Communication par files de messages (IPC System V)

EXERCICE 1 :

On vous demande de programmer un serveur de calcul utilisant les files de messages. Le serveur exécute des opérations simples (+, -, /, *). Il fonctionne de la façon suivante :

- Les clients déposent leurs requêtes (opération, opérandes et pid) dans un message dont le type est 1.
- Le serveur effectue l'opération demandée (+, -, /, *) et écrit le résultat dans un message dont le type est le pid du client.
- Le client, qui s'était bloqué en attente sur un message dont le type est son propre pid peut alors lire le résultat. Pour construire la partie texte des messages envoyés par le client on utilisera une structure de données qui contient :
  - Le type du message (long) ;
  - Une autre structure contenant le pid du client, l'opération à effectuer et les opérandes.
- Le serveur répond en envoyant un message structuré de la même façon. Le résultat de l'opération est enregistré sur le premier opérande.

Proposez une structure de message pour cet exercice (dans « `calcul.h` »), et complétez les deux fichiers fournis « `serveur.c` » et « `client.c` » (voir Annexe).

EXERCICE 2 :

Un processus client envoie au moyen d'une file de messages à un processus serveur un texte passé en ligne de commande. Le serveur « traduit » ce message (traduction : minuscules -> MAJUSCULES) et renvoie au client le message traduit par *une autre* file de messages. Le client affiche le message traduit et s'arrête. Le serveur s'arrête à la réception d'un message « @ ».

ANNEXE

Fichier `calcul.h`

```
#ifndef __CALCUL_H__
#define __CALCUL_H__

struct msg_struct {
    long type;
    /* A COMPLETER */
};

#endif /* __CALCUL_H__ */
```

Fichier `serveur.c`

```
#include "calcul.h"

#include <stdlib.h>
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/msg.h>


void raz_msg(int signal) {
     printf("Suppression de la file de message!\n");
     msgctl(/* A COMPLETER */);
}

int msg_id;

int main(int argc, char const *argv[])
{
     struct msg_struct msg;
     int i_sig;
     int result;

     /* liberer la zone de messages sur reception de n'importe
     quel signal */

     for (i_sig = 0 ; i_sig < NSIG ; i_sig++) {
     /* A COMPLETER */
     }
```

```
     msg_id = msgget(/* A COMPLETER */);
     printf("SERVEUR: pret!\n");
     while (1 == 1) {
          /* reception */
          msgrcv(/* A COMPLETER */);
          printf("SERVEUR: reception d'une requete de la part
                  de: %d\n",
          /* A COMPLETER */);
          /* preparation de la reponse */
          /* A COMPLETER */
          /* envoi de la reponse */
          msgsnd(/* A COMPLETER */);
     }
     return EXIT_SUCCESS;
}
```

Fichier `client.c`

```
#include "calcul.h"

#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/msg.h>


int main(int argc, char const *argv[])
{
     int msg_id;
     struct msg_struct msg;

     if (argc != 4) {
          printf("Usage: %s operande1 {+|-|*|/} operande2\n",
argv[0]);
          return EXIT_FAILURE;
     }

     /* il faut eviter la division par zero */
     /* A COMPLETER */
     /* ATTENTION : la file de messages doit avoir ete creee par
     le serveur. Il faudrait tester la valeur de retour (msg_id)
     pour verifier que cette creation a bien eu lieu. */

     msg_id = msgget(/* A COMPLETER */);
     printf("CLIENT %d: preparation du message contenant
            l'operation suivante:%d %c %d\n", getpid(),
            atoi(argv[1]), argv[2][0], atoi(argv[3]));
```

```
    /* On prepare un message de type 1 à envoyer au serveur
     avec les informations necessaires */
    /* A COMPLETER */
    /* envoi du message */

    msgsnd(/* A COMPLETER */);

    /* reception de la reponse */

    msgrcv(/* A COMPLETER */);
    printf("CLIENT: resultat recu depuis le serveur %d : %d\n",

    /* A COMPLETER */);
    return EXIT_SUCCESS;
}
```