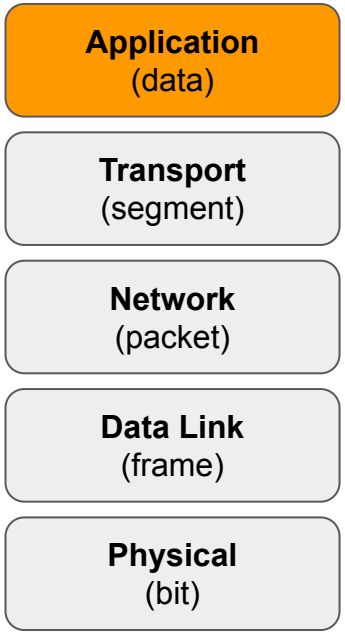
Cours 5

Couche Application & Socket

Introduction

La **Couche Application** du modèle Internet se divise en 3 couches dans le **Modèle OSI** :

- **Couche Session** : gestion d'une session qui persiste au-delà d'une connexion, mécanisme d'ouverture et de fermeture de session, identifier un utilisateur, authentification, …
- **Couche Présentation** : encodage des données applicatives (conversion des données au format "machine" dans un format "échangeable"), compression, chiffrement / déchiffrement, …
- **Couche Application** : point d'accès au service réseau ; non spécifiée dans le modèle OSI.



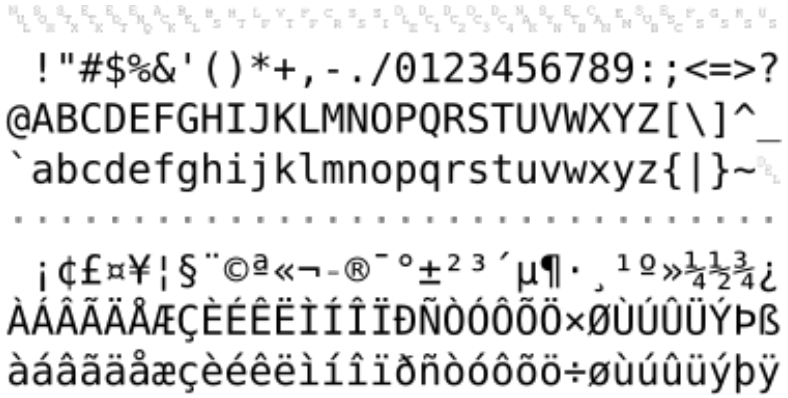
Exemples

- FTP, NFS, SMTP, POP, IMAP, NNTP, Telnet, SSH, X, HTTP, DNS, …

Codage de Caractères

Les Standards

- **ASCII** (en 1963) : codage des caractères anglais sur 7 bits…
- **Latin-1** (ISO 8859-1, en 1987) : extension de l'ASCII sur 8 bits, ajout des caractères latins manquants (accents, …), mais certains caractères sont manquants comme € !
- **UTF-8** (RFC 3629, en 1996) : extension de l'ASCII, implémentation du standard **Unicode** avec un répertoire de 150 000 caractères (*code point*), couvrant plus de 150 écritures (codage de taille variable entre 1 et 4 octets). Devenu le standard *de facto* avec 95% des sites web qui l'utilisent en 2020…



Caractères ISO Latin-1.

Problèmes de Codage !

Trois fichiers textes dans trois formats…

```
$ file test-*.txt
test-ascii.txt:  ASCII text              [8 octets]
test-latin1.txt: ISO-8859 text               [8 octets]
test-utf8.txt:   UTF-8 Unicode text      [9 octets]
```

Affichage dans un terminal UTF-8

```
$ cat test-ascii.txt
aurelien
$ cat test-latin1.txt
aur?lien
$ cat test-utf8.txt
aurélien
```

```
$ hexdump -C test-ascii.txt
61 75 72 65 6c 69 65 6e
$ hexdump -C test-latin1.txt
     61 75 72 e9 6c 69 65 6e
$ hexdump -C test-utf8.txt
61 75 72 c3 a9 6c 69 65 6e
```

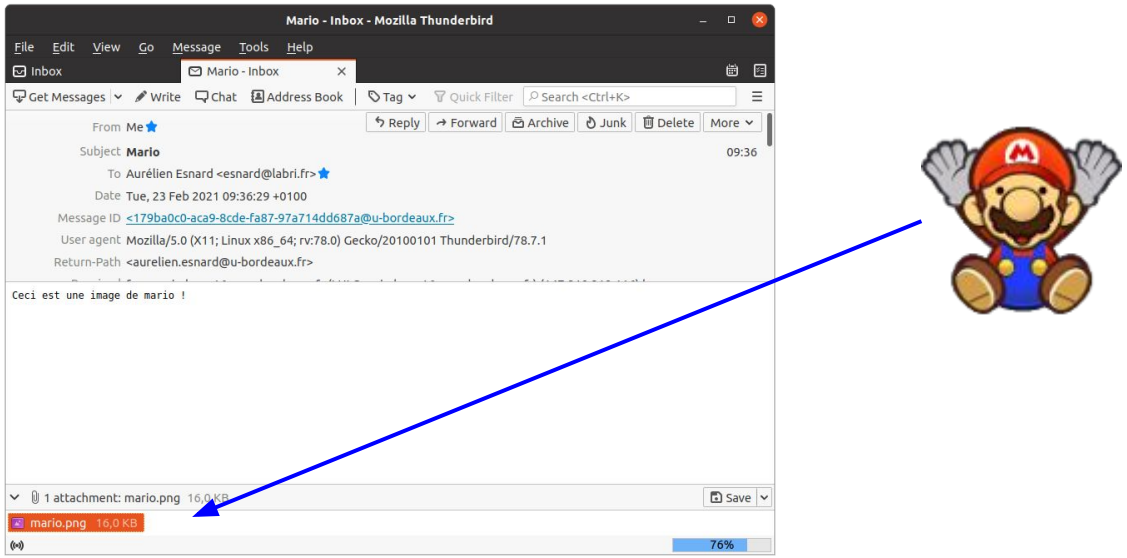
Utilisation du standard Unicode (ex. Emoji)

```
$ echo -e "\x62\x69\x65\x72"   ⇒   bier     [ASCII text (hex)]
$ echo -e "\U1F37A"            ⇒   🍺        [Unicode Code Point]
$ echo -e "\xf0\x9f\x8d\xba"   ⇒   🍺        [Unicode Character (hex)]
```

Encodage de Données Multimedia

MIME (Multipurpose Internet Mail Extensions)

- Envoi de mail via SMTP uniquement en ASCII à l'origine…
- Extension MIME nécessaire pour envoyer des mails avec d'autres jeux de caractères et pour envoyer des données binaires diverses (multimedia, …)
- MIME également utilisé avec HTTP
- Utilisation du format *Base64* pour convertir le binaire en ASCII
- Exemple d'un mail avec une image en pièce-jointe…



Exemple d’un Mail

```
Return-Path: <aurelien.esnard@u-bordeaux.fr>
Received: from v-zimboxp16.srv.u-bordeaux.fr
Received: from mta-in01.u-bordeaux.fr
Received: from v-zimmta03.u-bordeaux.fr
Received: from [192.168.0.106]
To: <esnard@labri.fr>
From: <aurelien.esnard@u-bordeaux.fr>
Subject: Mario
Message-ID: <179ba0c0-aca9-8cde-fa87-97a714dd687a@u-bordeaux.fr>
Date: Tue, 23 Feb 2021 09:36:29 +0100
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Thunderbird/78.7.1
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="--frontier--"
Content-Language: en-US
This is a multi-part message in MIME format.
--frontier--
Content-Type: text/plain; charset=utf-8; format=flowed
Content-Transfer-Encoding: 7bit
Ceci est une image de mario !
--frontier--
Content-Type: image/png; name="mario.png"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="mario.png"
```


```
iVBORw0KGgoAAAANSUhEUgAAAGQAAABkCAYAAABw4pVUAAAABmJLR0QA/wD/AP+gvaeTAAAA
CXBIWXMAAA7DAAAOwwHHb6hkAAAAB3RJTUUH4QIIChsnrxPiLwAAIABJREFUeNrsvXeYHUeV
//051d03zp2co2aU84xkWR4H2ZYDNsbGkWgyywZYwi6wsLzL/oAFFliivT/CwtrAEm2DsdfZ
luUgj4KlGUUrjUbSpDsjTZ4bu7vq/ePeUbAswIm2DsdfZ
luUgj4KlGUUrjUbSpDsjTZ4bu7vq/ePeUbAswIsJ+77Uo3p6nr73dledb518qgR/bn9uf25/
...
--frontier--
```

Base64

Conversion de données binaires en texte (ASCII)

Représentation de 6 bits avec 64 caractères ASCII (A-Z,a-z,0-9,+,/)
Un paquet de 3 octets est repésenté par 4 caractères (3x8 bits = 4x6 bits)

```
$ echo -ne "\x00\x00\x00\xFF\xFF\xFF" | base64
AAAA////
```

Exemple d'une image 100x100 en PNG

```
$ base64 mario.png > mario.b64          # encodage
$ base64 -d mario.b64 < mario.png       # decodage
```

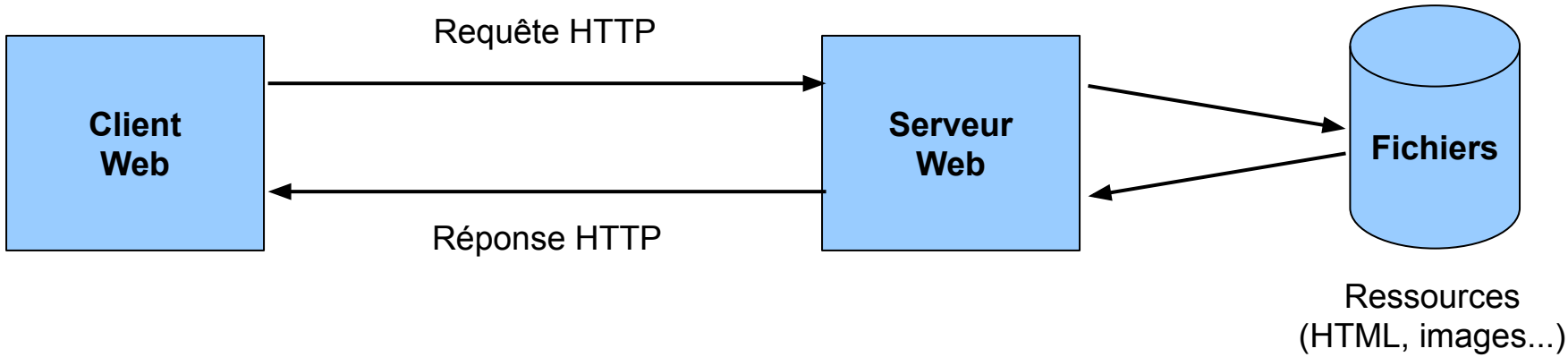
```
$ cat mario.b64
```


```
iVBORw0KGgoAAAANSUhEUgAAAGQAAABkCAYAAABw4pVUAAAABmJLR0QA/wD/AP+gvaeTAAAA
CXBIWXMAAA7DAAAOwwHHb6hkAAAAB3RJTUUH4QIIChsnrxPiLwAAIABJREFUeNrsvXeYHUeV
//051d03zp2co2aU84xkWR4H2ZYDNsbGkWgyywZYwi6wsLzL/oAFFliivT/CwtrAEm2DsdfZ
luUgj4KlGUUrjUbSpDsjTZ4bu7vevq/eHeUbAswIsJ+77Uo3p6nr73dledb518qgR/bn9uf25/
...
```

Protocole HTTP

HTTP (Hypertext Transfer Protocol)

- Protocole *stateless* basé sur TCP/IP inventé en 1990 (RFC 2616)
- Le serveur est à l'écoute sur le port 80 (ex. Apache, Nginx, …)
- Le client est en général un navigateur (ex. Chrome, Firefox, Edge, …)
- Le navigateur effectue une requête HTTP pour obtenir une ressource à partir d'une URI (Uniform Ressource Identifier)
- Le serveur traite la requête puis retourne une réponse HTTP, typiquement une page HTML…
- HTTPS : version sécurisée de HTTP (port 443)



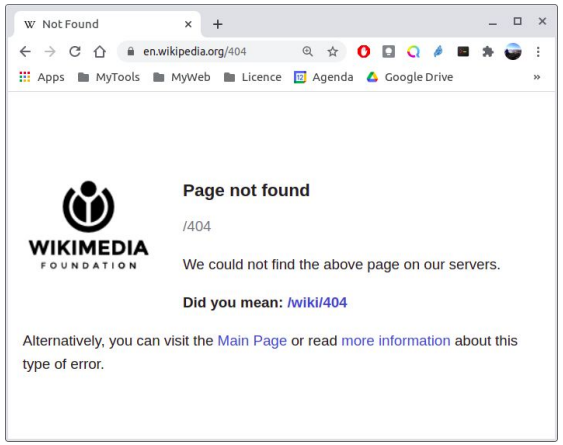
Protocole HTTP

Les principales requêtes

- **GET** : demander une ressource (ex. pages web, scripts, images, …)
- **POST** : envoyer des données au serveur (ex. message forum, formulaire)
- **Divers** : HEAD, TRACE, CONNECT, PUT, DELETE, ...

Codes d'état

- **200** : succès de la requête
- **301** : redirection permanente
- **404** : page non trouvée (erreur)

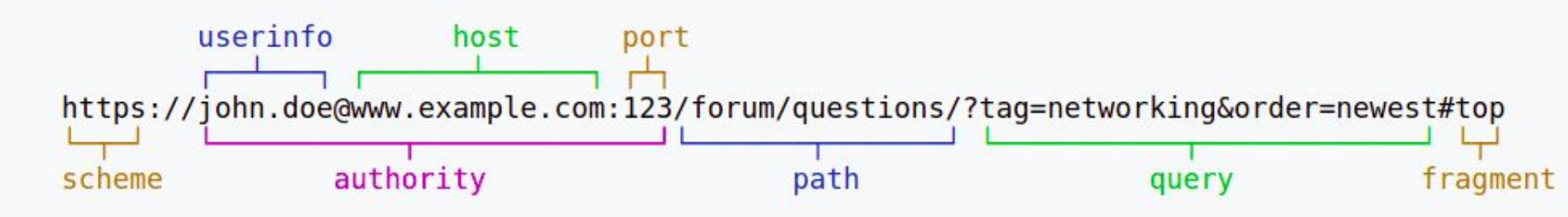


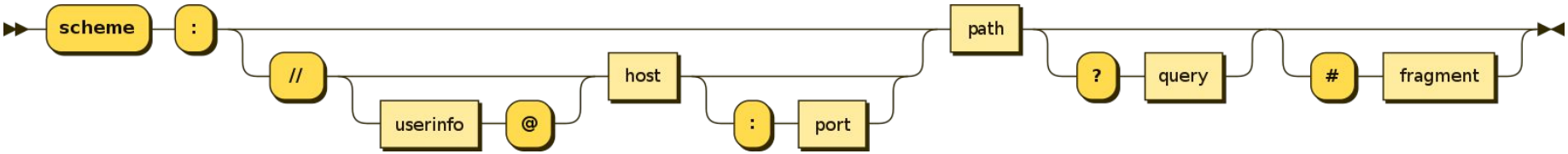
Historique

- **Version 0.9** : requête GET, réponse HTML
- **Version 1.0** : gestion de cache, type MIME (content-type), ...
- **Version 1.1** : connexion persistante (keep-alive), négociation de contenu (accept-*), ...

URI (Uniform Ressource Identifier)

URI = scheme:[//authority]path[?query][#fragment]





Un peu de HTML…

HTML (Hypertext Markup Language) : version 5 en 2014, W3C.

- Langage à balise ouvrante & fermante : <html> … </html>
- En-tête avec des metadonnées : <title>, <meta>, …
- Structuration hiérarchique : <body>, <h1>, <h2>, … <p>, …
- De la mise en forme : , <b>, <pre>, …
- Mais encore : des liens hypertextes <a>, des images <img>, des scripts <script>, des formulaires <form>, …

Exemple

```
<html>
   <head>
      <title>Hello World!</title>
   </head>
   <!-- un commentaire -->
   <body>
      <h1>Hello World!</h1>
      <h2>Subtitle</h2>
      <p>Ceci est un paragraphe.</p>
   </body>
</html>
```

Capture d'une Trace HTTP

Requête GET (en rouge) vers le site www.perdu.com et réponse en bleu…

```
GET / HTTP/1.1
User-Agent: Wget/1.20.1 (linux-gnu)
Accept: */*
Accept-Encoding: identity
Host: perdu.com
Connection: Close

HTTP/1.1 200 OK
Date: Wed, 19 Feb 2020 18:44:39 GMT
Server: Apache
Upgrade: h2
Connection: Upgrade, close
Last-Modified: Thu, 02 Jun 2016 06:01:08 GMT
ETag: "cc-5344555136fe9"
Accept-Ranges: bytes
Content-Length: 204
Vary: Accept-Encoding
Content-Type: text/html

<html><head><title>Vous Etes Perdu ?</title></head><body><h1>Perdu sur l'Internet ?
</h1><h2>Pas de panique, on va vous aider</h2><strong><pre>    * <----- vous
&ecirc;tes ici</pre></strong></body></html>
```

Capture Wireshark : http://aurelien-esnard.emi.u-bordeaux.fr/trace/http.pcap

Outils

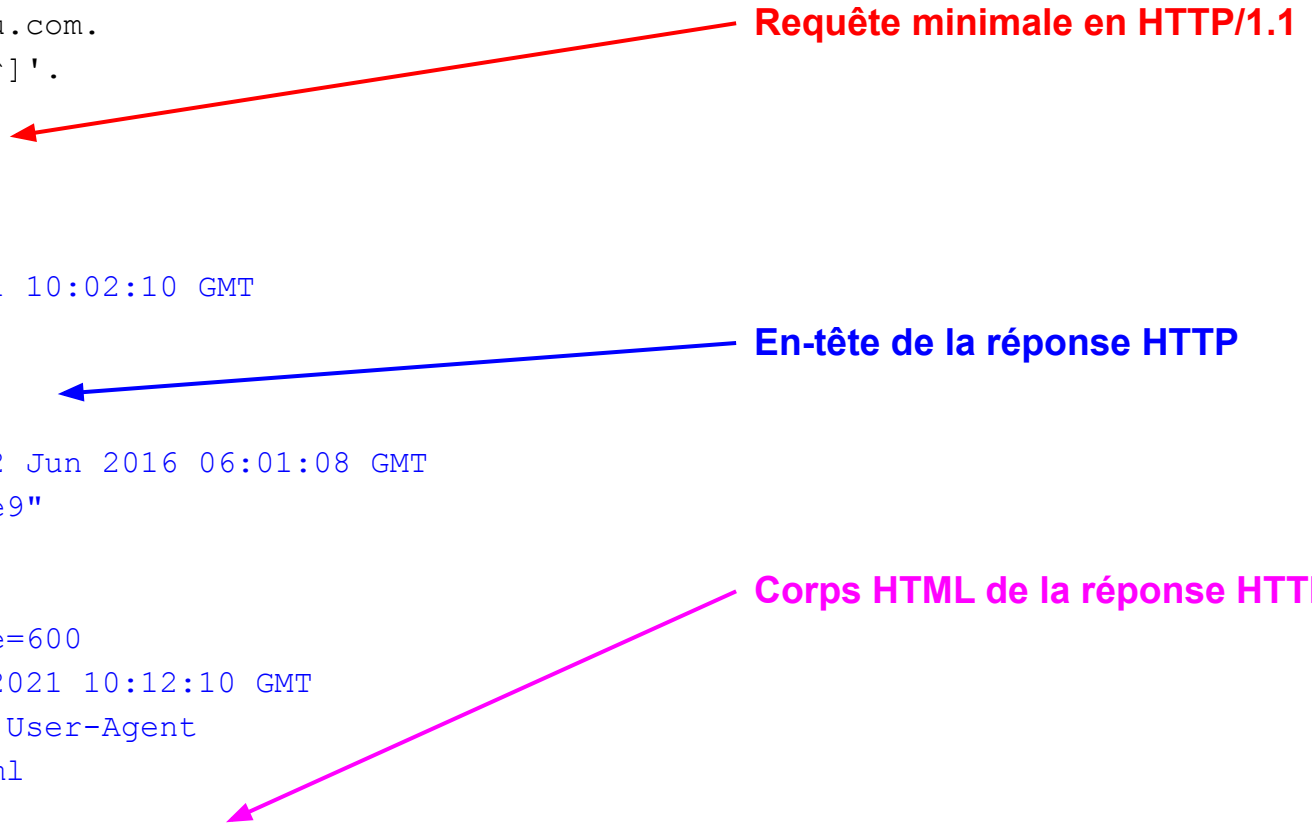
```
$ wget http://www.perdu.com   # ou curl
Resolving www.perdu.com...
Connecting to 208.97.177.124:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 204 [text/html]
Saving to: index.html
```

```
$ cat index.html
<html><head><title>Vous Etes Perdu ?</title></head><body><h1>Perdu sur l'Internet
?</h1><h2>Pas de panique, on va vous aider</h2><strong><pre>    * <----- vous &ecirc;tes
ici</pre></strong></body></html>
```

```
$ tidy index.html
<html>
<head>
<title>Vous Etes Perdu ?</title>
</head>
<body>
<h1>Perdu sur l'Internet ?</h1>
<h2>Pas de panique, on va vous aider</h2>
<pre><strong>    * <----- vous êtes ici</strong></pre>
</body>
</html>
```

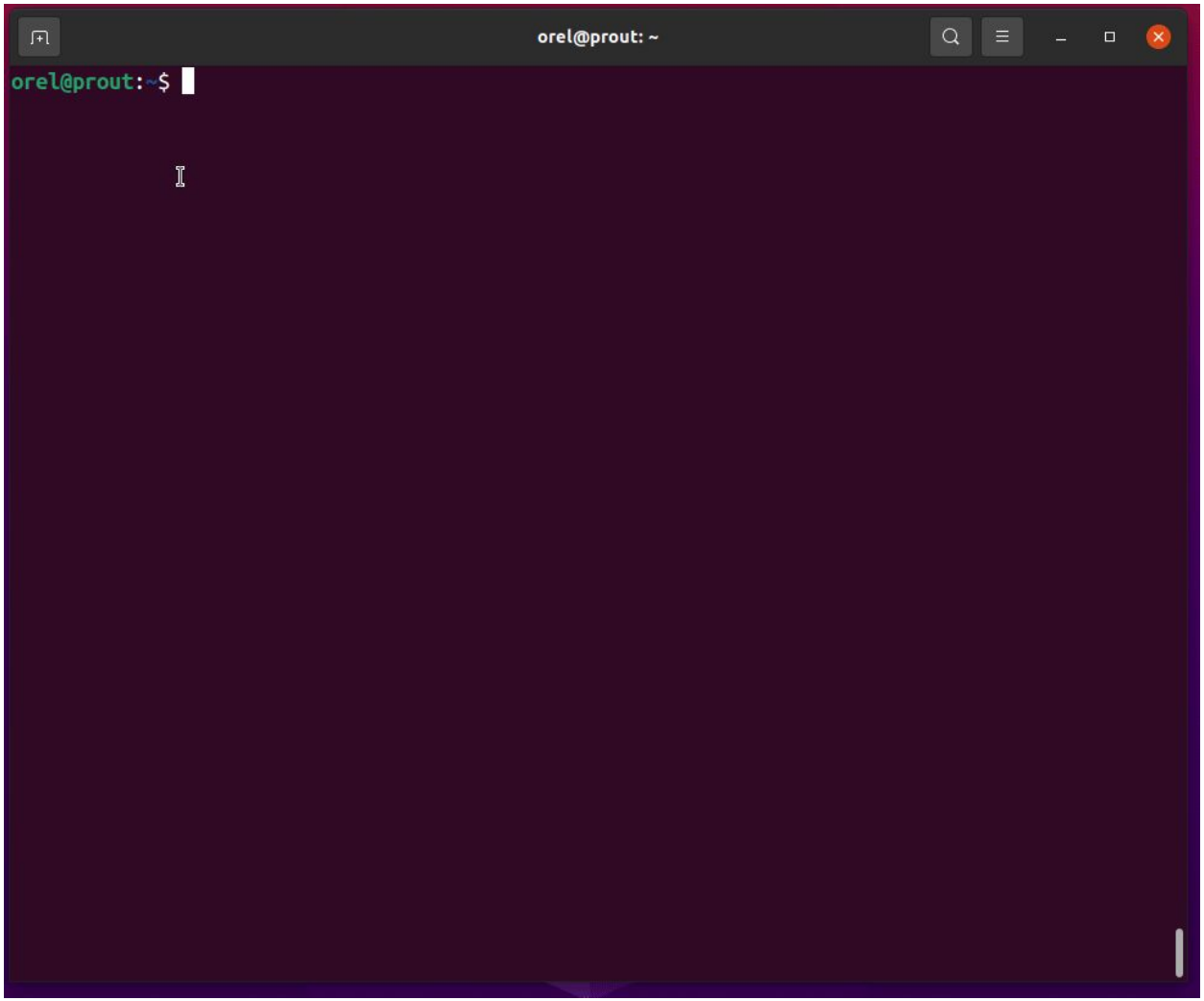
Requête à la Main avec Telnet

```
$ telnet www.perdu.com 80
Trying 208.97.177.124…
Connected to www.perdu.com.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.perdu.com

HTTP/1.1 200 OK
Date: Tue, 23 Feb 2021 10:02:10 GMT
Server: Apache
Upgrade: h2
Connection: Upgrade
Last-Modified: Thu, 02 Jun 2016 06:01:08 GMT
ETag: "cc-5344555136fe9"
Accept-Ranges: bytes
Content-Length: 204
Cache-Control: max-age=600
Expires: Tue, 23 Feb 2021 10:12:10 GMT
Vary: Accept-Encoding,User-Agent
Content-Type: text/html

<html><head><title>Vous Etes Perdu ?</title></head><body><h1>Perdu sur l'Internet
?</h1><h2>Pas de panique, on va vous aider</h2><strong><pre>    * <----- vous &ecirc;tes
ici</pre></strong></body></html>
Connection closed by foreign host.
```



Requête à la Main avec Telnet

Démo



Socket

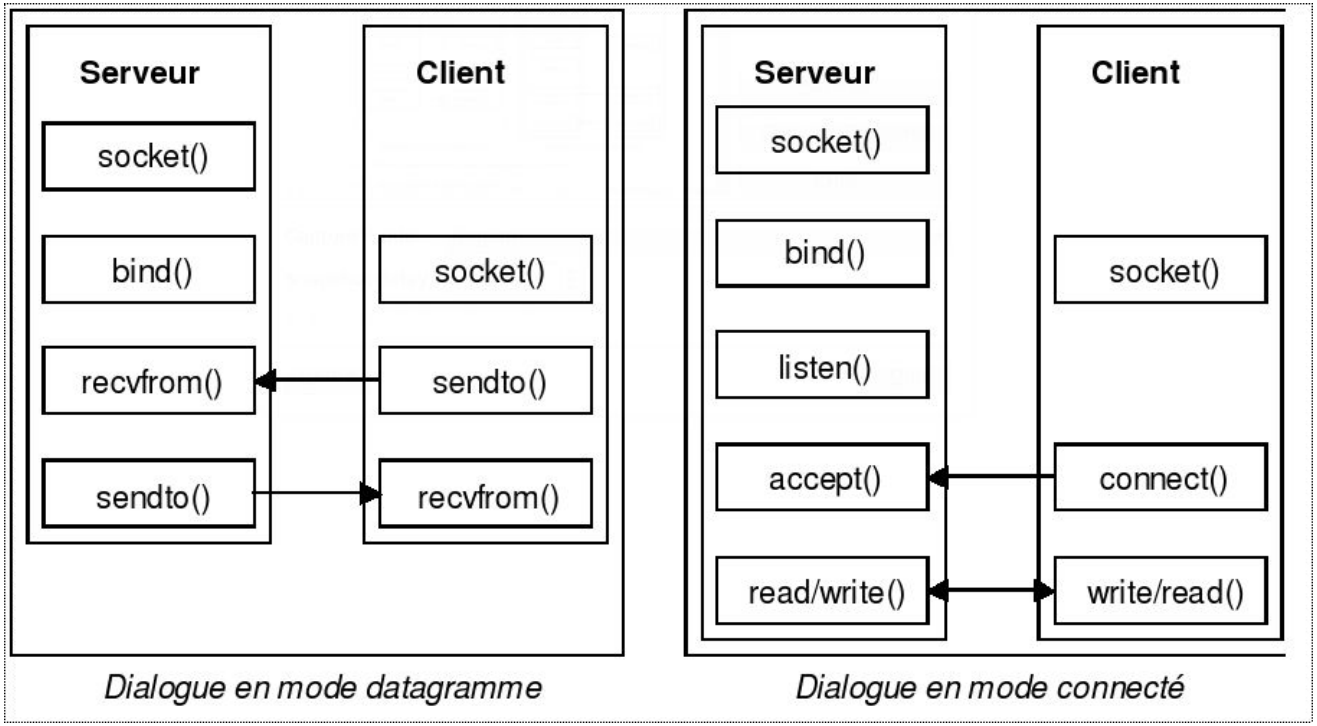
Comment programmer des applications réseaux au dessus de la couche Transport ?

- Création d'une *socket* avec la fonction socket()
  - IPv4 (AF_INET) ou IPv6 (AF_INET6)
  - TCP (SOCK_STREAM) ou UDP (SOCK_DGRAM)
- Connexion TCP
  - côté client : connect()
  - côté serveur : accept()
- Configuration d'un serveur : listen() / bind()
- Envoi et réception de données :
  - send() / sendall() / recv() en TCP
  - sendto() / recvfrom() en UDP
- Fermeture de la socket : close()

Primitive	Meaning
SOCKET	Create a new communication end point
BIND	Attach a local address to a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Block the caller until a connection attempt arrives
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

Socket

TCP versus UDP



Documentation pour le langage Python3

- API en Python : https://docs.python.org/3/library/socket.html
- How To : https://docs.python.org/3/howto/sockets.html

Python Tips

Les fonctions de la famille send()/recv() ne manipulent pas des string classiques, mais des byte-array :

```
string = "coucou"      # string classique de type str
byterray = b"coucou"   # byte array (notez le prefixe b)
sock.send(bytearray)
```

Pour convertir une string en byte-array (et inversement), vous pouvez utiliser les fonctions suivantes :

```
string = "coucou"                 # type str
bytearray = data.encode("utf-8")
bytearray = b"coucou"             # type bytearray
string = bytearry.decode("utf-8")
```

Support de nombreux encodages

```
"é".encode("latin-1")
b'\xe9'
"é".encode("utf-8")
b'\xc3\xa9'
```

Socket Python

Client Daytime (UDP)

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.sendto(b'', ('time-c.nist.gov',13))
data = s.recvfrom(1024)
print(data)
s.close()
```

Client Daytime (TCP)

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('time-c.nist.gov',13))
data = s.recv(1024)
print(data)
s.close()
```

Socket Python

Client HTTP, requête GET (TCP)

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('www.perdu.com', 80))
s.sendall(b'GET / HTTP/1.1\r\nHost: www.perdu.com\r\nConnection:
close\r\n\r\n')
data = s.recv(1024)
s.close()
print (data)
```

Socket Python

Exemple d'un Serveur Echo (TCP) : un seul client à la fois…

```
import socket
HOST = ''
PORT = 7777
sserver = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sserver.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
sserver.bind((HOST, PORT))
sserver.listen(1)
while True:
    sclient, addr = sserver.accept()
    print('Connected by', addr)
    while True:
        data = sclient.recv(1500)
        if data == b'' or data == b'\n' : break
        print(data)
        sclient.sendall(data)
    print('Disconnected by', addr)
    sclient.close()
sserver.close()
```