serveur HTTP, les protagonistes sont les suivants :

- HTML (HTML-2.0 niveau 2) qui propose la gestion de formulaires, ...
- HTTP/1.0 qui procure les méthodes *POST* et *GET* pour transmettre des informations à un serveur **W3** ;
- les serveurs WWW qui permettent l'appel de programmes externes pour traiter les informations transmises par les clients (*les CGI scripts*) ;
- des outils qui facilitent l'écriture de *CGI scripts* (*cgi-lib* et *Cgi-Parse* ici).

CGI

⇒ Un exemple de formulaire : le code HTML

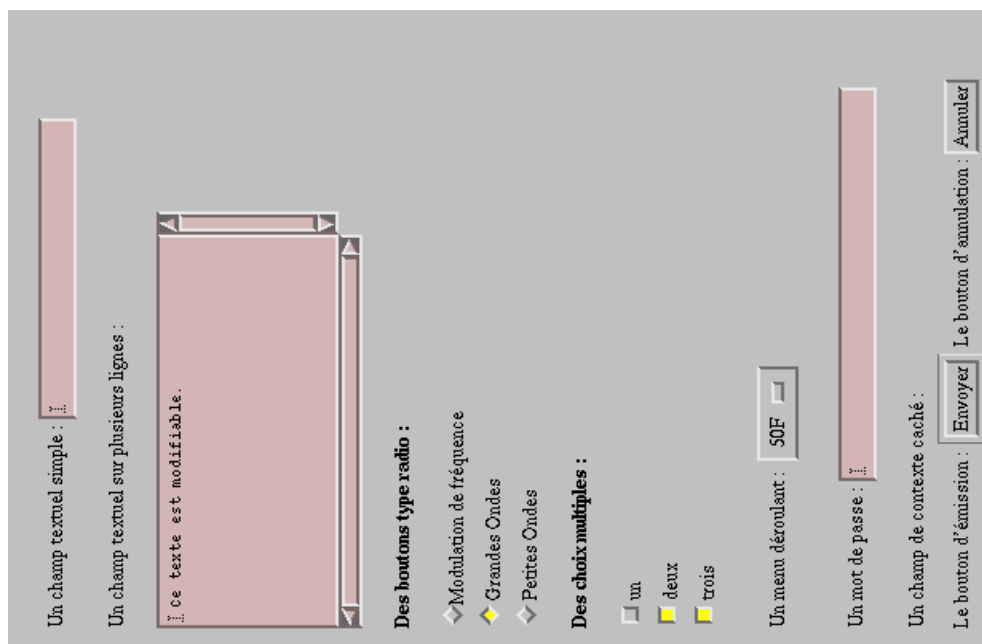
```

<HTML>
<HEAD> <TITLE> Un formulaire exemple </TITLE> </HEAD>
<BODY>
<H1>Un formulaire exemple</H1>
<FORM ACTION="http://www.univ-rennes1.fr/test-cgi-form.cgi"
METHOD="POST">
Un champ textuel simple : <INPUT NAME="simple" SIZE=30> <P>
Un champ textuel sur plusieurs lignes : <P>
<TEXTAREA NAME="lignes"
ROWS=10 COLS=40> Ce texte est modifiable.
</TEXTAREA> <P>
<STRONG>Des boutons type radio : </STRONG> <P>
<INPUT TYPE="radio" NAME="radio"
VALUE="FM" >Modulation de fréquence <BR>
<INPUT TYPE="radio" NAME="radio"
VALUE="GO" CHECKED> Grandes Ondes <BR>
<INPUT TYPE="radio" NAME="radio"
VALUE="PO"> Petites Ondes <P>
<STRONG> Des choix multiples : </STRONG> <P>
<INPUT TYPE="checkbox" NAME="qcm" VALUE=1> un <BR>
<INPUT TYPE="checkbox" NAME="qcm" VALUE=2> deux <BR>
<INPUT TYPE="checkbox" NAME="qcm" VALUE=3> trois <P>
Un menu déroulant :
<SELECT NAME="menu">
<OPTION SELECTED>50F
<OPTION>60F
<OPTION>70F
<OPTION>100F
</SELECT> <P>
Un mot de passe : <INPUT TYPE="password"
NAME="passwd" SIZE="40"> <P>
Un champ de contexte caché :
<INPUT TYPE="hidden" NAME="cache" value="secret"> <BR>
Le bouton d'émission :
<INPUT TYPE="submit" VALUE="Envoyer">
Le bouton d'annulation :
<INPUT TYPE="reset" VALUE="Annuler">
</FORM>
</BODY>
</HTML>

```

CGI

⇒ L'exemple interprété





CGI

⇒ Les formulaires

→ La description des formulaires fait partie de HTML-2.0 niveau 2, elle est bornée par `<FORM>` et `</FORM>`.

→ L'attribut `ACTION` donne l'URL de la procédure qui va traiter les champs du formulaire (une fois saisis), par défaut c'est l'URL courant (celui qui a servi pour afficher le formulaire) qui est sollicité.

→ On utilise couramment une seule procédure pour émettre un formulaire puis réceptionner les champs saisis.

→ L'attribut `METHOD` fait référence à HTTP et indique quelle est la méthode qui sera utilisée pour transmettre les champs du formulaire.



CGI

⇒ Les Formulaires

→ La méthode `POST` est conseillée, elle transmet les champs dans le corps d'un message MIME.

→ La méthode `GET` transmet les paramètres dans l'URL, divers problèmes liés à la taille et au nombre des champs peuvent survenir.



CGI

⇒ Les champs d'un formulaire

- texte simple sur une seule ligne ;
- texte sur plusieurs lignes ;
- mot de passe ;
- bouton radio ;
- choix multiples ;
- menu déroulant ;
- bouton de commande.



CGI

⇒ La transmission par HTTP du formulaire

```
POST /test-cgi-form.cgi HTTP/1.0
Accept: text/plain
Accept: application/x-html
Accept: application/html
Accept: application/postscript
.....
Accept: */*
User-Agent: NCSA Mosaic for the X Window System/2.4
Content-type: application/x-www-form-urlencoded
Content-length: 115
simple=Bonjour+%E0+tous&lignes=Ce+texte+est+modifiable
&radio=GO&qcm=1&qcm=2&menu=50F&passwd=en+clair&cache=secret
```



CGI

⇒ Les formulaires

- Les champs sont transmis derrière l'en-tête HTTP `Content-length`, ils sont encodés.
- Le serveur HTTP passe le contrôle à une procédure CGI (référéncée derrière `POST`).
- CGI Common Gateway Interface est une interface d'écriture/exécution de programmes qui se déroulent sous le contrôle d'un serveur W3.
- Les programmes (les *Gateway CGI*) peuvent être écrits dans n'importe quel langage produisant des fichiers exécutables, ils communiquent avec le serveur à l'aide de variables d'environnement.



CGI

⇒ Les formulaires

→ Le serveur transmet au programme (*Gateway CGI*) des paramètres émis par les clients :

- dans la variable d'environnement **QUERY_STRING** si les paramètres ont été émis par la méthode **GET**;
- sur l'entrée standard si les paramètres ont été émis par la méthode **POST**;



CGI

⇒ **Le programme peut émettre différents types de documents qu'il précise au moyen d'un type MIME (Text/Html, Text/Plain, ... cf. HTTP/1.0) :**

Content-type: text/html
+ une ligne vide
.... les résultats

⇒ **Le programme peut aussi retourner seulement une référence à un autre document en émettant :**

Location: URL



CGI

⇒ **Les variables d'environnement CGI Utilisées par les serveurs W3 pour communiquer avec les programmes :**

- REQUEST_METHOD indique quel est la méthode utilisée pour émettre la requête (**GET**, **POST** ou **HEAD**) ;
- QUERY_STRING contient les paramètres émis par le client dans le cas d'une requête utilisant la méthode **GET**;
- REMOTE_HOST indique (si possible) le nom de la machine cliente ;
- REMOTE_ADDR indique l'adresse IP du client ;
- CONTENT_LENGTH précise le nombre de caractères constituant les paramètres émis par le client;
- les en-têtes HTTP : HTTP_USER_AGENT, HTTP_REFERER ...



CGI

⇒ Synoptique des échanges

- Client : demande l'acquisition d'un URL par un accès HTTP utilisant la méthode GET (la description HTML d'un formulaire) ;
- Serveur : émission de l'URL demandé ;
- Client : interprète le HTML reçu ... affiche le formulaire ;
- Client : émet (après remplissage) les champs du formulaire par une requête HTTP ciblant un URL par la méthode POST ;
- Serveur : communique les champs saisis à la procédure référencée par l'URL.
- La procédure traite les paramètres (les champs du formulaire) et envoie un acquittement au client.



CGI

<pre>#!/bin/sh echo 'Content-Type: Text/html' echo " case \$REQUEST_METHOD in GET) cat <<FINHTML <HTML > <HEAD> <TITLE>Un formulaire exemple</TITLE> </HEAD> <BODY background="/images/jclair.gif"> <H1>Un formulaire exemple</H1> <FORM ACTION="http://www.univ-rennes1.fr/test-cgi-form.cgi" METHOD="POST"> Un champ textuel simple : <INPUT NAME="simple" SIZE=30> <P> Un champ textuel sur plusieurs lignes : <P> <TEXTAREA NAME="lignes" ROWS=10 COLS=40> Ce texte est modifiable. </TEXTAREA> <P> Des boutons type radio : <P> <INPUT TYPE="radio" NAME="radio" VALUE="FM" >Modulation de fréquence
 <INPUT TYPE="radio" NAME="radio" VALUE="GO" CHECKED> Grandes Ondes
 <INPUT TYPE="radio" NAME="radio" VALUE="PO"> Petites Ondes <P> Des choix multiples : <P> <INPUT TYPE="checkbox" NAME="qcm" VALUE=1> un
 <INPUT TYPE="checkbox" NAME="qcm" VALUE=2> deux
 <INPUT TYPE="checkbox" NAME="qcm" VALUE=3> trois <P> Un menu déroulant : <SELECT NAME="menu"> <OPTION SELECTED>50F <OPTION>60F <OPTION>70F <OPTION>100F </SELECT><P></pre>	<pre> Un mot de passe : <INPUT TYPE="password" NAME="passwd" SIZE="40"> <P> Un champ de contexte caché : <INPUT TYPE="hidden" NAME="cache" value="secret">
 Le bouton d'émission : <INPUT TYPE="submit" VALUE="Envoyer"> Le bouton d'annulation : <INPUT TYPE="reset" VALUE="Annuler"> </FORM></BODY></HTML> FINHTML ;; POST) echo '<HTML>' echo '<HEAD>' echo '<TITLE>Les champs du formulaires </TITLE>' echo '</HEAD>' echo '<BODY background="/images/jclair.gif">' echo '<H1> Les champs du formulaire</H1>' eval `usr/local/cern-http/bin/gziparse -form` echo '<HR>' echo Le champ textuel simple : \$FORM_simple '
' echo Le champ textuel sur plusieurs lignes : '
' \$FORM_lignes '
' echo Les boutons de type radio : \$FORM_radio '
' echo Les choix multiples : \$FORM_qcm '
' echo Le menu : \$FORM_menu '
' echo Le mot de passe : \$FORM_passwd '
' echo Le contexte caché : \$FORM_cache '
' echo '<HR>' echo Mais également les en-têtes HTTP : '
' echo User-Agent : \$HTTP_USER_AGENT '<P>' echo et aussi divers éléments de contexte : '
' echo Le client :\$REMOTE_HOST '
' echo La taille des parametres :\$CONTENT_LENGTH echo '<HR>' echo '</BODY>' echo '</HTML>' ;; *) echo '<HTML>' echo Hum ... echo '</HTML>' ;; esac</pre>
---	---



CGI

⇒ WWW et les images réactives

→ La présentation d'images réactives (*cliquables*) par les clients WWW facilite la réalisation d'interfaces graphiques conviviales :

- l'image est découpée en régions (sur un serveur HTTP), à chacune d'entre elles est associée une action (un URL) ;
- l'image est présentée au client avec l'attribut `ISMAP` (par l'intermédiaire d'une ancre) ;
- le client transmet au serveur HTTP les coordonnées de l'endroit *cliqué*.



CGI

⇒ WWW et les images réactives (suite)

```
<A HREF="/cgi-bin/imagemap/France">  
  <IMG SRC="/France/France.gif" ISMAP>  
</A>
```

⇒ Quand l'utilisateur «clique» dans une zone de la carte présentée, le client émet la requête suivante :

```
GET /cgi-bin/imagemap/France?323,425 HTTP/1.0
```

CGI

⇒ WWW et les images réactives (suite)

→ Les interfaces utilisateurs utilisant des images réactives ne sont pas toujours bien comprises des usagers.

→ Le client WWW n'a pas connaissance du découpage en régions, il ne peut donc pas signaler (par un effet vidéo) les zones sensibles de l'image.

→ Exemples:

`http://web.urec.fr:/France/France.html`
`http://www.univ-rennes1.fr/ROR/renater.html`

Harvest





Harvest

⇒ **Harvest est un ensemble d'outils permettant :**

- la collecte de documents diffusés par des services d'informations ;
- la diffusion des bases de données construites par un processus collecteur ;
- de répondre à des recherches dans l'espace de documents sondés ;
- de gérer un cache de documents.



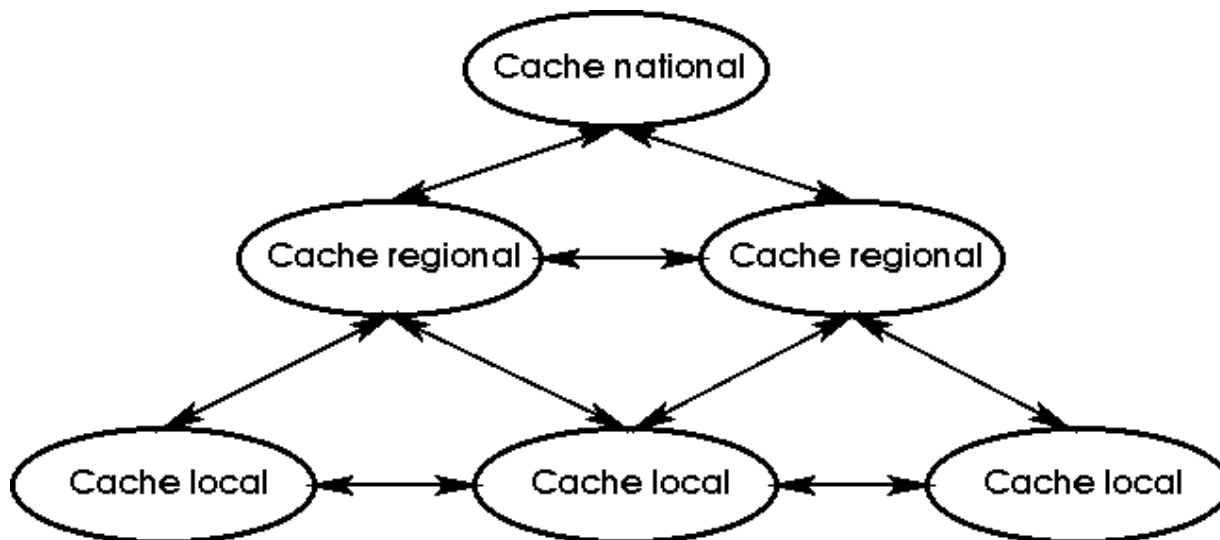
Harvest cache :

⇒ **Le cache d'Harvest est plus efficace que le cache du CERN httpd :**

- Harvest gère un cache à deux niveaux :
- en RAM avec migration par *water marks*. Un bon rendement nécessite de le tailler largement ;
- sur disque ;
- non *forky*, réalisé à partir d'accès réseaux-I/O non bloquants ;
- met en oeuvre un cache DNS ainsi que des *nslookups* non bloquants.

Harvest cache :

⇒ Un réseau de cache peut être mis en oeuvre de la façon suivante :



Harvest cache :

⇒ remarques

- la protection d'accès au cache s'effectue en manipulant des numéros IP;
- la durée de vie des documents s'effectue en positionnant un *Time to Live* (TTL) :

Tableau 3 :

	Regular Expr. matching URLs	Absolute TTL in minutes	Calculate TTL as this percentage of the object's age	
			% age	Max (min)
ttl_pattern	^http://	10080	50%	43200
ttl_pattern	^ftp://	10080	50%	43200
ttl_pattern	\.gif\$	10080	50%	43200
ttl_pattern	/cgi-bin/	0	0%	43200

Harvest cache :



⇒ la version 1.3 de *cached* ne gère pas l'en-tête HTTP *If-Modified-Since* ;



⇒ le contenu du cache est accessible au moyen d'une procédure utilisant des formulaires.

Conclusion

⇒ Le web est un outil très puissant de diffusion de l'information

Mais :

- encore faut-il le maîtriser => laissons la place aux professionnels de la communication
- il est l'image de l'organisme à l'extérieur
- l'Internet n'est pas un monde en dehors des lois



Conclusion

⇒ Perspectives

→ évolution des standards :

- HTML version 3, 4, ... (un peu de patience)

→ nouveaux standards :

- Java¹
- VRML²
- ...

→ nouvelles applications

1. <http://www.javasoft.com>
2. <http://webspac.sgi.com>