

## TP4 - IPC – File de messages

ING2-GSI – Programmation Système et Réseau

Année 2010 – 2021



### 1 Une file de messages – IPC Système V

#### 1.1 Un exemple simple d'utilisation d'une file de messages dans une architecture de type client/serveur

Cet exemple montre une utilisation d'une file de messages dans une architecture de type client/serveur [1]. Une fois lancé, le serveur attend une requête de clients (message dont le type est égal à 1) dans la file de messages. Le corps de la requête comprend un tableau de caractères et un entier indiquant le PID du processus client ayant émis la requête. Le client place dans le tableau de caractères une chaîne que le serveur doit inverser avant de la renvoyer comme réponse au processus ayant émis la requête. Le type de la réponse est tout simplement le PID du client.

Les fichiers sources sont présents sur Teams et consistent en trois fichiers, le fichier header **clsrv.h** et les deux fichiers sources **serveur.c** (pour le serveur) et **client.c** (pour le client). Après avoir compilé les deux sources, vous possédez deux exécutables : serveur et client. Ouvrez deux terminaux supplémentaires et organisez vos trois terminaux de sorte qu'ils soient tous visibles sur votre écran. Tous les terminaux doivent être à l'emplacement de vos deux exécutables. Lancez dans le premier terminal le serveur : **./serveur**. Ce terminal reste bloqué, vous y lirez les indications du serveur.

Dans le deuxième terminal, vous pouvez tout d'abord vérifier que la file de message est bien présente grâce à la commande **ipcs**. Vous devriez obtenir une sortie analogue à ce qui suit :

```
smv@smv:~/EISTI/Enseignement/Progr_Syst_et_Reseau/4-File_de_messages/exemple$ ipcs

----- Message Queues -----
key      msqid      owner      perms      used-bytes   messages
0x30051efa 0              smv        640          0             0

----- Shared Memory Segments -----
key      shmid      owner      perms      bytes       nattch     status
0x00000000 32771       smv        600        53248        2         dest
0x00000000 32772       smv        600        53248        2         dest
0x00000000 8           smv        600       134217728    2         dest
0x00000000 32777       smv        600       67108864    2         dest
0x00000000 60          smv        600       524288       2         dest

----- Semaphore Arrays -----
key      semid      owner      perms      nsems
```

Vous pouvez aussi vérifier que le serveur est en fonction en exécutant la commande **ps u** :

```
smv@smv:~/EISTI/Enseignement/Progr_Syst_et_Reseau/4-File_de_messages/exemple$ ps -u

USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
smv      10862  0.0  0.0  29924  5532 pts/0    Ss   09:10   0:00 bash
smv      10873  0.0  0.0  29952  5284 pts/1    Ss   09:10   0:00 bash
smv      11616  0.0  0.0   4372   796 pts/1    S+   09:45   0:00 ./serveur
smv      11625  0.0  0.0  46772  3720 pts/0    R+   09:45   0:00 ps -u
```

Lancez ensuite dans ce deuxième terminal un client : **./client**. Vous devriez obtenir quelque chose ressemblant à ce qui suit :

```
smv@smv:~/EISTI/Enseignement/Progr_Syst_et_Reseau/4-File_de_messages/exemple$ ./client
Client operationnel
--> █
```

Votre client attend une chaîne de caractères. Si vous saisissez la chaîne : « Bonjour », vous devriez observer dans le deuxième terminal quelque chose analogue à :

```
smv@smv:~/EISTI/Enseignement/Progr_Syst_et_Reseau/4-File_de_messages/exemple$ ./client
Client operationnel
--> Bonjour
Requete de 11648 envoyee: ->Bonjour<-
11648: reponse du serveur ->ruojnoB<-
--> █
```

tandis que dans la fenêtre du premier terminal, vous devriez obtenir quelque chose ressemblant à:

```
smv@smv:~/EISTI/Enseignement/Progr_Syst_et_Reseau/4-File_de_messages/exemple$ ./serveur
Serveur: requete ->Bonjour<- du processus 11648
Serveur: tmp=7
Serveur: rep.chaine ->ruojnoB<-
Serveur: reponse envoyee a 11648
```

Si vous lancez un autre client dans le troisième terminal, vous pouvez aussi y lancer des requêtes. Pour terminer un client, il suffit d'entrer la requête exit. Après avoir fermé les deux clients, l'arrêt du serveur est obtenu en envoyant le signal SIGINT au processus correspondant. Ce type d'arrêt permet l'effacement de la file de messages, qui disparaît des tables IPC comme vous pouvez le vérifier avec la commande `ipcs`.

## 1.2 Exercice pour les files de messages

En vous inspirant des sources précédents, écrivez une application client/serveur dont le serveur attend des requêtes contenant un entier. Le serveur répond alors au client par un message contenant autant d'entiers tirés aléatoirement que le nombre entier envoyé dans la requête. Le client affiche les entiers renvoyés par le serveur.

## 2 Une file de messages – IPC Posix

Dans la référence [2] vous trouverez un exemple d'utilisation d'une file de message POSIX. En utilisant des fonctions standardisées POSIX, calculer dans le serveur la somme de valeurs envoyées par le client. Le client affiche ensuite la somme renvoyée par le serveur.

### Références :

[1] S. Pillement et V. Gouret, Cours/TP de Systèmes Informatiques, 2018-2019, Polytech Nantes

[2] C. Blaess, Efficacité des IPC : les files de messages Posix, 2011 :

<https://www.blaess.fr/christophe/2011/09/17/efficacite-des-ipc-les-files-de-messages-posix/>