

Laboratorio di Sistemi Operativi

primavera 2009

System V IPC code di messaggi

System V IPC

I tre tipi di IPC di System V sono:

- ▶▶ Code di Messaggi
- ▶▶ Semafori
- ▶▶ Memoria Condivisa

Identificatori

- ▶▶ Ogni struttura IPC è individuata tramite un **identificatore** (intero > 0) che viene assegnato dal kernel quando la struttura viene creata.
 - ▶ Es. quando si deve mandare un messaggio in una coda di messaggi bisogna conoscere l'identificatore della coda
- ▶▶ Se una struttura è creata e poi rimossa il suo identificatore non viene riutilizzato (come per pid)

Chiavi

- ▶▶ Quando si crea una struttura IPC bisogna specificare una **chiave** (di tipo *key_t*) che sarà convertita in **identificatore** dal kernel.

Criteri di scelta della chiave

1. Utilizzare la chiave speciale **IPC_PRIVATE** che garantisce la creazione di una struttura nuova; l'**identificatore** ottenuto può essere passato tra i vari processi che utilizzeranno la struttura
2. Fissare a priori tra client e server una chiave (posta ad esempio in un header file) che sarà utilizzata dal server per creare la struttura
3. Fissare a priori tra client e server un **pathname** ed un **ID** ($0 \leq \text{char} \leq 255$) ed utilizzare la funzione **ftok** per convertire questi valori in una chiave

Laboratorio di Sistemi Operativi

Vantaggi e Svantaggi

- ▶▶ **IPC_PRIVATE** garantisce che la struttura creata sia nuova ma ha lo svantaggio di dover passare l'identificatore tra server e client spesso tramite file (comunicazione tramite file system!)
- ▶▶ Definire una **chiave** a priori (oppure una **path** ed un **ID**) è più veloce ma non garantisce che la **chiave** non sia già stata utilizzata per altre strutture.

Laboratorio di Sistemi Operativi

Creazione/Riferimento

- ▶▶ per creare/riferirsi ad una struttura IPC

- ▶ **msgget**
- ▶ **semget**
- ▶ **shmget**

- ▶▶ hanno 2 argomenti in comune

- ▶ **key_t** *key*
- ▶ **int** *flag*

Laboratorio di Sistemi Operativi

chiave

- ▶▶ è **IPC_PRIVATE** se si vuole creare una struttura nuova
- ▶▶ una chiave (ad esempio scambiata tramite un header file) che non è associata a nessuna altra struttura esistente
 - ▶ di solito è un long integer (spesso unsigned)
- ▶▶ se ci si riferisce ad una struttura esistente bisogna usare la chiave usata a suo tempo dal server quando la creò
 - ▶ tranne che se fu creata con **IPC_PRIVATE**; in tal caso (si bypassano le funzioni **get**) e si usa l'**identificatore**, generato a suo tempo, nelle chiamate ad IPC tipo **msgsnd**, **msgrcv**.

Laboratorio di Sistemi Operativi

flag

- ▶▶ contengono i permessi di uso della struttura (tranne quelli di esecuzione, che sono ignorati)
 - ▶ per esempio: 660, 420, etc...
- ▶▶ se si vuole creare una struttura si fa l'**OR** con il bit **IPC_CREAT**
- ▶▶ se si vuole essere sicuri che non esiste già la struttura si fa l'**OR** anche con il bit **IPC_EXCL**
 - ▶ analogo alla open

Record ipc_perm

- ▶▶ quando è creata una struttura IPC viene associata ad essa un record **ipc_perm** che definisce i permessi, il proprietario, il gruppo, la chiave, etc...
- ▶▶ tali dati sono inizializzati alla creazione
- ▶▶ alcuni di questi possono essere modificati solo dal creatore o da root con le funzioni di controllo
 - ▶ **msgctl**
 - ▶ **semctl**
 - ▶ **shmctl** } analoghi di *chmod*, *chown*

IPC vs pipe/fifo

Le strutture IPC sono systemwide

- ▶▶ una coda di messaggi non è cancellata quando termina il processo che l'ha creata
 - ▶ la si può cancellare con **ipcrm**
- ▶▶ una pipe è rimossa automaticamente.
- ▶▶ una fifo resta nel sistema, ma è svuotata quando l'ultimo processo che vi si riferisce, termina.

Code di messaggi

- ▶▶ sono liste linkate di messaggi, create con **msgget**
- ▶▶ ci si riferisce tramite l'identificatore della coda
- ▶▶ nuovi messaggi sono inclusi alla fine con **msgsnd**
- ▶▶ prelevati dalla coda con **msgrcv** (non necessariamente con ordine fifo, ma in base al loro **tipo**)
- ▶▶ ogni msg è formato da tre parti:
 - ▶ un **tipo** (long integer)
 - ▶ la **lunghezza** (≥ 0)
 - ▶ i **dati** effettivi (di qualunque tipo, in genere stringhe)

struttura msqid_ds

» ogni coda di messaggi ha la struttura

`msqid_ds`

associata ad essa che ne definisce lo stato corrente

```
struct msqid_ds {
    struct ipc_perm msg_perm; /* see Section 14.6.2 */
    struct msg *msg_first; /* ptr to first message on queue */
    struct msg *msg_last; /* ptr to last message on queue */
    ulong msg_cbytes; /* current # bytes on queue */
    ulong msg_qnum; /* # of messages on queue */
    ulong msg_qbytes; /* max # of bytes on queue */
    pid_t msg_lspid; /* pid of last msgsnd() */
    pid_t msg_lrpid; /* pid of last msgrcv() */
    time_t msg_stime; /* last-msgsnd() time */
    time_t msg_rtime; /* last-msgrcv() time */
    time_t msg_ctime; /* last-change time */
};
```

Laboratorio di Sistemi Operativi

Limiti di sistema

Name	Description	Typical Value
MSGMAX	The size in bytes of the largest message we can send.	2048
MSGMNB	The maximum size in bytes of a particular queue (i.e., the sum of all the messages on the queue).	4096
MSGMNI	The maximum number of messages queues, systemwide.	50
MSGTQL	The maximum number of messages, systemwide.	40

Laboratorio di Sistemi Operativi

Code di Messaggi

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

```
int msgget(key_t key, int flag);
```

Descrizione: crea o apre una coda di messaggi (in relazione a *key*)

Restituisce: ID della coda se OK
-1 in caso di errore

Laboratorio di Sistemi Operativi

esempio di uso

```
int msqid;
msqid = msgget(IPC_PRIVATE, IPC_CREAT|0660);
```

```
key_t key ;
key = ftok(".", 'm'); /* creazione chiave da un pathname e un char*/
int msqid ;
msqid = msgget (key, IPC_CREAT | 0660);
```

```
int msqid;
key_t key=100;
msqid = msgget(key, IPC_CREAT|0660);
```

Laboratorio di Sistemi Operativi

Funzione msgctl

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

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

Restituisce: 0 se OK
-1 in caso di errore

Funzione msgctl

Effettua varie funzioni sulla coda identificata da *msqid* al variare di *cmd* :

- ▶▶ IPC_STAT : mette in *buf* la struttura *msqid_ds* della coda
- ▶▶ IPC_SET : setta i 4 campi (*msg_perm.uid*, *msg_perm.gid*, *msg_perm.mode*, *msg_qbytes*) della struttura puntata da *buf* nei corrispondenti campi della struttura della coda
- ▶▶ IPC_RMID : rimuovi la coda e ciò che contiene

esempio d'uso

```
int msqid;
... /* rimozione della coda */
msgctl(msqid, IPC_RMID, 0 );
```

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

chiave: personale

```
int main(void)
{
    int msqid;
    key_t key=100;

    if ( (msqid = msgget(key, IPC_CREAT | IPC_EXCL | 0660 )) < 0 ) {
        perror("msgget error");
        exit (1);
    }
    printf("creata la coda di messaggi con id = %d\n", msqid);
    system("ipcs"); sleep(2);
    if ( msgctl(msqid, IPC_RMID,0) < 0 ) {
        perror("remove error");
        exit (1);
    }
    printf("rimossa la coda di messaggi con id = %d\n", msqid);
    sleep(1); system("ipcs"); sleep(2);
    exit(0);
}
```

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

chiave: IPC_PRIVATE

```
int main(void)
{
    int msqid;

    if ( (msqid = msgget(IPC_PRIVATE, IPC_CREAT | IPC_EXCL | 0660)) < 0 ) {
        perror("msgget error");
        exit(1);
    }
    printf("creata la coda di messaggi con id = %d\n", msqid);
    sleep(1); system("ipcs"); sleep(2);
    if ( msgctl(msqid, IPC_RMID, 0) < 0 ) {
        perror("remove error");
        exit(1);
    }
    printf("rimossa la coda di messaggi con id = %d\n", msqid);
    sleep(1); system("ipcs"); sleep(2);
    exit(0);
}
```

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

chiave: ftok

```
int main(void)
{
    int msqid;
    key_t key;

    key=ftok(".", 'm');
    if ( (msqid = msgget(key, IPC_CREAT | IPC_EXCL | 0660)) < 0 ) {
        perror("msgget error");
        exit(1);
    }
    printf("creata la coda di messaggi con id = %d\n", msqid);
    sleep(1); system("ipcs"); sleep(2);
    if ( msgctl(msqid, IPC_RMID, 0) < 0 ) {
        perror("remove error");
        exit(1);
    }
    printf("rimossa la coda di messaggi con id = %d\n", msqid);
    sleep(1); system("ipcs"); sleep(2);
    exit(0);
}
```

Funzione **msgsnd**

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

```
int msgsnd(int msqid, const void *ptr, size_t nbytes, int flag);
```

Descrizione: manda il messaggio puntato da *ptr* alla coda di messaggi il cui identificatore è *msqid*; *flag* può essere IPC_NOWAIT così da evitare di bloccare **msgsnd** se la coda è piena

Restituisce: 0 se OK
-1 in caso di errore

Struttura del messaggio

ptr punta

1. ad un *long integer* che contiene il *tipo* del messaggio
 - ▶ può essere usato dal ricevente per prelevare i dati in base al tipo e non in ordine fifo

questo a sua volta è seguito immediatamente dai dati del messaggio.

2. ad una struttura del tipo:

```
struct mtypes {
    long mtype;
    char mtext[512];
};
```

```

struct mymesgbuf {
    long mtype;
    char mtext[512 ];
} qbuf;

qbuf.mtype = 10;                /* un qualsiasi intero */
sprintf(qbuf.mtext, "FINE");

if ( (msgsnd( qid, &qbuf, sizeof(qbuf), 0 ) ) == -1) {
    perror("msgsnd");
    exit(1);
}

```

Funzione **msgrcv**

```

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

```

```
int msgrcv(int msqid, void *ptr, size_t nbytes, long type, int flag )
```

Descrizione: rimuove dalla coda *msqid* il primo messaggio di tipo *type* e restituisce il suo puntatore con *ptr*

Restituisce: la dimensione del campo mtext se OK
-1 in caso di errore

Argomento *type*

- ▶▶ *type* == 0 (primo messaggio, il più vecchio!)
- ▶▶ *type* > 0 (primo messaggio di questo tipo)
- ▶▶ *type* < 0 (primo msg con tipo $\leq \text{abs}(type)$)

Argomento *flag*

- ▶▶ Se *flag* ha settato il bit IPC_NOWAIT esce con errore ENOMSG se non e' disponibile nessun messaggio del tipo specificato
- ▶▶ Altrimenti, si blocca fino a che
 - un messaggio del tipo specificato e' disponibile
 - la coda e' rimossa dal sistema
 - e' catturato un segnale ed il signal handler lo sblocca
- ▶▶ Se *flag* ha settato il bit MSG_NOERROR un messaggio piu' lungo di nbytes e' troncato e nessuna notifica e' fatta
- ▶▶ Altrimenti c'e' notifica

```
struct mymesgbuf {
    long  mtype;
    char  mtext[512 ];
} qbuf;

printf(" leggo un messaggio...\n");
if (msgrcv(qid, &qbuf, sizeof(qbuf), 10, 0 )== -1) {
    perror("msgrcv error");
    exit (1);
}
printf(" Tipo: %ld Testo: %s\n", qbuf.mtype, qbuf.mtext);

printf(" leggo un altro messaggio...\n");
if (msgrcv(qid, &qbuf, sizeof(qbuf), 0, 0 )== -1) {
    perror("msgrcv error");
    exit (1);
}
printf(" Tipo: %ld Testo: %s\n", qbuf.mtype, qbuf.mtext);
```