SONMEZTURK Emre
LUQUET Quentin
RIOUFOL Simon

CR TP4 : OS302

Exercice 1 :

Code : serveur.c

```c
#include "calcul.h"
#include <stdlib.h>
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/msg.h>
int msg_id;
void raz_msg(int signal) {
    printf("Suppression de la file de message!\n");
    msgctl(msg_id,IPC_RMID,NULL);
    exit(EXIT_SUCCESS);
}

int main(int argc, char const *argv[])
{
    struct msg_struct msg;
    int i_sig;
    int result;
    key_t cle = ftok("serveur.c", 0);


    /* liberer la zone de messages sur reception de n'importe quel signal */
    for (i_sig = 0 ; i_sig < NSIG ; i_sig++) {
        signal(i_sig, raz_msg);
    }

    msg_id = msgget(cle, IPC_CREAT | 0666);
    if(msg_id == -1)
        perror("MSGET "), exit(1);

    printf("SERVEUR: pret!\n");
    while (1 == 1) {
        /* reception */
        if(msgrcv(msg_id, &msg, sizeof(struct infos),1L,0) <= 0) //message vide ou erreur
```

```
            fprintf(stderr,"Error msgrcv (ligne : %d).",__LINE__), exit(1);

        printf("SERVEUR: reception d'une requete de la part de: %d\n",msg.pid);
        /* preparation de la reponse */
        switch(msg.operation)
        {
            case '+' :
                result = msg.ope1 + msg.ope2;
                break;
            case '-':
                result = msg.ope1 - msg.ope2;
                break;
            case 'x' :
                result = msg.ope1 * msg.ope2;
                break;
            case '/':
                result = msg.ope1 / msg.ope2;
                break;
            default :
                fprintf(stderr, "Erreur opérateur non valide {%c}.\n",msg.operation);
                exit(1);
        }
        /* envoi de la reponse */
        printf("%d %c %d = %d\n", msg.ope1, msg.operation, msg.ope2, result);
        msg.type = msg.pid;
        msg.pid = getpid();
        msg.ope1 = result;
        if(msgsnd(msg_id, &msg, sizeof(struct infos),0) == -1)
            perror("msgsnd"),exit(1);
    }
    return EXIT_SUCCESS;
}
```

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[])
```

```c
{
    int msg_id;
    struct msg_struct msg;

    if (argc != 4) {
        printf("Usage: %s opereande1 {+|-|x|/} operande2. \n",argv[0]);
        return EXIT_FAILURE;
    }

    /* il faut eviter la division par zero */
    if(argv[2][0] == '/' && argv[3]== 0)
        fprintf(stderr, "Erreur division par 0.\n"), exit(1);

    key_t cle = ftok("serveur.c",0);
    /* 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(cle,0);
    if(msg_id == -1)
        perror("MSGET (creation) "), exit(1);
    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 */
    msg.type = 1;
    msg.pid = getpid();
    msg.operation = argv[2][0];
    msg.ope1 = atoi(argv[1]);
    msg.ope2 = atoi(argv[3]);
    /* envoi du message */
    if(msgsnd(msg_id, &msg, sizeof(struct infos),0)== -1)
        perror("msgsnd"),exit(1);
    printf("On attend la reponse.\n");
    /* reception de la reponse */
    if( msgrcv(msg_id, &msg, sizeof(struct infos), getpid(), 0) == -1)
        perror("msgrcv"), exit(1);
    printf("CLIENT: resultat recu depuis le serveur %d : %d\n", msg.pid,msg.ope1);
    return EXIT_SUCCESS;
}
```

calcul.h :

```c
#ifndef __CALCUL_H__
#define __CALCUL_H__

#include <unistd.h>
#include <sys/types.h>

struct msg_struct {
    long type;
    pid_t pid;
    char operation;
    int ope1;
    int ope2;
};

struct infos{
    pid_t pid;
    char operation;
    int ope1;
    int ope2;
};

#endif /* __CALCUL_H__ */
```

Sortie Terminal :

```
cryoxia@DESKTOP-MP488LU:~/OS312/TP4$ make
gcc -o serveur serveur.c
gcc -o client client.c
cryoxia@DESKTOP-MP488LU:~/OS312/TP4$ ./serveur &
[1] 3583
cryoxia@DESKTOP-MP488LU:~/OS312/TP4$ SERVEUR: pret!

cryoxia@DESKTOP-MP488LU:~/OS312/TP4$ ./client 3 + 3
CLIENT 3584: preparation du message contenant l'operation suivante:
On attend la reponse.
SERVEUR: reception d'une requete de la part de: 3584
3 + 3 = 6
CLIENT: resultat recu depuis le serveur 3583 : 6
```

Exercice 2 :

Code : client.c

```c
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <string.h>


int main(int argc, char const *argv[])
{
        int msg_id1; /*request*/
        int msg_id2; //reponses
        struct msg_struct msg;
        key_t key1; //clé des requetes
        key_t key2;
        if((key1 = ftok("serveur.c",0))==-1) { //file des requetes
                perror("prb ftok");
                exit(0);
        }
        if((key2 = ftok("client.c",0))==-1) { //file des reponses
                perror("prb ftok");
                exit(0);
        }

        if((msg_id1 = msgget(key1,0))==-1) {
                perror("prb msget");
                exit(0);
        }
        if((msg_id2 = msgget(key2,0))==-1) {
                perror("prb msget");
                exit(0);
        }
        printf( "Veuillez saisir la phrase : " );
        scanf( "%[^\n]", msg.message.buffeur);
        /* On prepare un message de type 1 à envoyer au serveur avec les
         * informations necessaires */
        /* A COMPLETER */
        msg.type = 1;
        msg.message.pid = getpid();
        /* envoi du message */

        if(msgsnd(msg_id1,&msg,sizeof(struct message),0)==-1) {
                perror("prb msgsnd");
                exit(0);
        }

        /* reception de la reponse */

        if(msgrcv(msg_id2,&msg,sizeof(struct message),getpid(),0)==-1) {
                perror("prb msgrcv");
                exit(0);
        }
        printf("CLIENT: resultat recu depuis le serveur %d : %s\n",getpid(),msg.message.buffeur);
        return EXIT_SUCCESS;
}
```

serveur.c

```c
#include "calcul.h"
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <ctype.h>

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

int msg_id1;
int msg_id2;

int main(int argc, char const *argv[])
{
        int  end = 0;
        struct msg_struct msg;
//      int i_sig;
        /* liberer la zone de messages sur reception de n'importe quel signal */
/*
        for (i_sig = 0 ; i_sig < NSIG ; i_sig++) {
        }*/
        key_t key1; //file des requetes
        key_t key2;
        if((key1=ftok("serveur.c",0))==-1) {
                perror("prb ftok serveur");
                exit(0);
        }
        if((key2=ftok("client.c",0))==-1) {
                perror("prb ftok serveur");
                exit(0);
        }

        if((msg_id1 = msgget(key1,IPC_CREAT | 0660))==-1) {
                perror("prb msget serveur");
                exit(0);
        }
        if((msg_id2 = msgget(key2,IPC_CREAT | 0660))==-1) {
                perror("prb msget serveur");
                exit(0);
        }

        printf("SERVEUR: pret!\n");
    while (end!=1) {
          /* reception */
          if((msgrcv(msg_id1,&msg,sizeof(struct message),1L,0))==-1) {
                  perror("prb recieve serveur");
                  exit(0);
          }
          if(strcmp(msg.message.buffeur,"@")==0){
                  end = 1;
                  strcpy(msg.message.buffeur,"fin de transmission");
          }
          else {
                  printf("SERVEUR: reception d'une requete de la part de: %d\n",msg.message.pid);
                  for(int i = 0;i<strlen(msg.message.buffeur);i++){
                          if (islower(msg.message.buffeur[i])){
                          msg.message.buffeur[i] = toupper(msg.message.buffeur[i]);
                          }
                  }
          }
                  msg.type = msg.message.pid;
                  if(msgsnd(msg_id2,&msg,sizeof(struct message),0)==-1) {
                          perror("prb send client");
                          exit(0);
                  }
    }
    return EXIT_SUCCESS;
```

calcul.h :

```c
#ifndef __CALCUL_H__
#define __CALCUL_H__
#include <stdlib.h>
#include <sys/types.h>


struct message {
        pid_t pid;
        char buffeur[20];
};

struct msg_struct {
    long type;
    struct message message;
};

#endif /* __CALCUL_H__ */
```

Sortie Terminal :

Coté client

```
bureau@bureau-G3-3500:~/Téléchargements/tpcs$ ./client
Veuillez saisir la phrase : bonjour test phrase
CLIENT: resultat recu depuis le serveur 6521 : BONJOUR TEST PHRASE
bureau@bureau-G3-3500:~/Téléchargements/tpcs$ ./client
Veuillez saisir la phrase : @
CLIENT: resultat recu depuis le serveur 6527 : fin de transmission
bureau@bureau-G3-3500:~/Téléchargements/tpcs$
```

Coté serveur

```
bureau@bureau-G3-3500:~/Téléchargements/tpcs$ ./serveur
SERVEUR: pret!
SERVEUR: reception d'une requete de la part de: 6515
SERVEUR: reception d'une requete de la part de: 6521
fin de transmission
bureau@bureau-G3-3500:~/Téléchargements/tpcs$
```