Projet Unix
IESE4 – S8

Projet Client - Serveur

Équipe d'enseignement : E. Gascard, F. Rousseau
A. Skaf, V. Isaac–Chassande, T. Bricout

Responsable du projet : Frederic.Rousseau@univ-grenoble-alpes.fr

2022–2023

Introduction

Prérequis et enseignants

Prérequis

- Programmation système sous Unix (CM, TD)
- Algorithmique et Programmation (S5, S6)

Enseignants

Groupes 1 et 2

- Frédéric Rousseau <frederic.rousseau@univ-grenoble-alpes.fr>
- Thaddée Bricout <thaddee.bricout@cea.fr>

Groupes 3 et 4

- Ahmad Skaf <ahmad.skaf@univ-grenoble-alpes.fr>
- Valentin Isaac–Chassande <valentin.isaacchassande@cea.fr>

Organisation et évaluation

Organisation :

- **Présentiel (salle 216)** : 2 enseignants assurent l'encadrement du projet 4h par semaine, sur les créneaux indiqués sur ADE.
- **Support par mail** : possible sur des questions spécifiques.

Évaluation :

- **À la fin de chaque séance** : il est indispensable de remplir une **feuille de contrôle** décrivant le travail effectué. Cette feuille doit être validée par un enseignant.
- **Date limite de rendu du projet** : fin de la 4ème séance.

Le travail en binôme est encouragé !

Objectif

Objectif du projet

Écrire un **programme client** qui affiche en temps réel les données numériques générées par un programme serveur

Serveur

- Le code source du serveur est fourni
- Les fichiers se trouvent dans le répertoire :

  `/home/iese-2024/Partage/Projet_Unix/`
- Copier et décompresser le fichier **`Serveur.tgz`**
- Compilation et exécution (voir fichier `README`)

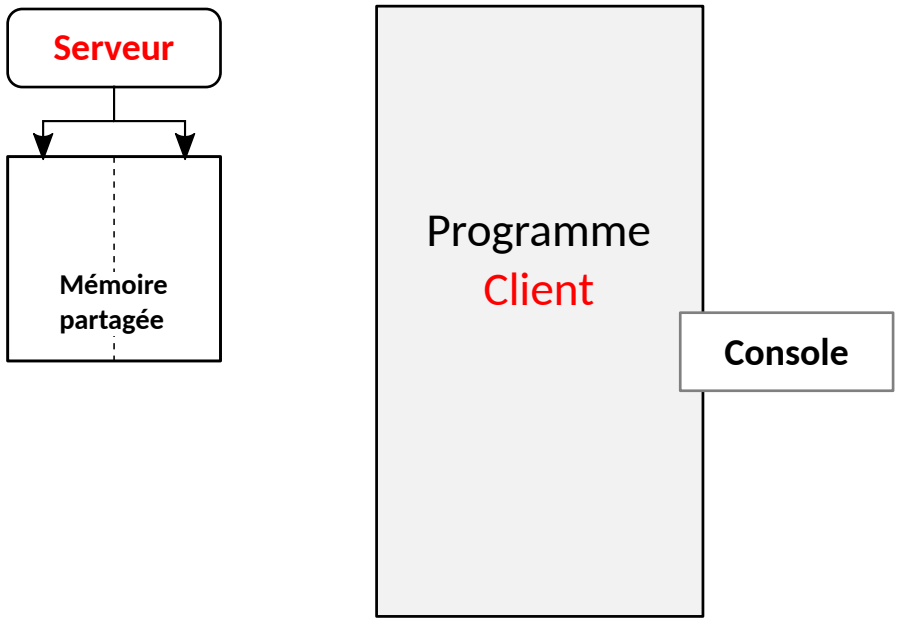
Client

- Vous disposez des fichiers :

  `CL_include` et **`CL_def.h`**

Description des séances

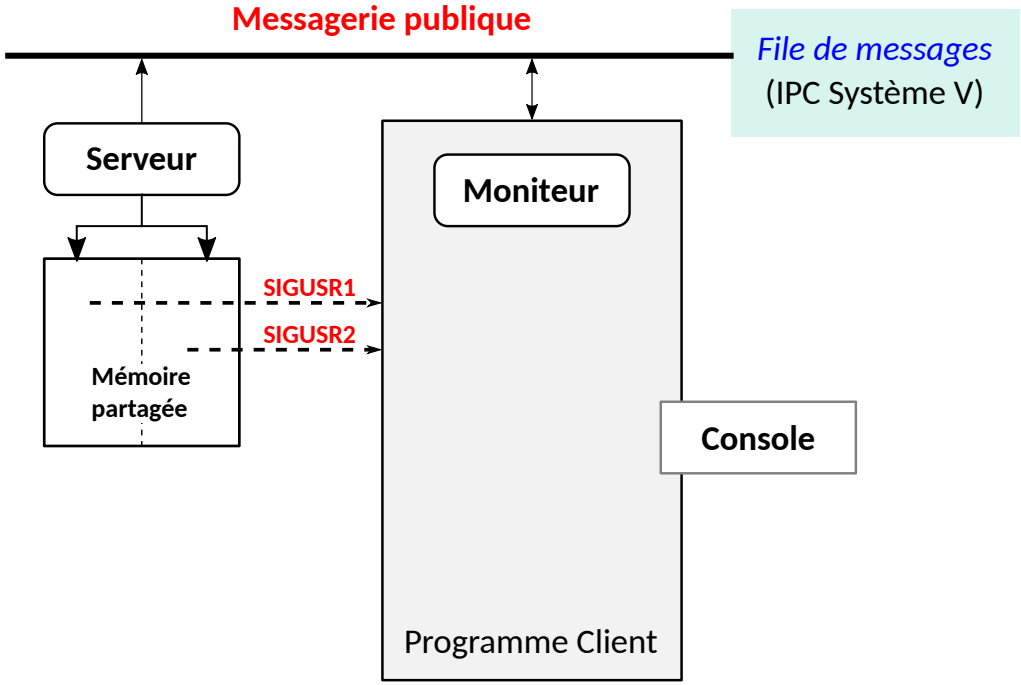
Projet Client - Serveur



- Le Serveur génère des données toutes les secondes
- Ces données sont écrites dans l'une des `NVOIES` d'entrée
- Le Client lit chaque donnée écrite dans la mémoire partagée et l'affiche dans un terminal

Première séance

Première séance



Connexion client-serveur :

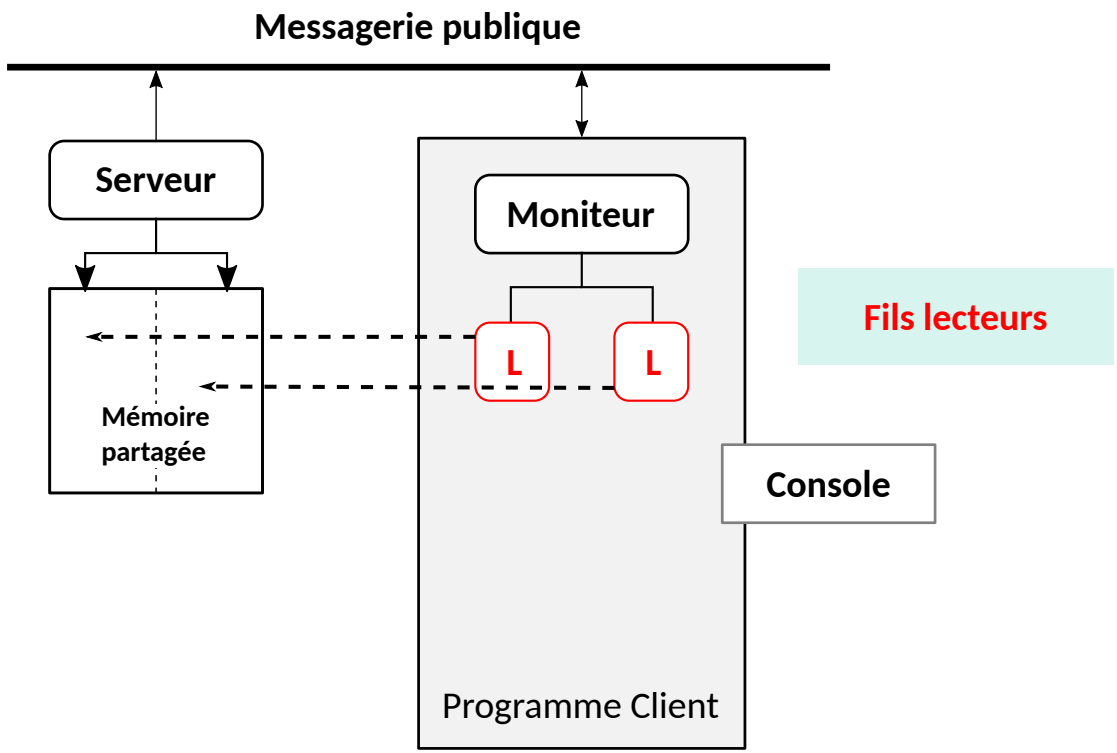
1. Le **Client** identifie la messagerie (en utilisant le fichier `CleServeur`)
2. Le **Client** demande la connexion au serveur (en envoyant un message de type `CONNECT`)
3. Le **Client** reçoit la clé d'accès (`CleClient`) (c'est le **Serveur** qui la fournit)
4. Le **Client** envoie un acknowledgement pour confirmer la réception de la clé (`CleClient`)

Attachement et lecture de la mémoire partagée :

5. Le **Client** obtient l'id (`shmget`) et attache le segment de *mémoire partagée* (`shmat`)
6. Le **Client** reçoit le *signal* du Serveur qui informe de l'arrivée d'une donnée
7. Le **Client** lit une des voies de la *mémoire partagée* et affiche les données

Deuxième séance

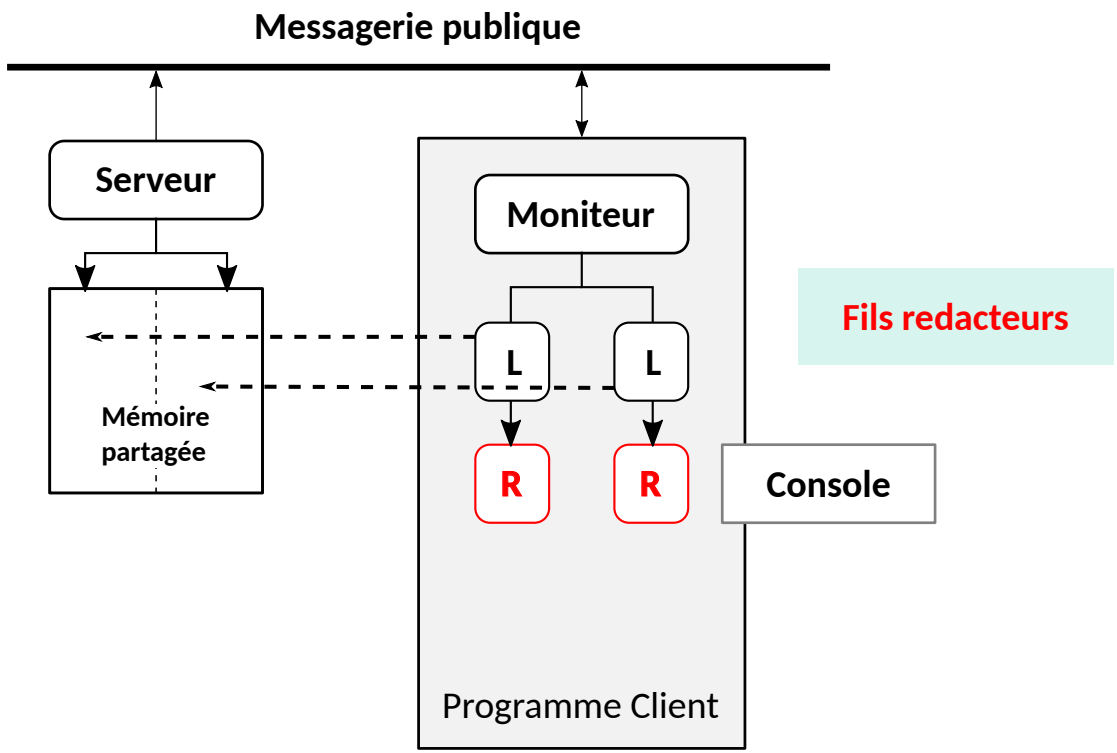
Deuxième séance



1. Le **Serveur** informe de l'arrivée d'une donnée par des *signaux*
2. Le **Moniteur** (ou le Client ou encore le processus père) transmet cette information au lecteur. Pour cela, il utilise des *sémaphores* (`V()` fonction)
3. Les **Lecteurs** (L) attendent sur les *sémaphores* (`P()` fonction)
4. À l'arrivée d'une nouvelle donnée, un **lecteur** accède à la *mémoire partagée* (l'accès se fait par un *sémaphore* déjà créé par le serveur)

Troisième séance

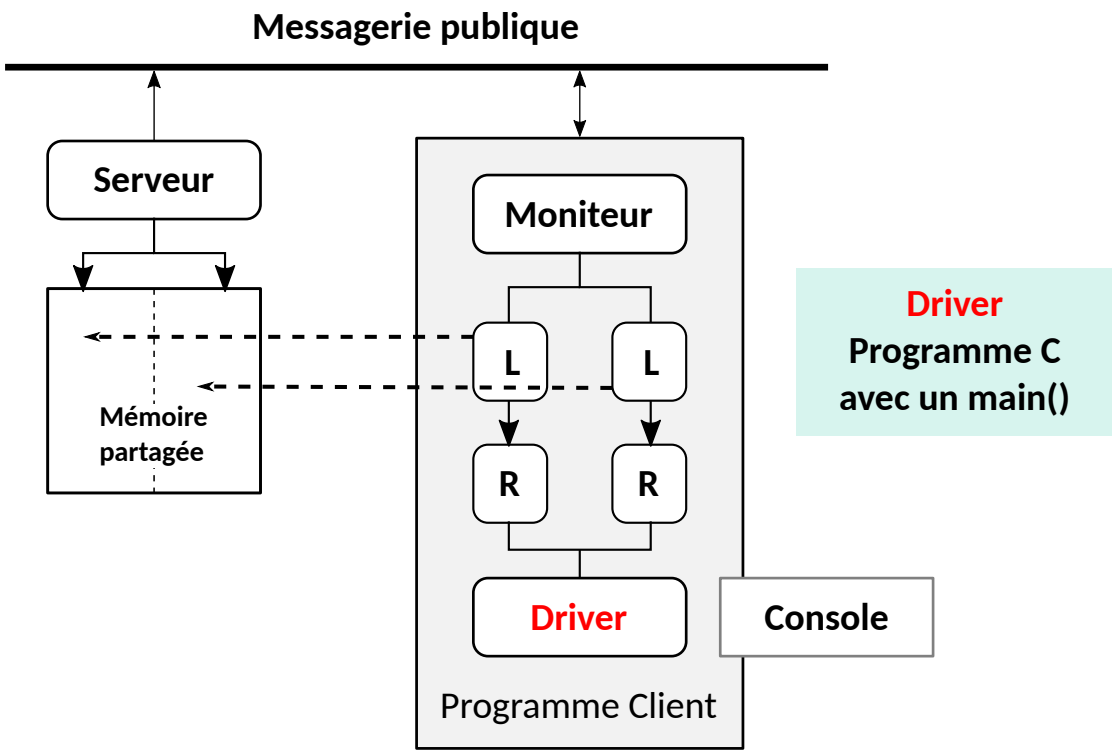
Troisième séance



1. Le **Lecteur (L)** fait la lecture des données écrites dans la mémoire partagée
   Attention : il crée une chaîne de caractères (donnée + date) et l'envoie au **R**
2. Le **Rédacteur (R)** fait l'affichage.
   Attention : il attend la réception de 5 données avant de faire l'affichage
3. Un *pipe* doit être utilisé pour la communication entre le lecteur et le rédacteur

Quatrième séance

Quatrième séance



1. Le **Rédacteur (R)** demande l'accès au Driver, transmet les données et libère la ressource Driver.
2. L'affichage est maintenant géré par le **Driver**
3. Le **Driver** communique avec un seul des **rédacteurs (R)** à la fois
4. Un seul *pipe* est utilisé pour la communication entre les rédacteurs et le driver
5. La synchronisation se fait à l'aide d'un *sémaphore* d'exclusion mutuelle

Gestion de processus

Mécanismes de gestion de processus

Gestion de processus sous UNIX

- **Création/terminaison** : `fork()`, `wait()` et `waitpid()`, `exit()`, `getpid()` et `getppid()`, `execlp()`.
- **Communication** : files de messages, mémoire partagée, pipes.
- **Synchronisation** : sémaphores, signaux.

Interface	File de messages	Mémoire partagée	Sémaphores	Signaux	Pipes
Créer/ouvrir	`msgget()`	`shmget()` + `shmat()`	`semget()`	–	`pipe()`
Fermer	–	`shmdt()`	–	–	`close()`
Opération IPC	`msgsnd()` `msgrcv()`	accès à la mémoire dans la région partagée	`semop()`	`signal()` `kill()`	`read()` `write()`
Gestion/contrôle	`msgctl()`	`shmctl()`	`semctl()`	–	–

IPC Système V – Files de messages (I)

- Structure générale d'un message :

```
struct mymsg {
   long type;        /* Type de message : CONNECT, ACK, pid d'un proc, etc. */
   char txt[L_MSG]; /* Contenu du message */
}
```

- `msgget()` permet d'obtenir l'identifiant d'une file de messages existant ou d'en créer une si besoin à partir d'une clé

```
int msgget (key_t cle, int option)
```

- `msgsnd()` permet d'envoyer des messages. Cette fonction utilise l'identifiant de la file de messages `msgid`

```
int msgsnd (int msgid, const void* msg, int taille, int op)
```

IPC Système V – Files de messages (II)

- `msgrcv()` permet de lire un message. Cette fonction utilise l'identifiant de la file de messages

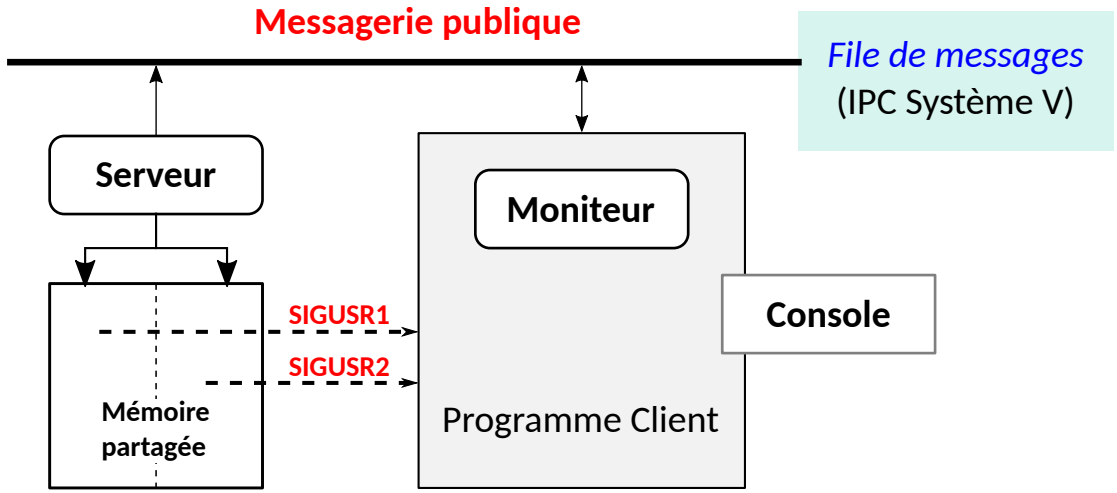
```
int msgrcv (int msgid, void* msg, int taille, long type, int op)
```

- `msgctl()` permet de paramétrer ou de supprimer la file de messages

```
int msgctl (int msgid,int op, struct msgid_ds *p_msg)
```

Connexion client-serveur

IPC Système V – Files de messages (II) — Connexion client-serveur



- Le **Client** obtient l'identifiant de la file de messages (fonction `msgget()`)
- Le **Client** s'identifie en émettant un message de type `CONNECT` contenant son PID (fonction `msgsnd()`)
- Le **Serveur** répond avec un message contenant le nom du fichier `CleClient` (le **Client** attend la réponse du serveur - fonction `msgrcv()`)
- Le Client envoie un acquittement en émettant un message de type `ACK` qui contient le `PID` du client (fonction `msgsnd()`)

Connexion Client - Serveur

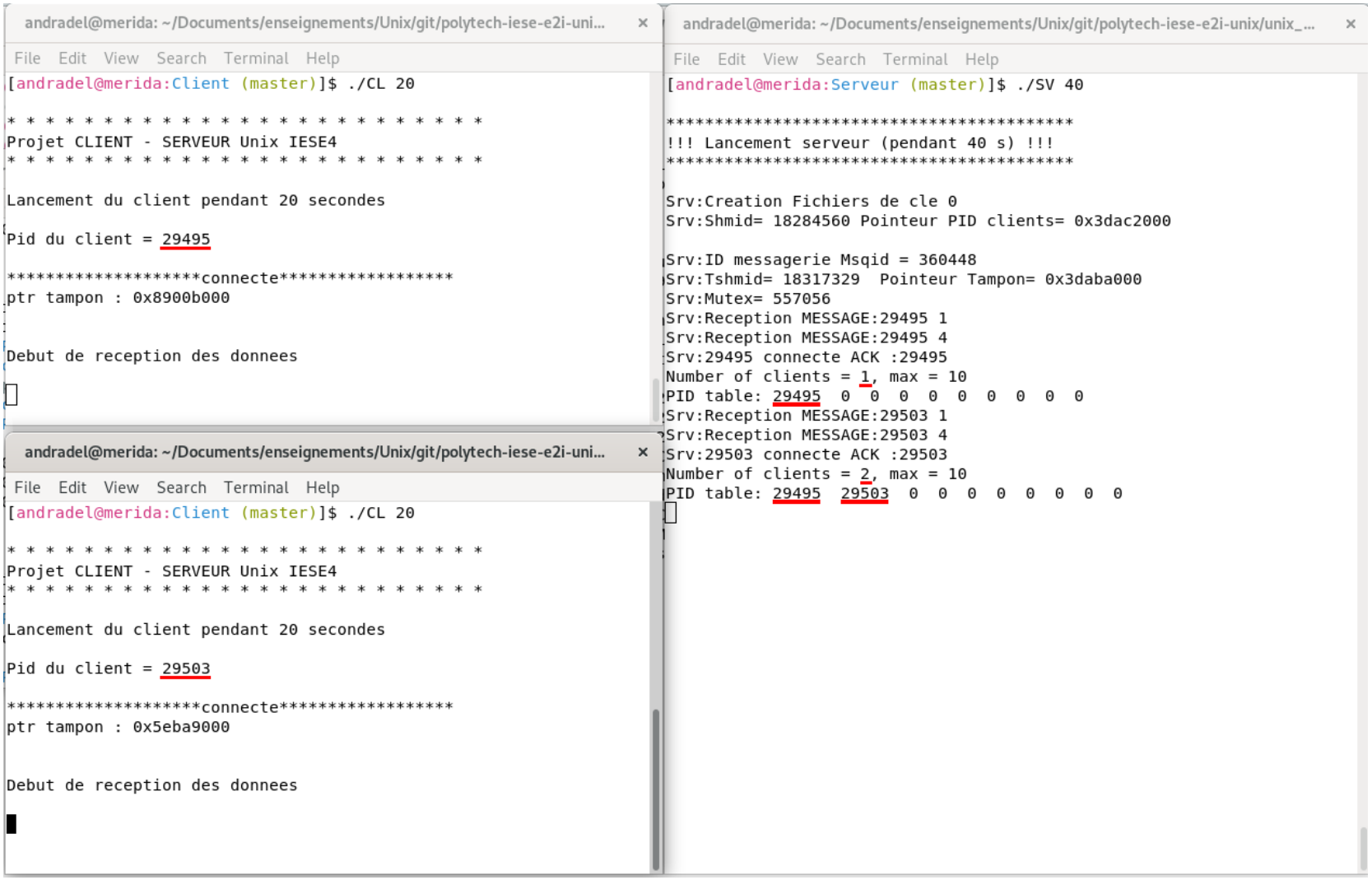
```
File  Edit  View  Search  Terminal  Help
[andradel@merida:Client (master)]$ ./CL 20

* * * * * * * * * * * * * * * * * * * * * * * * *
Projet CLIENT - SERVEUR Unix IESE4
* * * * * * * * * * * * * * * * * * * * * * * * *

Lancement du client pendant 20 secondes

Pid du client = 29433

********************connecte******************
ptr tampon : 0x34202000


Debut de reception des donnees

Donnee no 0 de la voie 1 : 1 le Tue Mar 15 22:36:05 2022
Donnee no 1 de la voie 1 : 2 le Tue Mar 15 22:36:06 2022
Donnee no 2 de la voie 1 : 3 le Tue Mar 15 22:36:07 2022
Donnee no 3 de la voie 1 : 4 le Tue Mar 15 22:36:08 2022
Donnee no 4 de la voie 1 : 5 le Tue Mar 15 22:36:11 2022

Donnee no 0 de la voie 0 : 3599 le Tue Mar 15 22:36:09 2022
Donnee no 1 de la voie 0 : 3598 le Tue Mar 15 22:36:10 2022
Donnee no 2 de la voie 0 : 3597 le Tue Mar 15 22:36:13 2022
Donnee no 3 de la voie 0 : 3596 le Tue Mar 15 22:36:15 2022
Donnee no 4 de la voie 0 : 3595 le Tue Mar 15 22:36:18 2022

Donnee no 0 de la voie 1 : 6 le Tue Mar 15 22:36:12 2022
Donnee no 1 de la voie 1 : 7 le Tue Mar 15 22:36:14 2022
Donnee no 2 de la voie 1 : 8 le Tue Mar 15 22:36:16 2022
Donnee no 3 de la voie 1 : 9 le Tue Mar 15 22:36:17 2022
Donnee no 4 de la voie 1 : 10 le Tue Mar 15 22:36:23 2022


********************deconnecte**********************
[andradel@merida:Client (master)]$
```

Client

```
File  Edit  View  Search  Terminal  Help
[andradel@merida:Serveur (master)]$ ./SV 40

****************************************
!!! Lancement serveur (pendant 40 s) !!!
****************************************

Srv:Creation Fichiers de cle 0
Srv:Shmid= 18219024 Pointeur PID clients= 0xe66fb000

Srv:ID messagerie Msqid = 327680
Srv:Tshmid= 18251793  Pointeur Tampon= 0xe66f3000
Srv:Mutex= 491520
Srv:Reception MESSAGE:29433 1
Srv:Reception MESSAGE:29433 4
Srv:29433 connecte ACK :29433
Number of clients = 1, max = 10
PID table: 29433  0  0  0  0  0  0  0  0  0
Srv:Reception MESSAGE:29433 2
Srv:NB_clients : 0; pid :29433 deconnecte
Number of clients = 0, max = 10
PID table: 0  0  0  0  0  0  0  0  0  0

Srv:FIN Del.Fich.Cle 0

Srv:FIN Mort du fils receptionniste
Srv:FIN Mort du fils informateur

Srv:FIN RelachMsg 0
Srv:FIN RelachMem 0
Srv:FIN Mutex 0
[andradel@merida:Serveur (master)]$
```

Serveur

Connexion deux Clients - Serveur



Deux clients **Serveur**

Remarques finales

Remarques finales

- Ne pas modifier l'architecture imposée :
  - Client-Serveur : *file de messages*
  - Moniteur-Lecteur : *sémaphores*
  - Lecteur-SHM : *sémaphore créé par le serveur*
  - Lecteur-Rédacteur : *pipe*
  - Rédacteur-Driver : *pipe et sémaphore*
- Créer un dossier par séance. Chaque dossier montrera l'évolution de votre code par rapport à la séance précédente.
- **Faites valider votre code par un enseignant**
- N'oubliez pas de gérer les erreurs et de commenter votre code.