

## 1. GENERALITES

Il s'agit d'une implémentation du concept de boite à lettres. Le but est d'échanger ,de façon asynchrone, des données stockées dans le noyau du système, sous forme de paquets d'octets identifiables. C'est un mode de communication opposé à celui des tubes parce qu'une opération de lecture ne peut extraire que le produit d'une opération d'écriture.

Avec une seule file de message, on peut envoyer des paquets à plusieurs processus (notion de multiplexage).

La structure d'un message est donnée dans `/usr/include/sys/msg.h` par :

```
struct msg_buf
{
    long mtype;      /* type du message */
    char mtext [1]; /* contenu du message */
};
```

Elle est inutilisable en l'état (taille 1 !!!). La taille maximale d'un texte de message est définie à la configuration du système.

Les messages sont rangés dans une file de messages, zone de mémoire centrale gérée par le système. Le processus qui a créé une file de messages en est le propriétaire.

Une file de messages est associée à une clé (l'équivalent numérique d'un nom) utilisée pour obtenir l'identificateur de la file de messages **msgid**, fourni par UNIX au processus qui donne la clé.

Tout processus qui a connaissance de l'existence de la file de messages et qui possède des droits d'accès peut réaliser des opérations sur la file :

- création d'une file,
- consultation des caractéristiques d'une file :
  - \* nombre de messages dans la file,
  - \* taille de la file,
  - \* pid du dernier processus ayant écrit dans la file,
  - \* pid du dernier processus ayant lu (extrait) dans la file,
- envoi d'un message dans la file,
- lecture (extraction) d'un message, en liaison avec son type.

Le fichier `/usr/include/sys/msg.h` contient la définition de la structure **msgid\_ds** consultable par la fonction système **ipcs**, et de diverses constantes.

## 2. CREATION D'UNE FILE DE MESSAGES

La fonction prototypée par :

### **int msgget (key\_t cle, int option)**

retourne l'indicateur de la file (ou -1 en cas d'erreur).

**key\_t** : type défini dans /usr/include/sys/ipc.h, équivalent à long

**option** : combinaison, avec l'opérateur | de droits d'accès et de constantes définies dans /usr/include/sys/msg.h et /usr/include/sys/ipc.h :

MSG\_R (0400), MSG\_W (0200): permission en lecture, en écriture  
IPC\_CREAT (0001000) : création d'une file de messages  
IPC\_EXCL (0002000) : échec si la file existe déjà

**cle**: si **cle** = **IPC\_PRIVATE**, une nouvelle file est créée

sinon si **cle** ne correspond pas à une file existante  
si **IPC\_CREAT** n'est pas positionné : erreur, retour de -1  
sinon, une nouvelle file est créée associée à cette clé avec les droits d'accès spécifiés. Le propriétaire effectif est le propriétaire effectif du processus. Le groupe propriétaire et créateur est le groupe propriétaire effectif du processus.

sinon si **IPC\_CREAT** et **IPC\_EXCL** sont positionnés : erreur, retour de -1  
sinon, la fonction retourne l'identificateur de la file

La fonction prototypée par :

### **key\_t ftok (char \*nom, int id)**

dans /usr/include/sys/types.h calcule une clé unique dans tout le système à partir du fichier de chemin **nom** et de **id**.

Exemple : récupération de l'identificateur d'une file de messages  
num = msgget (ftok (CHEMIN, cle), 0)

Exemple : création d'une file de messages

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
main ()
{
    key_t cle;
    int flag, num, i;
    printf ("donnez la cle entière associée à la file à créer : ");
    scanf ("%ld", &cle);
    flag = MSG_W | MSG_R | IPC_CREAT;
    /* autre solution : flag = IPC_CREAT | IPC_EXCL | 0666 ; */
    if ((num = msgget (cle, flag)) == -1)
    /* autre solution : if (((num = msgget (ftok (CHEMIN, cle), flag)) == -1) */
    {
        fprintf (stderr, "création impossible\n");
        exit (1);
    }
    printf ("file créée avec l'identificateur %d\n", num);
}
```

### **3. CONSULTATION, MODIFICATION DES CARACTERISTIQUES D'UNE FILE DE MESSAGES.**

On utilise à cet effet la fonction prototypée par :

```
int msgctl (int msgid, int cmd, struct msqid_ds *buf)
```

retourne 0 (ou -1 en cas d'échec)

**msgid** : identificateur de la file

**cmd** : définit les opérations à effectuer, selon sa valeur :

IPC\_RMID : la file est supprimée

IPC\_SET : mise à jour de quelques valeurs de permission (champs de la structure `msg_perm` associée à `msgid` : `msg_perm.uid`, `msg_perm.gid` et `msg_perm.mode`)

IPC\_STAT : recopie dans `buf` les informations relatives à la file contenues dans l'objet de type `struct msqid_ds`, associé à la file.

Exemple :

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

main ()
{
    int num;
    struct msqid_ds buf;
    printf ("numéro de la file à étudier ? ");
    scanf ("%d", &num);
    if (msgctl (num, IPC_STAT, &buf) == -1)
        { fprintf (stderr, "consultation impossible\n");
          exit (1);
        }
    printf ("caractéristiques de la file : %d\n", num);
    printf ("nombre de messages : %d\n", buf.msg_qnum);
    printf ("droits d'accès : %o\n", buf.msg_perm.mode);
    if (msgctl (num, IPC_RMID, &buf) == -1)
        /* ou bien : if (msgctl (num, IPC_RMID, NULL ) == -1) */
        { fprintf (stderr, "suppression impossible\n");
          exit (2);
        }
    printf ("suppression de la file %d réussie\n", num);
}
```

Un exemple intéressant est donné dans A.B. FONTAINE et Ph. HAMMES, p. 359 à 361.

## 4. ENVOI DE MESSAGE A UNE FILE

La fonction prototypée par :

```
int msgsnd (int msgid, struct msgtxt *msgp, int msgz, int msgflg)
```

retourne 0 (ou -1 en cas d'erreur), avec :

**msgid** : identificateur de la file

La structure msgbuf (inutilisable, cf. p. 1) est remplacée par exemple par :

```
struct msgtxt
{
    long mtype;      /* type du message */
    char mtext [256]; /* contenu du message */
};
```

**msgz** : taille réelle du message

**msgflg** peut prendre les valeurs :

0 : blocage de la fonction si la file est pleine

IPC\_NOWAIT : si la file est pleine, l'appel à la fonction n'est pas bloquant

Exemple :

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#define TAILLE 100 /* taille maximale d'un message */
struct msgtxt
{
    long mtype;      /* type du message */
    char mtext [256]; /* contenu du message */
};

main ()
{
    int i, num;
    long t;
    char ch [TAILLE];
    struct msqid_ds buf;
    struct msgtxt message;
    printf ("numéro de la file à étudier ? ");
    scanf ("%d", &num);
    if (msgctl (num, IPC_STAT, &buf) == -1)
        { fprintf (stderr, "erreur\n");
          exit (1);
        }
    printf ("caractéristiques de la file : %d\n", num);
    printf ("nombre de messages avant écriture : %d\n", buf.msg_qnum);
    printf ("pid du dernier processus écrivain : %d\n", buf.msg_lspid);
    printf ("type du message : ");
    scanf ("%ld", &t);
    message.mtype = t;
    printf ("texte du message (%d car. maxi) : ", TAILLE-1);
    gets (ch);
    strcpy (message.mtext, ch);
```

```

        i = msgsnd (num, &message, strlen (ch), ~ IPC_NOWAIT);
        if (i != 0) fprintf (stderr, "message non envoyé\n");
        else {      printf ("numéro du processus %d\n", getpid ());
                  msgctl (num, IPC_STAT, &buf);
                  printf ("nombre de messages après écriture : %d\n", buf.msg_qnum);
                  printf ("pid du dernier processus écrivain : %d\n", buf.msg_lspid);
                }
    }
}

```

## **5 . EXTRACTION D'UN MESSAGE D'UNE FILE.**

La fonction prototypée par :

**int msgrev (int msgid, struct msgtxt \*buf, int msgz, long msgtyp, int msgflg)**

retourne la longueur du message extrait (-1 en cas d'erreur), avec :

**msgid** : identificateur de la file

la structure msgbuf (inutilisable, cf. p.1) est remplacée par :

```

struct msgtxt
{
    long mtype;          /* type du message      */
    char mtext [256]; /* contenu du message */
};

```

**msgz** : taille maximale d'un message recevable

**msgtyp** : code du type du message à extraire :

0 : le premier message dans la file, quelque soit son type  
 > 0 : le premier message de type msgtyp dans la file  
 < 0 : le premier message de type 0 ... |msgtyp|

**msgflg** précise l'action à réaliser si aucun message du type attendu n'est présent dans la file :

si IPC\_NOWAIT est positionné, on retourne -1 et l'appel est non bloquant  
 sinon, la fonction se bloque jusqu'à l'arrivée d'un message satisfaisant ou  
 jusqu'à réception d'un signal par le processus appelant ou jusqu'à disparition de la file.

Exemple :

```

#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#define TAILLE 100      /* taille maximale d'un message */
struct msgtxt
{
    long mtype;          /* type du message      */
    char mtext [256]; /* contenu du message */
};

main ()
{
    int i, num;
    long type;

```

```

struct msqid_ds buf;
struct msgtxt message;
printf ("numéro de la file à étudier ? ");
scanf ("%d", &num);
if (msgctl (num, IPC_STAT, &buf) == -1)
    { fprintf (stderr, "erreur\n");
      exit (1);
    }
printf ("caractéristiques de la file : %d\n", num);
printf ("nombre de messages avant lecture : %d\n", buf.msg_qnum);
printf ("pid du dernier processus lecteur : %d\n", buf.msg_lrpid);
printf ("type du message : ");
scanf ("%ld", &type);
i = msgrev (num, &message, TAILLE, type, ~ IPC_NOWAIT);
if (i == -1) fprintf (stderr, "pas de message\n");
else {
    printf ("pid du processus courant %d\n", getpid ());
    msgctl (num, IPC_STAT, &buf);
    printf ("nombre de messages après lecture : %d\n", buf.msg_qnum);
    printf ("pid du dernier processus lecteur : %d\n", buf.msg_lrpid);
}
}

```