Licence Professionnelle ASSR MRIT
Université Sorbonne Paris Nord
(USPN)

Module M32
Partie « Initiation client-serveur »

Jean-Vincent Loddo
Département R&T – IUT de Villetaneuse

Supports

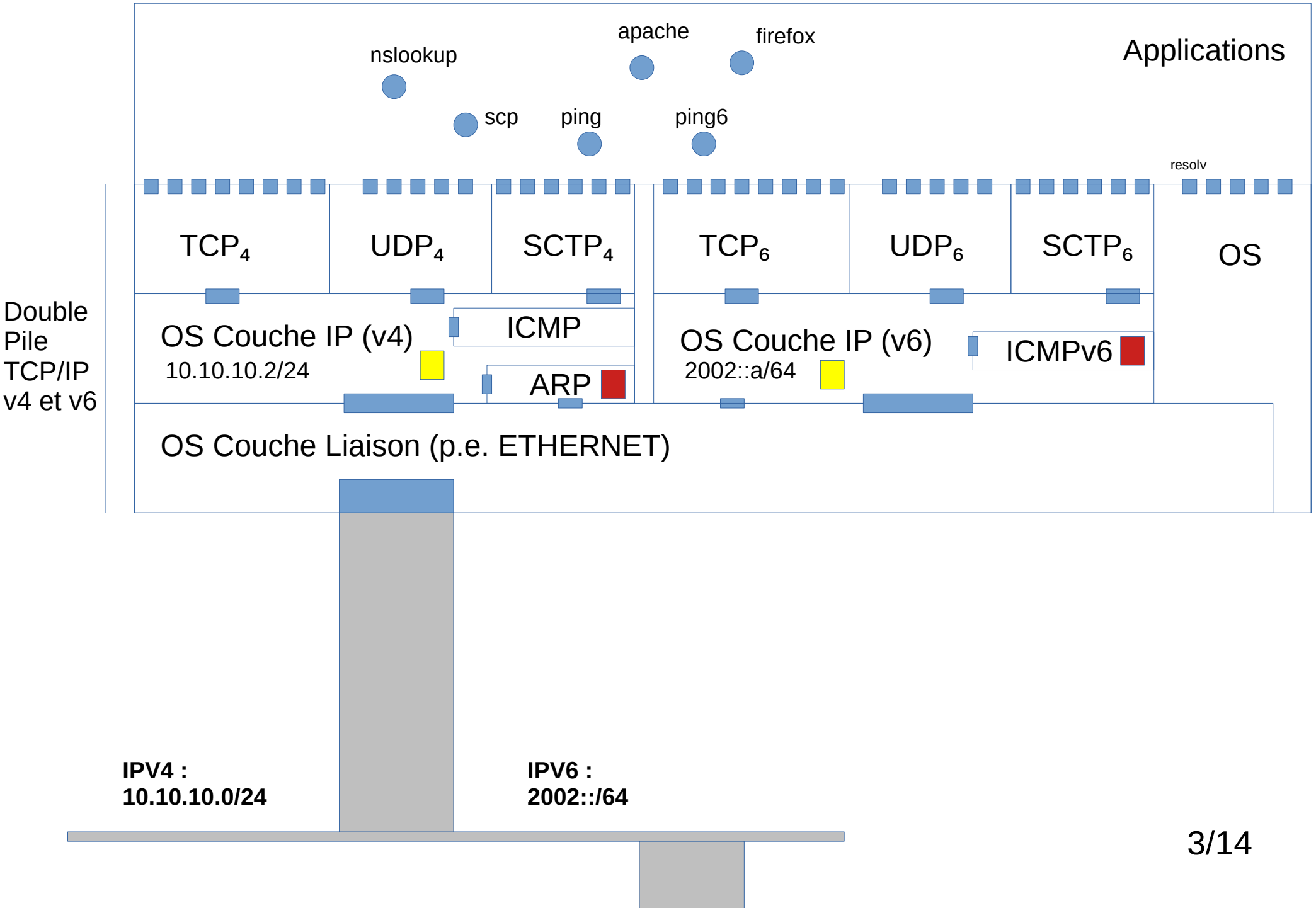
- **Ce document**

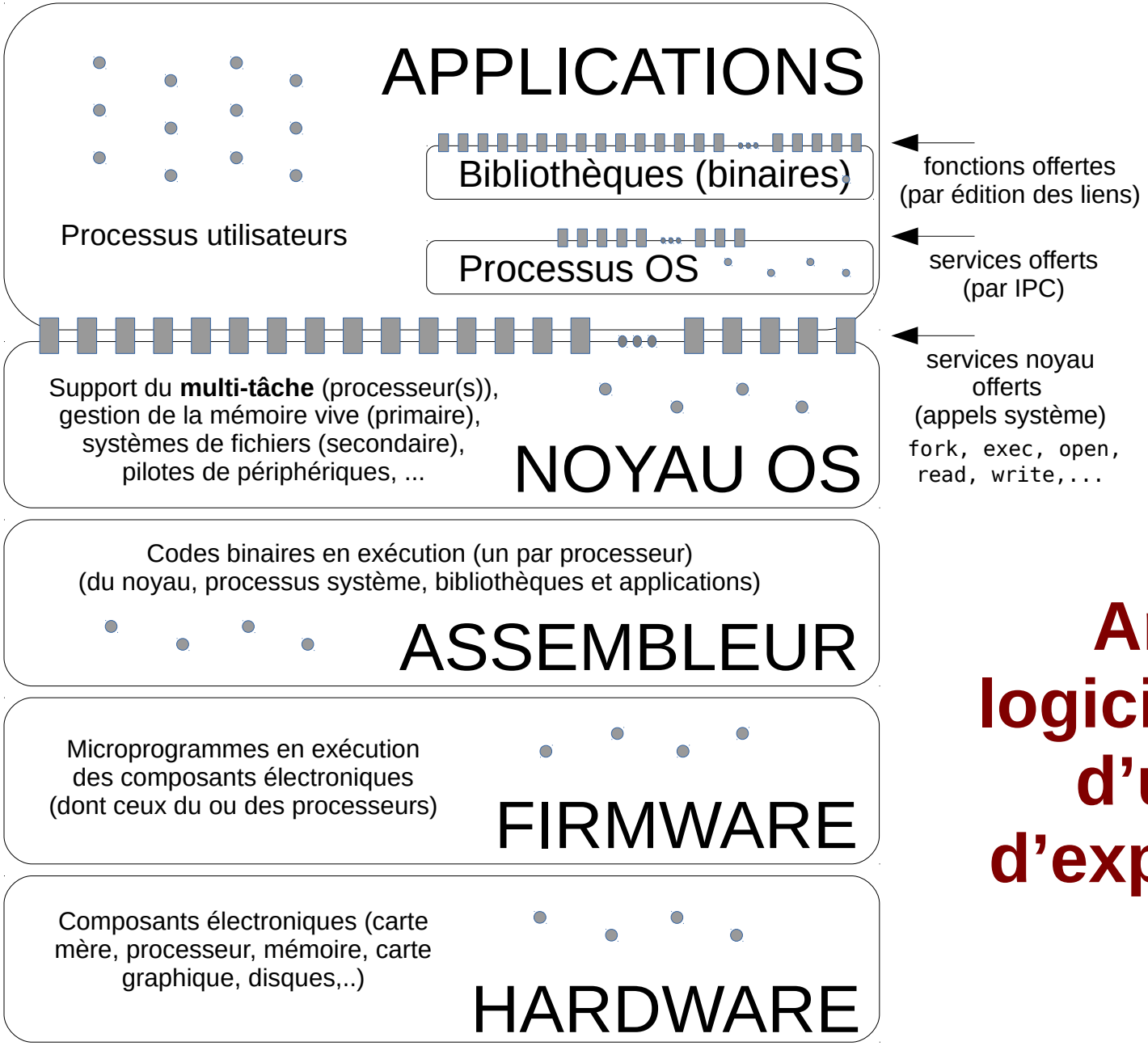
qui complète vos notes de cours **manuscrites**
en reproduisant la plupart des contenus dessinés au tableau par votre enseignant

- **Logiciel (pour les travaux pratiques) :**

Marionnet
J.V. Loddo et L. Saiu

Téléchargeable à l'adresse :
`https://www.marionnet.org/downloads/Ubuntu_16.04_LTS_20180905.ova`


Applications
nslookup
apache
firefox
scp
ping
ping6
resolv
TCP4
UDP4
SCTP4
TCP6
UDP6
SCTP6
OS
Double
Pile
TCP/IP
v4 et v6
OS Couche IP (v4)
10.10.10.2/24
ICMP
ARP
OS Couche IP (v6)
2002::a/64
ICMPv6
OS Couche Liaison (p.e. ETHERNET)
IPV4 :
10.10.10.0/24
IPV6 :
2002::/64

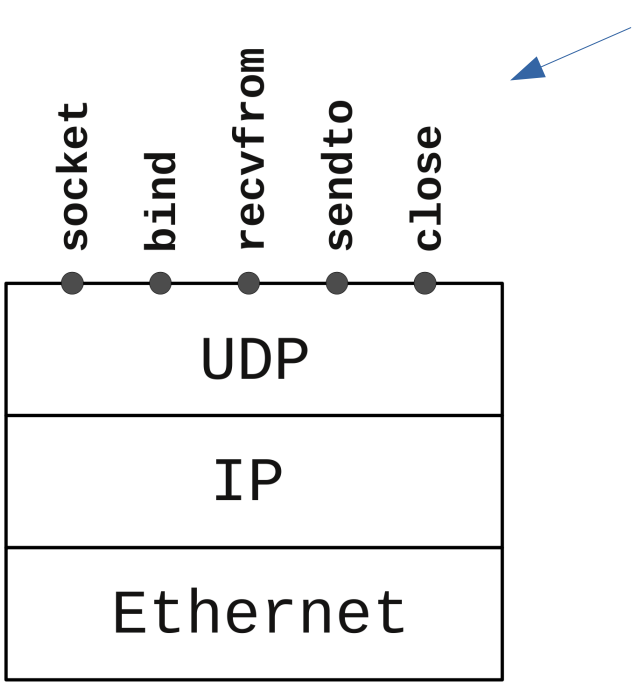


Architecture logicielle/matérielle d'un système d'exploitation (OS)

UDP = IP + ports + interface logicielle

Un **port** (UDP) est une sorte d'adresse numérique (sur 16 bits) qui peut être associée à un processus qui le demande (ou qui en demande un **quelconque**). Autrement dit, un port est une **sorte d'identité** (UDP) pour un processus sur le réseau.

Le **nombre de ports** (UDP) possibles et attribuables (à ses processus) est 2^16 = 2^10 * 2^6 = **64 Ki**



Interface logicielle :

socket : obtenir un port quelconque (49152..65535)

bind : obtenir un port particulier (p.e. bien connu 0..1023)

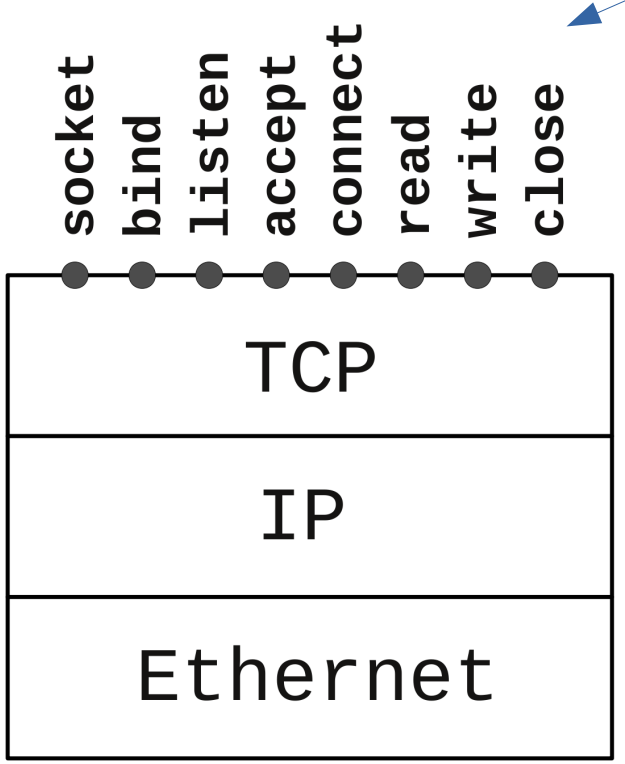
sendto : envoyer un message à un interlocuteur (adresse-IP-DST, UDP, port-DST)

recvfrom : recevoir un message ; une fois reçu, on pourra connaître l'interlocuteur (adresse-IP-SRC, UDP, port-SRC)

close : libérer le port

TCP = IP + ports + interface logicielle
+ garantie du transport (acquittements)
+ longueur illimité (gestion de la fragmentation)
+ contrôle de flux (fenêtres)
(le tout en full-duplex)

Un **port** (TCP) est une sorte d'adresse numérique (sur 16 bits) tout comme pour les ports UDP. Le **nombre de ports** (TCP) possibles et attribuables (à ses processus) est 2^16 = 2^10 * 2^6 = **64 Ki**



Interface logicielle

socket : obtenir un port quelconque (49152..65535)

bind : obtenir un port particulier (p.e. bien connu 0..1023)

listen : indiquer à TCP qu'il s'agit d'un port « d'écoute » (pas de « service » ou de « dialogue » avec le client)

accept : se suspendre en demandant à TCP d'être réveillé lorsque un client demande une connexion

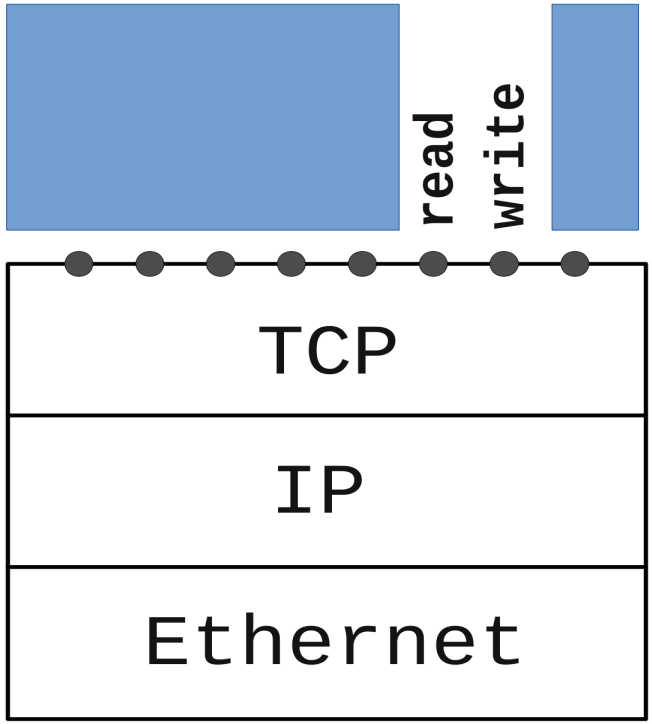
connect : demander à TCP de se connecter à un interlocuteur (adresse-IP-DST, TCP, port-DST)

read : se suspendre en attente de recevoir des caractères depuis l'interlocuteur connecté

write : envoyer des caractères à l'interlocuteur connecté

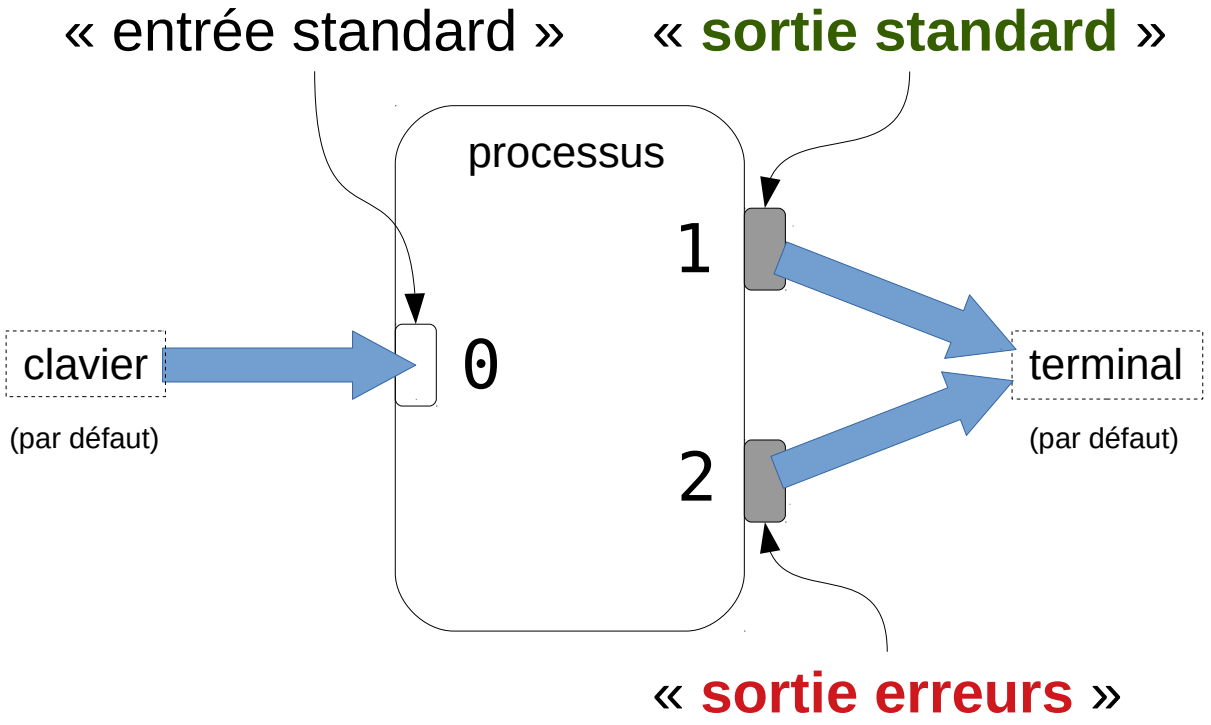
close : libérer le port

socat / netcat = oublier l'aspect administratif des sockets (se concentrer sur la communication)



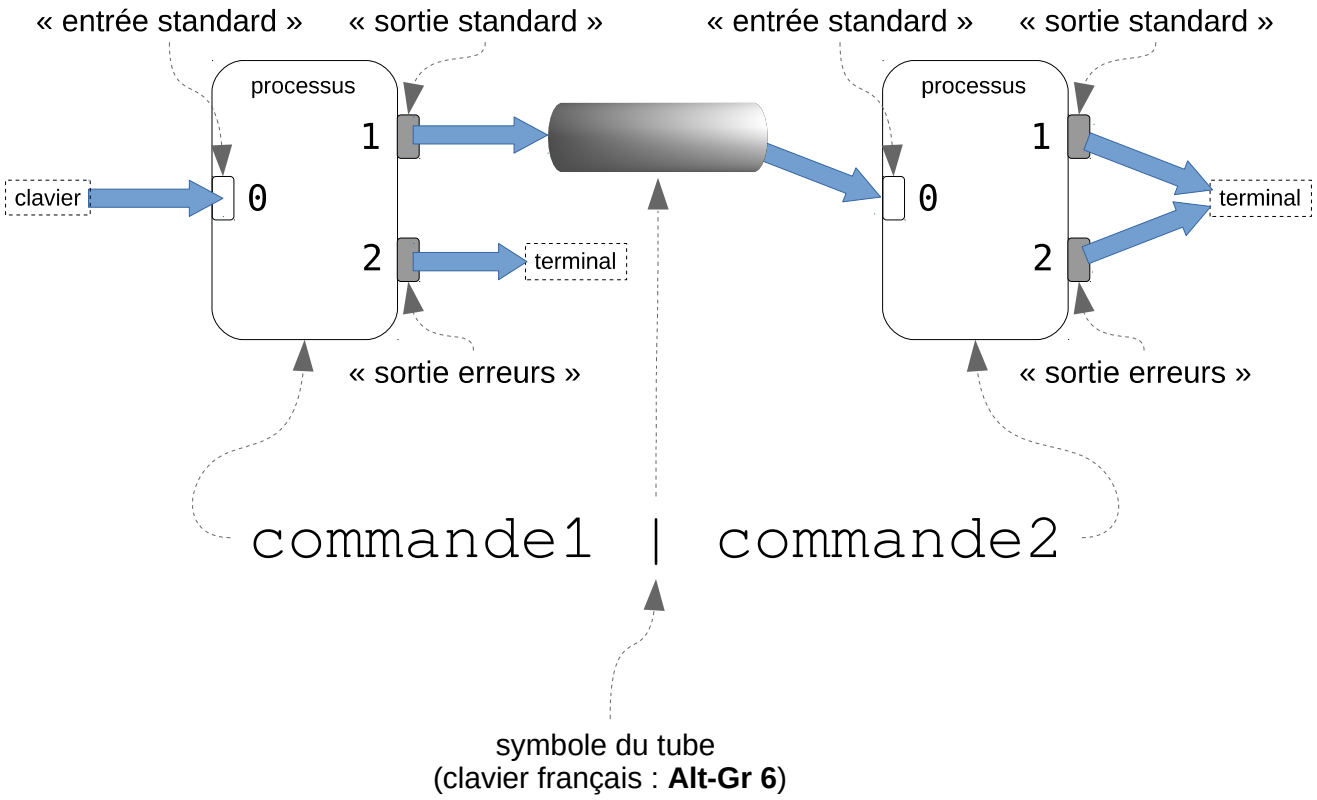
- Sous Unix (p.e. GNU/Linux) on programmera le ***logiciel client*** et/ou le ***logiciel serveur*** en travaillant avec l'entrée/sortie standard (canaux 0 et 1)
- Par exemple, en Bash par des **`read`**/**`echo`**, en Python par des **`input`**/**`print`**, en C par des **`scanf`**/**`printf`**

Canaux standards d’un processus Unix (OS de la famille Unix, p.e. Linux)

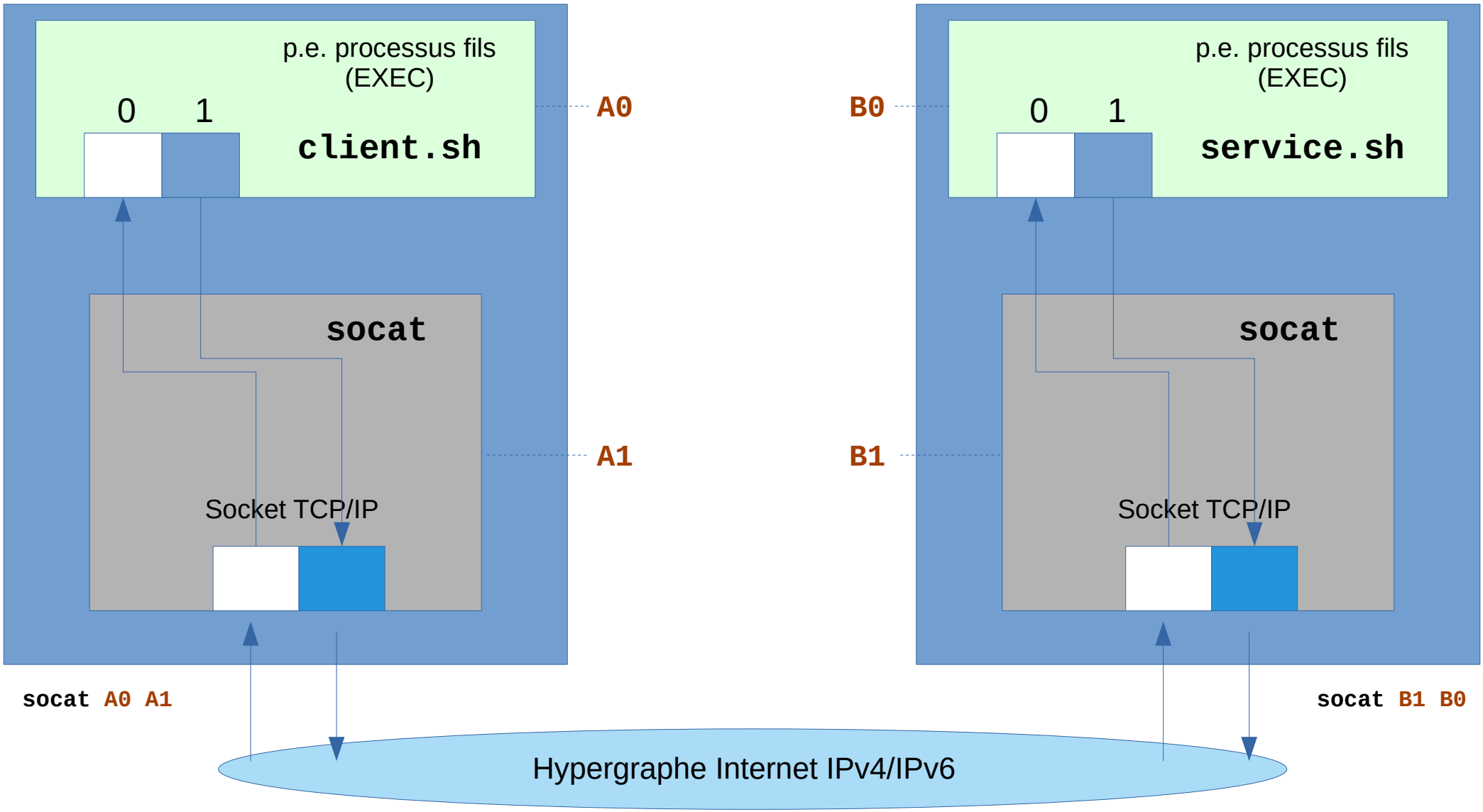


Tubes ou pipelines (chaînes de montage) de processus Unix

Tubes (pipelines)



socat
(principe)



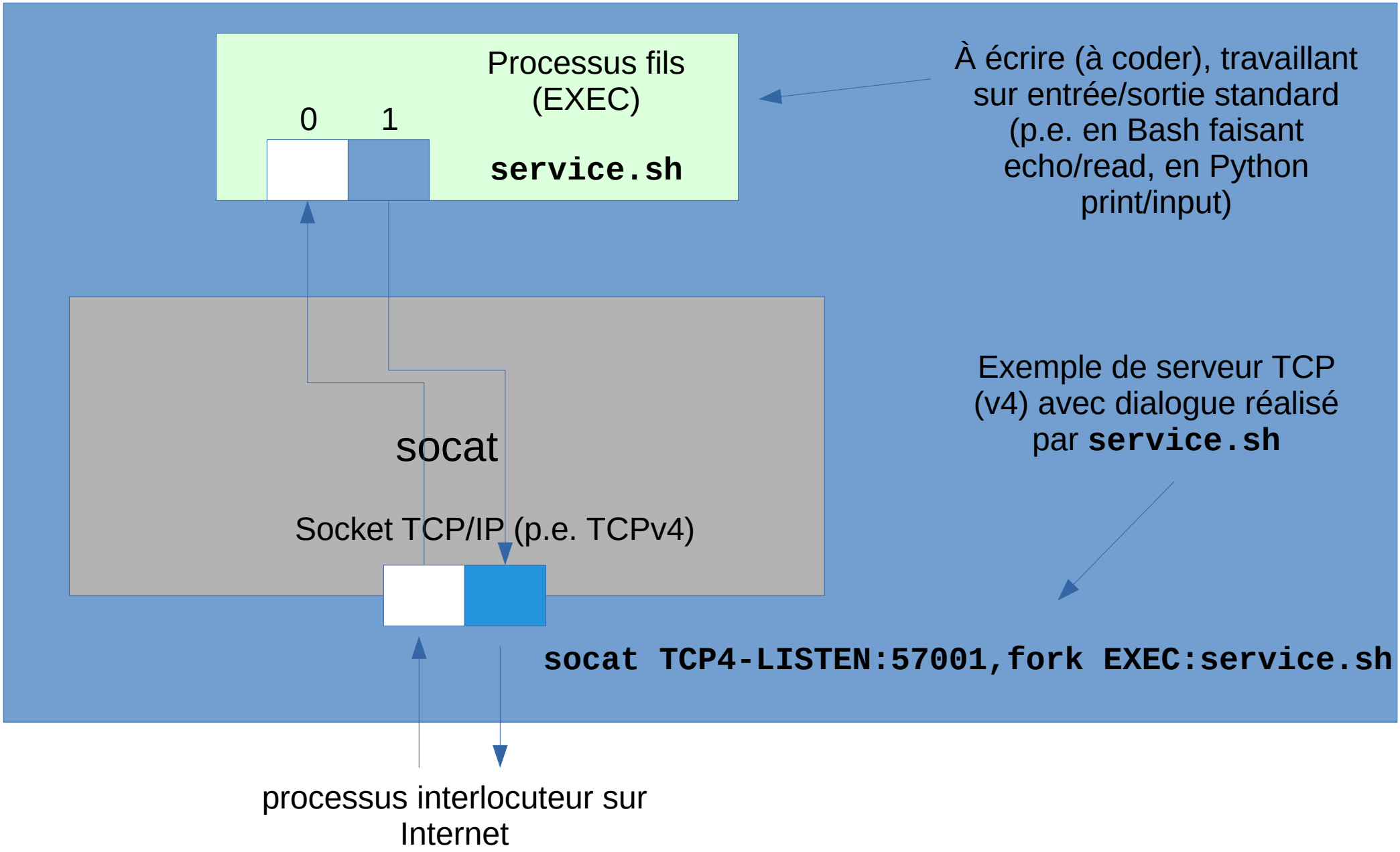
Exemple

```
socat  EXEC:client.sh  TCP4:20.30.40.50:57001
```

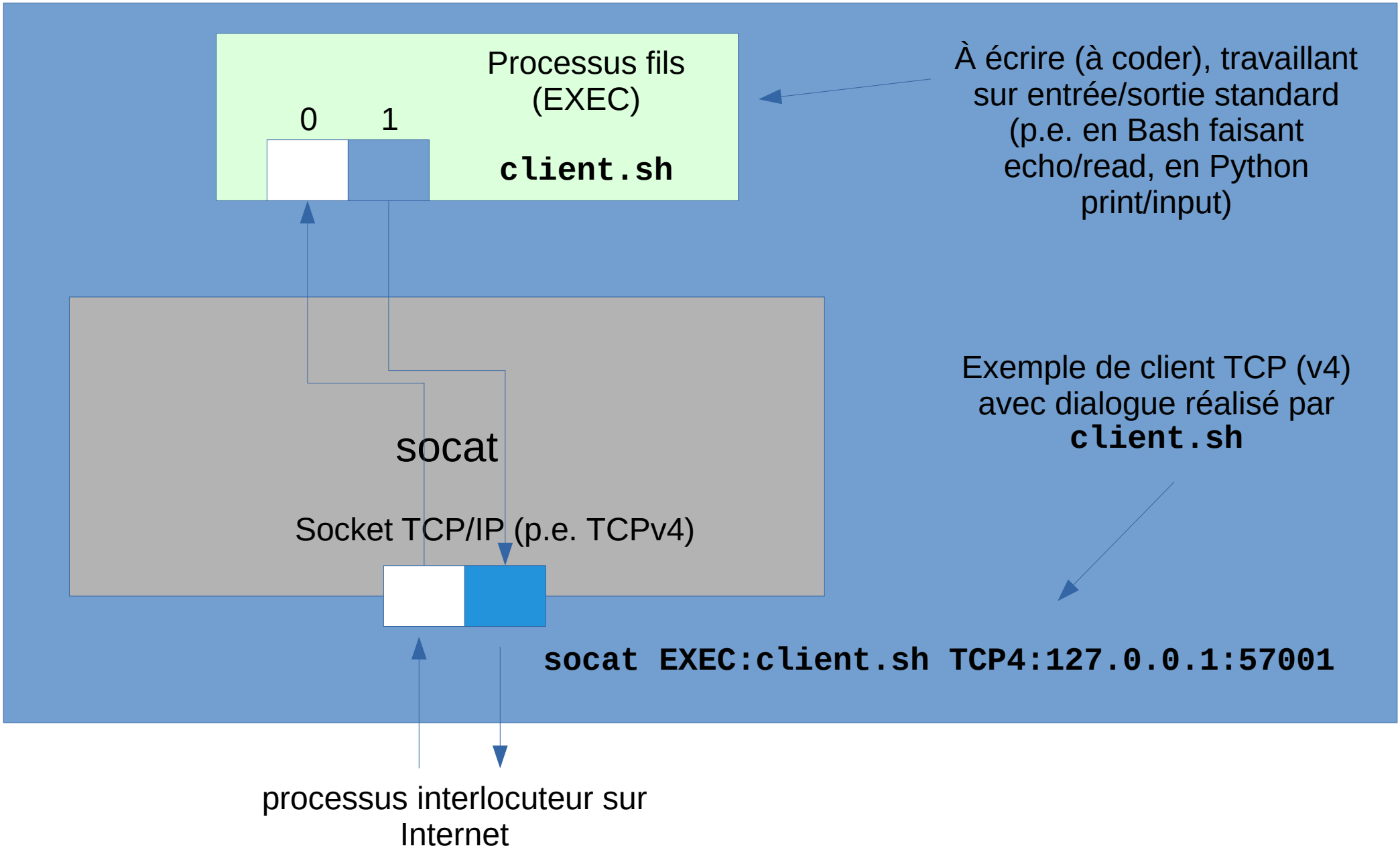
Exemple

```
socat  TCP4-LISTEN:57001,fork  EXEC:service.sh
```

socat
(exemple serveur)

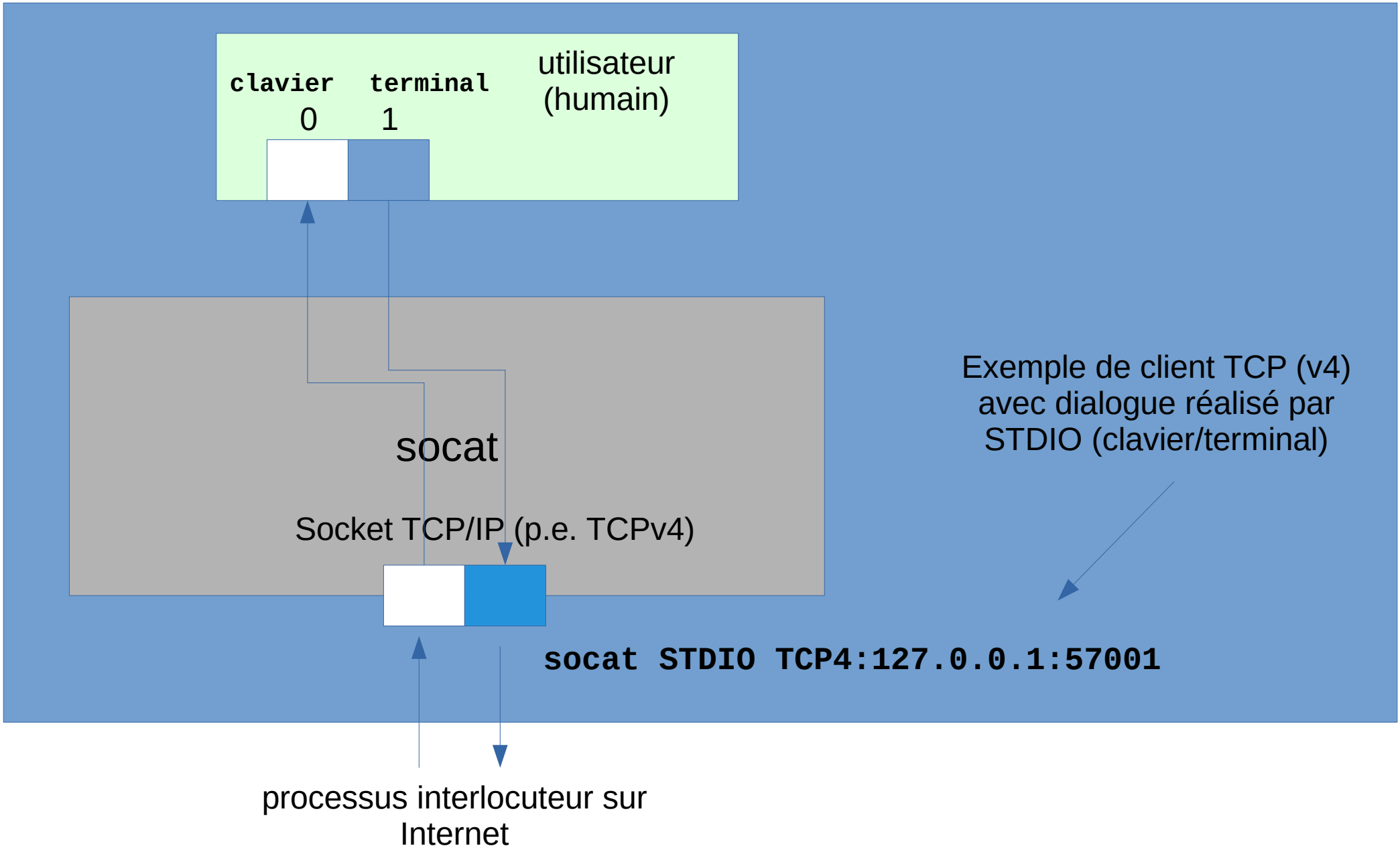


socat
(exemple client)



socat

(exemple client = utilisateur humain)



TP client/serveur avec Mariotel

Sachant que la commande Unix :

```
date "+%s%N"
```

produit (sur sa sortie standard 1) un identifiant entier unique (tenant sur 128 bits), **écrire et tester** un code Bash serveur **fournissant à chaque client un identifiant numérique unique**

- tester avec un client (`socat`) sur la même machine (destinataire `127.0.0.1`)
- tester avec un client (`socat`) depuis la machine (workstation) d'un(e) camarade de la classe Mariotel
- répliquer (et tester) le service en IPv6, de façon à fournir un service **double pile** TCP/IP v4 & v6