

Linux 2 : Appels Systèmes - BINV2181 (tp08- poll)

8.1. Pierre, papier, ciseaux - appels systèmes : `socket`, `poll`

Ecrivez 2 programmes : un client et un serveur. Le serveur permettra aux clients de rejoindre un salon de jeu (cfr solution ex 7.1) pour ensuite commencer une partie du jeu « pierre, papier, ciseaux ».

Plus précisément, le serveur attendra 15 secondes (ce temps doit être paramétrable) l'inscription de 2 joueurs au maximum. Si le nombre de 2 joueurs n'est pas atteint à la fin de ce délai, la partie est annulée. Sinon la partie démarre. Le serveur attendra une proposition (pierre, papier ou ciseaux) de la part des joueurs, affichera à l'écran les propositions des joueurs et calculera le gagnant. A noter que le serveur doit immédiatement afficher la proposition d'un joueur lorsque celui-ci la lui envoie. Il ne doit pas rester bloqué sur un client.

Le client sera un programme utilisé par les joueurs pour envoyer leur demande d'inscription au serveur. Le client enverra ensuite un message avec une proposition (pierre, papier ou ciseaux).

Indications utiles :

1. Le programme serveur peut être décomposé en 2 phases : la phase inscription et la phase jeu. Faites-en premier la phase inscription, testez-la et ensuite passez à la phase jeu.
2. POLL est-il utile durant la phase d'inscription ? Non !
3. Les 15 secondes d'inscription peuvent se gérer avec une alarme.
4. Les clients ne jouent qu'un seul coup et le serveur affiche le gagnant.

Résultat attendu :

Scénario :

1. Le serveur est lancé
2. Le serveur attend 15 secondes les inscriptions
3. « Oli » s'inscrit
4. « José » s'inscrit
5. Fin des inscriptions, les 10 secondes sont écoulées, la partie démarre
6. « José » envoie P pour papier
7. « Oli » envoie C pour ciseaux
8. « Oli » a gagné

```
iplt@laborezo: ~/ExSocket 8.3
iplt@laborezo:~/ExSocket 8.3$ ./server
Le serveur tourne sur le port : 9502
Inscription demandée par le joueur : oli
Nb Inscriptions : 1
Inscription demandée par le joueur : José
Nb Inscriptions : 2
FIN DES INSCRIPTIONS
José joue PAPIER
oli joue CISEAUX
GAGNANT : oliiplt@laborezo:~/ExSocket 8.3$

iplt@laborezo:~/ExSocket 8.3$ ./client
Bienvenue dans le programme d'inscription au serveur de jeu
Pour participer entrez votre nom :
oli
Réponse du serveur : Inscription acceptée
Envoyez P pour Papier, C pour Ciseaux, R pour Pierre
C
iplt@laborezo:~/ExSocket 8.3$
iplt@laborezo:~/ExSocket 8.3$

iplt@laborezo:~/ExSocket 8.3$ ./client
Bienvenue dans le programme d'inscription au serveur de jeu
Pour participer entrez votre nom :
José
Réponse du serveur : Inscription acceptée
Envoyez P pour Papier, C pour Ciseaux, R pour Pierre
P
iplt@laborezo:~/ExSocket 8.3$
iplt@laborezo:~/ExSocket 8.3$
```

Petites Questions :

- Que fait le processus serveur pendant son temps libre (c'est-à-dire lorsqu'il ne réceptionne pas de message des clients) ?
- Peut-on mieux exploiter « le temps libre » du processus serveur ?
- Augmenter le timeout du poll est-il intéressant ?

8.2 Exercice sur les appels systèmes : `poll`, `pipe` (BONUS)

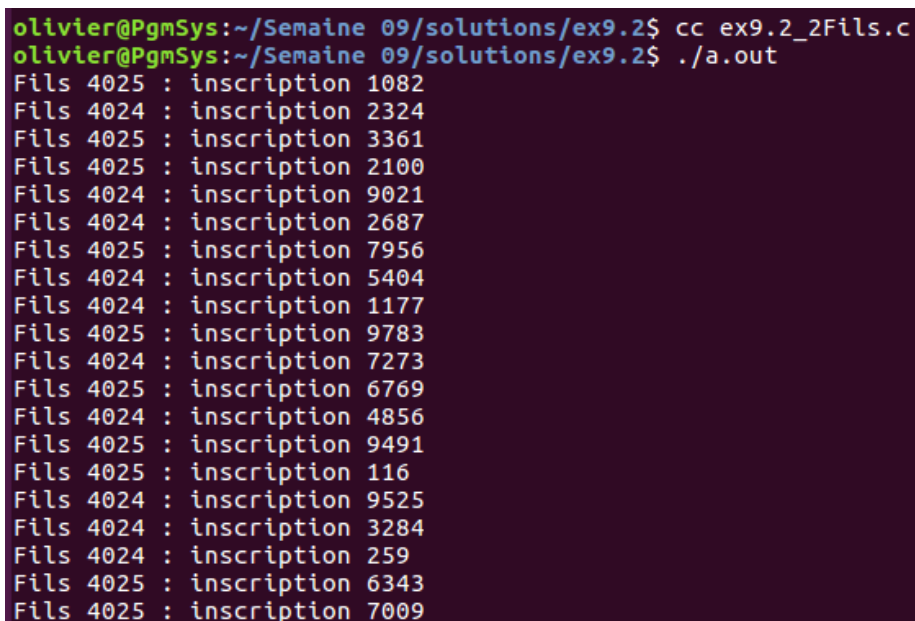
Écrivez un programme de simulation de traitement de demandes d'inscriptions. Des demandes d'inscriptions seront envoyées au père via des pipes aléatoirement par 2 processus fils. Chaque fils enverra 10 demandes d'inscriptions.

De manière plus précise, les processus fils s'endormiront pour un laps de temps aléatoire(`random_number_time`) compris entre 0 microsecondes et 10000 microsecondes et enverront ensuite un message d'inscription de la forme « Fils <pid> : inscription <random_number_time> ». Le processus père réceptionnera les messages de ses fils.

Quelques remarques :

1. Afin de s'assurer du caractère aléatoire dans les processus fils, initialiser le générateur de nombres aléatoires dans chacun des processus fils par le pid du processus :
`srand(getpid()) ;`
2. La fonction `usleep` permet d'endormir un processus pour un certain nombre de microsecondes.
3. Le processus parent sera arrêté avec CTRL-C.

Résultat attendu :



```
olivier@PgmSys:~/Semaine 09/solutions/ex9.2$ cc ex9.2_2Fils.c
olivier@PgmSys:~/Semaine 09/solutions/ex9.2$ ./a.out
Fils 4025 : inscription 1082
Fils 4024 : inscription 2324
Fils 4025 : inscription 3361
Fils 4025 : inscription 2100
Fils 4024 : inscription 9021
Fils 4024 : inscription 2687
Fils 4025 : inscription 7956
Fils 4024 : inscription 5404
Fils 4024 : inscription 1177
Fils 4025 : inscription 9783
Fils 4024 : inscription 7273
Fils 4025 : inscription 6769
Fils 4024 : inscription 4856
Fils 4025 : inscription 9491
Fils 4025 : inscription 116
Fils 4024 : inscription 9525
Fils 4024 : inscription 3284
Fils 4024 : inscription 259
Fils 4025 : inscription 6343
Fils 4025 : inscription 7009
```

Généraliser le programme pour que le nombre de fils soit configurable. Exemple avec 4 fils :

```
olivier@PgmSys:~/Semaine 09/solutions/ex9.2$ cc ex9.2_MultiFils.c
olivier@PgmSys:~/Semaine 09/solutions/ex9.2$ ./a.out
Fils 4294 : inscription 4330
Fils 4293 : inscription 4897
Fils 4294 : inscription 90
Fils 4292 : inscription 9381
Fils 4295 : inscription 9240
Fils 4294 : inscription 5840
Fils 4293 : inscription 9923
Fils 4294 : inscription 3158
Fils 4295 : inscription 5248
Fils 4295 : inscription 358
Fils 4292 : inscription 7371
Fils 4294 : inscription 7642
Fils 4293 : inscription 8107
Fils 4294 : inscription 597
Fils 4295 : inscription 8853
Fils 4294 : inscription 1711
Fils 4292 : inscription 9483
Fils 4295 : inscription 5794
Fils 4293 : inscription 9300
Fils 4292 : inscription 7728
Fils 4294 : inscription 8995
Fils 4293 : inscription 2588
Fils 4294 : inscription 2334
Fils 4295 : inscription 8615
```