TD 4 : Files de messages

Le cours sur les files de messages est celui de la semaine 5 (cours5 2021) sur Moodle, consultez-le !

Exercice 1. File de message

Écrire deux programmes, un serveur et un client :

- Le serveur (**serveur_randint.c**) envoie chaque seconde dans une file de message un message contenant un nombre aléatoire sous forme d'un entier (int). Ce nombre est généré par la commande `rand()`, et il est affiché avant de l'envoyer sur la file de message.
- Le client (**client_randint.c**) lit les messages et affiche les nombres aléatoires contenus dans ces messages.

Le nom de la file de message (obligatoirement sous la forme **/nom_de_la_file_de_message**) est passé sur la ligne de commande (donc lu en utilisant `argc` et `argv` en *C*).

Vous lancerez les deux programmes dans deux terminaux différents. Testez en lançant plusieurs clients.

Remarques :

- On peut voir les files de messages par **ls -l /dev/mqueue**
- On peut les manipuler avec les commandes classiques du shell (rm notamment).
- Options de compilation : **-lrt**

Fonctions utiles :

---

includes

```
#include <fcntl.h> /*For O_* constants */
#include <sys/stat.h> /*For mode constants */
```

```
#include <mqueue.h>
```

fonctions

```
int mq_send(mqd_t mqdes, const char *msg_ptr,size_t msg_len, unsigned int msg_prio);

int mq_getattr(mqd_t mqdes, struct mq_attr *attr);

mqd_t mq_open(const char *name, int oflag, mode_t mode,struct mq_attr *attr);

ssize_t mq_receive(mqd_t mqdes, char *msg_ptr,

size_t msg_len, unsigned int *msg_prio);
```

Exercice 2. File de message 2

Modifiez les deux programmes précédents pour que le serveur n'émette qu'un nombre limité de nombres aléatoires (par exemple 10) puis se termine.

Le client lit les nombres mais si au bout de 3 secondes il n'a pas reçu de message il se termine.

Fonctions utiles :

includes	fonctions
`#include <unistd.h>`	`unsigned alarm(unsigned seconds);`
`#include <mqueue.h>`	`int mq_close(mqd_t mqdes);`
`#include <mqueue.h>`	`int mq_unlink(const char *name);`

La fonction `alarm` déclenche un signal SIGALRM au bout de « seconds » secondes.

Remarque :

En quittant le programme, n'oubliez pas de faire appel aux fonctions `mq_close` et `mq_unlink` pour supprimer la file de messages.

Exercice 3. File de message 3

Ecrivez deux programmes : un client et un serveur.

- Le serveur (**serveur_mdp.c**) doit créer des mots de passe de longueurs variables et les communiquer à des clients (**client_mdp.c**).

Pour cela, le serveur crée une file de message « **/serveur_mdp** », qui permet de recevoir le nom de la file de message qui permettra de communiquer le mot de passe au client.

Il y a donc une file de message pour le serveur et *une file de message pour chaque client*. Ces files de messages des clients se nomment « **/client_mdp_xxxx** », où xxxx est le pid du client.

La génération de mot de passe utilisera la fonction suivante :

```
1  void rand_str(char *dest, size_t length) {
2
3  char charset[] = "0123456789"
4                   "abcdefghijklmnopqrstuvwxyz"
5                   "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
6
7  while (length-- > 0) {
8      size_t index = (double) rand() / RAND\_MAX * (sizeof(charset) - 1);
9      *dest++ = charset[index];
10 }
11
12 *dest = '\0'; // fin de chaine de caracteres
13
14 }
```

La longueur du mot de passe sera un nombre aléatoire entre *8 et 20*, elle sera générée par :

`random()%13+8`

Lors de son lancement, le client se connecte au serveur, lui envoie le nom de sa file de message (« **/client_mdp_xxxx** ») puis se met en attente de l'appui sur une touche clavier (par la fonction `fgets`). Quand l'appui a lieu le client lit un mot de passe, l'affiche puis recommence l'attente. Lorsque le client se termine par un appui sur **ctrl+d**, il termine en supprimant sa file de message.