Message Queue

Abdelkader Gouaïch

IUT de Montpellier

2014-2015

Objectif du cours :

- établir une communication entre processus
- échange de messages : message queue

- Les "messages queue" sont similaires aux tubes (pipe/fifo) mais il y a des différences :
- Le handle utilisé pour identifier une message queue est donné par msgget()
- La communication est orientée message : lecture d'un message entier à la fois
- Les messages possèdent un type (entier) : la réception des messages peut se faire en FIFO ou bien par type

Creation d'une message queue

```
#include <sys/types.h>
#include <sys/msg.h>
int msgget(key_t key, int msgflg);
```

key est une clé utilisée pour identifier la queue

msgflg : une liste d'options de création

Génération de la clé

Plusieurs méthodes pour créer la clé :

- Manuellement en mettant l'identifiant dans un fichier de configuration
- En utilisant la constante `IPC_PRIVATE` (génération d'une nouvelle clé automatiquement)
- En utilisant la fonction `ftok()` (file to key) qui prend un path et le transforme en un entier unique.

Flags d'ouverture

- `IPC_CREAT` : si aucune file avec la clé n'existe alors la file sera créée
- `IPC_EXCL` : si `IPC_CREAT` est spécifié et qu'une file existe déjà alors retourner une erreur `EEXIST`

opérations msgsnd() et msgrcv()

Deux fonctions pour échanger les messages msgsnd et msgrcv

- Le premier argument dans les deux fonctions est l'identifiant que la file de message
- Le second argument est un pointeur vers la structure de message

Structure de message

Le programmeur peut définir sa propre structure de message
Il est important que cette structure commence avec un `long` qui représente le type
Le contenu est libre même vide.
Exemple :

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

Envoyer un message

int msgsnd(**int** msqid, **const void** *msgp, size_t msgsz, **int** msgflg);

- msgid : identifiant de la file de message
- msgp : pointeur vers la structure du message
- msgsz : taille du message
- msgflg : flags de contrôle

Flags de contrôle

Un seul flag est définie : IPC_NOWAIT
Si la file de message est pleine alors l'envoi du message est bloquant jusqu'à la libération de l'espace.
Avec IPC_NOWAIT l'opération retourne immédiatement avec un code d'erreur EAGAIN.

Réception d'un message

ssize_t msgrcv(**int** msqid, **void** *msgp, size_t maxmsgsz, **long** msgtyp, **int** msgflg);

retirer un message de la file et le copier en mémoire (pointeur).

- msqid : id de la file
- msgp : pointeur de destination
- maxmsgsz : taille maximale du contenu du message. Si le message dépasse la taille une erreur E2BIG est signalée.
- msgtyp : le type de message
- msgflg : flags de contrôle

Les types des messages

La lecture des messages ne se fait pas nécessairement dans l'ordre des envois
Nous pouvons sélectionner les messages par type, c'est le but de l'argument `msgtyp`

- 0, alors le premier message de la file est retiré
- supérieur à 0, le premier message qui possède le même type est retiré
- inférieur à 0, la file est considérée comme une file de priorités. Le message qui possède un type inférieur ou égal à la valeur abs de msgtyp est retiré.

Exemple

(1) type : 300, contenu : ...
(2) type : 100, contenu : ...
(3) type : 200, contenu : ...
(4) type : 400, contenu : ...
(5) type : 100, contenu : ...

msgrcv(id, &msg, maxmsgsz, -300, 0);

Messages retirés sont : 2,5,3,1 et puis blocage ($400 > 300$).

Flags de contrôle

Le msgflg se construit avec des | entre 0 et les valeurs suivantes :

- `IPC_NOWAIT` : une réception non bloquante. Retourne `ENOMSG` si aucun message (avec le bon type) n'est présent
- `MSG_EXCEPT` : utilisable si le type est supérieur à 0 ; dans ce cas on retourne toutes les valeurs différentes de types (à l'exception de ...)
- `MSG_NOERROR` : par défaut si la taille du message dépasse la taille max alors il y a une erreur. Si ce flag est mis, le message est retiré sans erreur mais tronqué à la taille max.

Opérations de contrôle

int msgctl(**int** msqid, **int** cmd, **struct** msqid_ds *buf);

cmd peut prendre les valeurs suivantes :

- `IPC_RMID` efface la file de message.
- `IPC_STAT` copie la structure msqid_ds dans le pointeur buf
- `IPC_SET` mise à jour des champs de la structure msqid_ds à partir du pointeur buf.

Structure de données associée à la file

Chaque file de messages a une structure de données associée.

```
struct msqid_ds {
    struct ipc_perm msg_perm; /* Ownership and permissions */
    time_t msg_stime; /* Time of last msgsnd() */
    time_t msg_rtime; /* Time of last msgrcv() */
    time_t msg_ctime; /* Time of last change */
    unsigned long __msg_cbytes; /* Number of bytes in queue */
    msgqnum_t msg_qnum; /* Number of messages in queue */
    msglen_t  msg_qbytes; /* Maximum bytes in queue */
    pid_t msg_lspid; /* PID of last msgsnd() */
   pid_t msg_lrpid; /* PID of last msgrcv() */
}
```