Communication simple

A) Enoncé

Il s'agit de simuler l'envoi de 3 messages par un processus P0 et la réception par trois processus fils de P0: P1, P2, P3 chacun spécialisé pour un type de messages.
un message de type 1 est un couple (nom, prénom)
un message de type 2 est un couple (nom, prénom, âge)
un message de type 3 est un couple (nom, prénom, âge, adresse)

Le processus réceptionne les messages par l'entrée standard et les envoie dans la file de messages. Il constate la fin d'envoi des messages d'un type donné lorsqu'il reçoit pour un message de ce type des chaînes vides pour le nom et le prénom.

programme.h

```
#include<stdio.h>
#include<errno.h>
#include<wait.h>
#include<stdlib.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/msg.h>
#include<sys/stat.h>

void PosterMsg (int id);
void TraiterMsg (long type, int id);

typedef struct
{
        long type;
        char Nom[15];
        char Prenom[15];
}SMsg1;

typedef struct
{
          long type;
          char Nom[15];
          char Prenom[15];
          char Age[3];
}SMsg2;

typedef struct
{
        long type;
        char Nom[15];
        char Prenom[15];
        char Age[3];
        char Adresse[15];
}SMsg3;

void erreur (const char *);
void PosterMsg (int id);
void TraiterMsg (long type, int id);
```

programme.c

```
#include "prog1.h"
/*************************************************************************/
 /* La fonction TraiterMsg extrait un message d'un type donne (parametre 1
de la fonction) de la file d'identifiant id (parametre 2), et affiche
ce message tant que les chaines pour le nom ou le prenom n'est pas vide*/
/*************************************************************************/

void TraiterMsg (long type, int id)
{
  SMsg1 Msg1;
  SMsg2 Msg2;
  SMsg3 Msg3;

  switch (type)
  {
            case 1l:
            msgrcv(id, &Msg1, sizeof(SMsg1)-sizeof(long), 1l, 0);
            while(strcmp(Msg1.Nom, "")!=0 || strcmp (Msg1.Prenom, "") !=0)
            {
                        printf("Type 1 extrait : %s %s \n", Msg1.Nom, Msg1.Prenom);
                        msgrcv(id, &Msg1, sizeof(SMsg1)-sizeof(long), 1l, 0);
            }
            break;
            case 2l:
            msgrcv(id, &Msg2, sizeof(SMsg2)-sizeof(long), 2l, 0);
            while(strcmp(Msg2.Nom, "")!=0 || strcmp (Msg2.Prenom, "") !=0)
            {
                        printf("Type 2 extrait: %s %s %s\n", Msg2.Nom, Msg2.Prenom,Msg2.Age);
                        msgrcv(id, &Msg2, sizeof(SMsg2)-sizeof(long), 2l, 0);
            }
            break;
            case 3l:
            msgrcv(id, &Msg3, sizeof(SMsg3)-sizeof(long), 3l, 0);
            while(strcmp(Msg3.Nom, "")!=0 || strcmp (Msg3.Prenom, "") !=0)
            {
                        printf("Type 3 extrait: %s %s %s %s\n", Msg3.Nom, Msg3.Prenom,Msg3.Age,Msg3.Adresse);
                        msgrcv(id, &Msg3, sizeof(SMsg3)-sizeof(long), 3l, 0);
            }
            break;
   }
}
```

```
/*************************************************************************/
/* La fonction PosterMsg(int id) récupère de l'entrée standard les messages
qu'elle place dans la file de messages connaissant son identifiant
id (paramètre de la fonction)*/
/* Cette fonction envoie les messages d'un type donné dans la file
jusqu'a ce qu'elle récupère de l'entree standard pour ce type de message
des chaînes vides pour le nom et le prénom */
/*************************************************************************/

void PosterMsg(int id)
{
    int type;
    SMsg1 Msg1 = {1l, "", ""};
    SMsg2 Msg2 = {2l, "", "",""};
    SMsg3 Msg3 = {3l, "", "","","",""};
    int M1 = 1, M2 = 1, M3 = 1;

    while (M1 != 0 || M2 != 0 || M3 != 0)
    {
            printf("M1= %d M2= %d M3= %d\n", M1, M2, M3);
            printf("donnez le type de message(1,2,3) \n");
            fflush(stdin);
            scanf("%d", &type);
            switch(type)
            {
                case 1:
                        printf("Message de type Msg1\n");
                        printf("---------------------\n");
                        printf("Nom: \t\t");
                        fflush(stdin);
                        gets(Msg1.Nom);
                        printf("Prenom: \t");
                        fflush(stdin);
                        gets(Msg1.Prenom);
                        msgsnd(id, &Msg1, sizeof(SMsg1) - sizeof(long), 0);
                        if (strcmp(Msg1.Nom, "") ==  0 && strcmp(Msg1.Prenom, "") == 0)
                        M1 = 0;
                        break;
                 case 2:
                        printf("Message de type Msg2\n");
                        printf("---------------------\n");
                        printf("Nom: \t\t");
                        fflush(stdin);
                        gets(Msg2.Nom);
                        printf("Prenom: \t");
                        fflush(stdin);
                        gets(Msg2.Prenom);
                        printf("Age: \t\t");
                        fflush(stdin);
                        gets(Msg2.Age);
                        msgsnd(id, &Msg2, sizeof(SMsg2) - sizeof(long), 0);
                        if (strcmp(Msg2.Nom, "") ==  0 && strcmp(Msg2.Prenom, "") == 0)
                        M2 = 0;
                        break;
                 case 3:
                        printf("Message de type Msg2\n");
                        printf("---------------------\n");
                        printf("Nom: \t\t");
                        fflush(stdin);
                        gets(Msg3.Nom);
                        printf("Prenom: \t");
                        fflush(stdin);
```

```
                        gets(Msg2.Prenom);
                        printf("Age: \t\t");
                        fflush(stdin);
                        gets(Msg2.Age);
                        printf("Adresse: \t");
                        fflush(stdin);
                        gets(Msg3.Adresse);
                        msgsnd(id, &Msg3, sizeof(SMsg3) - sizeof(long), 0);
                        if (strcmp(Msg3.Nom, "") ==  0 && strcmp(Msg3.Prenom, "") == 0)
                        M3 = 0;
                        break;
                }
        }
}
```

```
/*************************************************************************/
/* Dans le main le processus pere cree une file de messages, puis envoie
des messages de 3 types differents dans cette file, puis cree 3 processus
fils chacun se charge de traiter un type de message qu'il aura extrait de
la file*/
/*************************************************************************/

void main()
{
key_t cle;
int IdMsg;
pid_t pid;
int statut;

cle=ftok("msg.c",0);
IdMsg=msgget(cle,IPC_CREAT|IPC_EXCL| S_IRUSR|S_IWUSR);
if(IdMsg==-1) erreur("Erreur msgget");
printf("cle %d Id %d\n",cle,IdMsg);

pid = fork();
if (pid == 0)
                {
                TraiterMsg(1l,IdMsg);
                printf("Fils 1 termine\n");
                exit(0);
                }
pid = fork();
if (pid == 0)
                {
                TraiterMsg(2l,IdMsg);
                printf("Fils 2 termine\n");
                exit(0);
                }
pid = fork();
if (pid == 0)
                {
                TraiterMsg(3l,IdMsg);
                printf("Fils 3 termine\n");
                exit(0);
                }

PosterMsg(IdMsg);

wait(&statut);
wait(&statut);
wait(&statut);

// appel pour liberer la file de message
msgctl(IdMsg, IPC_RMID, NULL);
}

void erreur(const char * message)
{
perror(message);
exit(-1);
}
```