CR 5 HTTP

Noémie PRIN et Maria-Bianca ZUGRAVU

mdp: TPweb2018

Remarques pratiques

Connexion parallele : les navigateurs peuvent envoyer plusieurs requêtes simultanément sur plusieurs connexions TCP. Un serveur est limité en nombre total de connexions qu'il peut traiter, il y a un risque de le saturer.

Connexion persistante: permettent au serveur de ne pas fermer la connexion TCP dès que la requête est terminée. Le client pourra alors réutiliser la connexion sans devoir la re-négocier. HTTP 1.0 ne supporte pas les connexions persistantes nativement. Sauf avis contraire, en HTTP 1.1, toute connexion est par défaut persistante (cette caractéristique à été longuement débattue à l'époque). Au contraire avec HTTP 1.0, sauf si le serveur répond avec un en-tête Connection: keep-alive et que le client a expréssement demandé une connexion persistante avec le même en-tête dans sa requête : la connexion est fermée par défaut.

Comment donner les droit d'acces?

- donner accès en exécution à leur répertoire personnel **chmod g+x~**
- donner accès en exécution, lecture et écriture à leur répertoire personnel : **dhmod g+x/~public_html**
- Pour les scripts CGI, une moulinette créé les liens symboliques depuis /var/www/cgi-bin/LOGIN vers /home/pnom/cgi-bin. Les scripts CGI seront alors accessibles via l'URL racine http://asi-technoweb. insa-rouen.fr/cgi-bin/LOGIN/

ou en utilisant FileZilla

Modifier des fichiers directement sur le terminal : utilisation de **vim**

- **vim nom_fichier**
- i: pour inserer du texte et on se place où on veut faire l'insertion
- **esc** et **:wp** pour sauvegarder
- pour tout quitter: **:qa!**

Si on veut modifier les fichiers directement dans le repertoire local :

On clique sur **autres emplacements**. Puis dans le champ **Connexion à un serveur**: sftp://192.168.56.101/ et on se connecte (identifiant: **root** mdp: **TWweb2018**). Puis le répertoire distant sera disponible et on retrouvera les fichiers .php qui nous intéressent dans *2REMOVE*

Scripts

1. Serveur Web Apache HTTP

1 Tester le script CGI

On test le fichier **hello-perl.cgi** directement dans le navigateur: http://asi-technoweb.insa-rouen.fr/cgi-bin/hello-perl.cgi

2. Test des scripts depuis le navigateur

le lancement du script de création est fait automatiquement juste en se connectant en mode ssh: création des répertoires cgi-bin et public-html

Connexion au serveur : ssh asi-technoweb.insa-rouen.fr

Copier des fichiers dans /public-html :

- on se deconnecte de ssh et on se place dans le repertoire depuis lequel on veut copier les fichiers à déposer sur le serveur –> commande **scp** : * pour selectionner tous les fichiers

scp * nprin@asi-technoweb.insa-rouen.fr:~/public_html/

Pour ouvrir un fichier depuis serveur et dans navigateur : http://asi-technoweb.insa-rouen.fr/~login/<fichier_a_ouvrir.html>

Exercice 3 : Modification de la page d'inscription pour que le traitement du formulaire soit effectué en GET par le CGI

Fichier modifié: TP1/inscription.html

- **externe** :
- **en interne** :

Script forum_ASI.php qui se trouve dans public-public_html

Exercice 3 : Modification de la page d'inscription pour que le traitement du formulaire soit effectué en POST par le CGI

Pareil :

```
<form action="http://asi-technoweb.insa-rouen.fr/cgi-bin/inscription-get.cgi" action="POST">
```

Différence entre GET et POST

Au niveau de lien sur le navigateur: pour **GET** on obtient un lien qui contient les donnees transmises par le formulaire, separees par des caracteres speciaux representant les espaces entre les differents elements.

Différence entre GET interne et externe

- **GET EXTERNE**:

```
<form name="formulaire"
action="http://asi-technoweb.insa-rouen.fr/cgi-bin/inscription-get.cgi">
```

- **GET INTERNE**:

```
<form name="formulaire" action="/cgi-bin/inscription-get.cgi" method="get">
```

- **POST EXTERNE**:

```
<form name="formulaire"
action="http://asi-technoweb.insa-rouen.fr/cgi-bin/inscription-post.cgi"
method="post" enctype="application/x-www-form-urlencoded">
```

- **POST INTERNE**:

```
<form name="formulaire" action="/cgi-bin/inscription-post.cgi" method="post"
enctype="application/x-www-form-urlencoded">
```

- **POST HOME**:

```
<form name="formulaire" action="/cgi-bin/apauchet/inscription-post.cgi" method="post"
enctype="application/x-www-form-urlencoded">
```

2. Protocole HTTP

Test des requêtes aupres de la VirtualBox

Connexion : *>>> ssh root@192.168.56.101*

mdp: *TPweb2018*

Verifier les scripts php : directement dans le navigateur en tapant: 192.168.56.101 (l'ip trouvé dans la VirtualBox)

1. Requête pour la date de dernière modification d'une page

- **si page modifiée depuis une certaine date**:

GET /~apauchet/index.html HTTP/1.1

Host: asi-technoweb.insa-rouen.fr

If-Modified-Since: 'Sun, 09 Sep 2007 17:00:00 GMT'

- **si la page n'a pas été modifiée depuis une certaine date**:

GET /~apauchet//index.html HTTP/1.1

Host: asi-technoweb.insa-rouen.fr

If-Unmodified-Since: 'Sun, 09 Sep 2007 17:00:00 GMT'

2. Requête récupérant que les caractères 2 à 5 d'une page web

Le premier caractere est a 0!

GET /~apauchet/index.html HTTP/1.1

Host: asi-technoweb.insa-rouen.fr

Range: bytes=1-4

3. Requête effectuant une négociation sur le type de média demandé

Les 2 fichiers coexistent, image.png et image.jpg.

- **negociation pour privilégier les images au format jpeg par rapport au format png**

HEAD /~apauchet/image HTTP/1.1

Host: asi-technoweb.insa-rouen.fr

Accept: image/jpeg; q=1, image/png; q=0.5

- **l'inverse**

HEAD /~apauchet/image HTTP/1.1

Host: asi-technoweb.insa-rouen.fr

Accept: image/jpeg; q=0.5, image/png; q=1

4. Requêteavec négociation de contenu sur la langue

GET /~apauchet/index.html HTTP/1.1

Host: asi-technoweb.insa-rouen.fr

Accept-Language: en;q=0.5, fr;q=1

5. Requête HTTP/1.1 sans connexion persistante

Par défaut pour un telnet, la connexion n'est pas persistante avec un GET; il faut donc tester avec un HEAD. Pour netcat, la connexion est persistante avec un GET.

HEAD /~apauchet/index HTTP/1.1

Host: asi-technoweb.insa-rouen.fr

Connection: close

6. 2 requêtes retournant les codes suivants : 404, 501.

- **404**: *page inconnue*

HEAD /~apauchet/Fichier-N-Existant.pas HTTP/1.1

Host: asi-technoweb.insa-rouen.fr

- **501**: *commande inconnue*

COMMANDE-INCONNUE /~apauchet/index HTTP/1.1

Host: asi-technoweb.insa-rouen.fr

Points importants CM

Principes de fonctionnement:

- le **maître** (utilisateur *root* qui écoute le port standard) reçoit la connexion
- le **maître** crée un **esclave** et lui transmet le canal de communication
- l'esclave traite la requête et retourne le résultat

~ : indique que le fichier est personnel

Exemple requête

requête

```
 > netcat localhost 80
GET /phrase.txt HTTP/1.1
Host: localhost
```

entête réponse

```
HTTP/1.1 200 OK
Date: Wed, 15 Jul 2009 13:08:49 GMT
Server: Apache/2.2.11 (Ubuntu) PHP/5.2.6-3ubuntu4.1 with Suhosin-Patch
Last-Modified: Tue, 14 Jul 2009 18:24:33 GMT
ETag: "31c06d-1c-46eae8cd55a40"
```

```
Accept-Ranges: bytes
Content-Length: 28
Content-Type: text/plain
```

corps réponse

```
Voici un exemple de phrase.
```

Requête

Request-Line

```
METHODE URI [HTTP-Version]
```

Les méthodes:

- *OPTIONS*: demande les méthodes utilisables sur l'UR
- *GET*: demande les informations et les données de l'URI
- *POST*: envoie de données (ex : formulaire) traitées par l'URI
- *HEAD*: demande uniquement les informations sur l'URI
- *PUT* : enregistre le corps de la requête à l'URI
- *DELETE* : supprime les données pointées par l'URI
- *TRACE* : retourne ce qui a été envoyé par le client (comme un echo)

La méthode **GET** *est surtout celle utilisée pour charger les images et les pages HTML. Il existe d'autres méthodes, appelées aussi verbes comme* **POST** *qui est utilisée notamment pour envoyer les données d'un formulaire sur un serveur.*

Les autres params pour une requête

- *Cache-Control* : définit la politique de cache pour la ressource
- *Date* : date du message
- *Pragma* : utilisé pour spécifier des comportements aux serveurs intermédiaires (proxy)
- *Transfer-Encoding* : types de transformations appliquées au corps du message
- *Via* : indique les intermédiaires par lesquels est passée la requête
- *Connection* : paramètre de gestion de la connexion (ex: *Connection: close*)
- *Upgrade* : spécifie quels autres protocoles supporte le client
- *Accept* : types de médias acceptés (ex : Accept: text/html)
- *Accept-Charset* : spécifie les jeux de caractères acceptés
- *Accept-Encoding* : spécifie les types de transformations (compressions) du message acceptés
- *Accept-Language* : spécifie les langues acceptées
- *From* : e-mail de l'utilisateur du client (nécessite accord)
- *Host* : spécifie le serveur (et le port) pour la requête
- *If-Modified-Since*, *If-Unmodified-Since* : requête conditionnelle sur la dernière date de modification de l'URI
- *Range* : précise la portion de données de la ressource

- *Referer* : spécifie l'URI à l'origine de la requête
- *User-Agent* : contient l'identifiant du navigateur client
- *Allow* : liste les méthodes autorisées
- *Content-Encoding* : indique l'encodage utilisé pour la ressource (complément au type de média du Content-Type)
- *Content-Language* : défini la langue utilisée
- *Content-length* : taille du corps du message
- *Content-Location* : donne la véritable URI de la ressource si celle-ci a été trouvée grâce à une autre URI
- *Content-Range* : donne la plage de données récupérées sur la totalité de la ressource
- *Content-Type* : le type du média (ex : text/html; charset=ISO-8859-1)
- *Expires* : date d'expiration de la ressource
- *Last-Modified* : date de dernière modification

Status-Line

```
HTTP-Version Status-Code Reason-Phrase
```

- *Status-Code* : code numérique représentant le succès où l'échec de la requête
  - **1XX** : Information
  - **2XX** : Succès
  - **3XX** : Redirection
  - **4XX** : Erreur client
  - **5XX** : Erreur serveur
- *Reason-Phrase* : texte expliquant le Status-Code

Reponse

- *Accept-Ranges* : informe l'acceptation des requêtes Range par le serveur
- *Location* : redirige la requête vers une autre URI (ex: *Status-Code: 3XX*)
- *Server* : indique le type du serveur web répondant à la requête

Exemple

```
GET / HTTP/1.1
Host: www.lemonde.fr
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/54.0.2840.34 Safari/537.36
```

On peut faire des négociations sur:

- Langue

  GET /info.txt HTTP/1.1 Host: localhost Accept-Language: fr;q=1,en;q=0.5

  HTTP/1.1 200 OK Date: Wed, 15 Jul 2009 13:50:02 GMT Server: Apache/2.2.11 (Ubuntu) PHP/5.2.6-3ubuntu4.1 with Suhosin-Patch

Content-Location: info.txt.fr Vary: negotiate,accept-language TCN: choice Last-Modified: Tue, 14 Jul 2009 18:24:33 GMT ETag: “31c070-15-46eae8cd55a40;46ebeabae2180” Accept-Ranges: bytes Content-Length: 21 Content-Type: text/plain Content-Language: fr

Ceci est du francais.

- Type MIME: text/plain, text/html, image/jpeg, image/png, audio/mpeg, audio/ogg„ video/mp4, application/octet-stream
- Charset (encodage des caractères)
- Encodage (compression, encodage, etc.)