Laboratorio di Sistemi Operativi

primavera 2009

System V IPC

Memoria condivisa

Memoria Condivisa

- Permette a due o più processi di condividere una zona di memoria
- È l'IPC più veloce perché in questo caso non è necessario copiare i dati tra server e client
- L'unico problema è la sincronizzazione per l'accesso (spesso i semafori sono usati per effettuare tale sincronizzazione)

Struttura associata

```
struct shmid_ds {
  struct ipc_perm  shm_perm; /* see Section 14.6.2 */
  struct anon_map *shm_amp;  /* pointer in kernel */
  int      shm_segsz;   /* size of segment in bytes */
  ushort   shm_lkcnt;   /* number of times segment is being locked */
  pid_t    shm_lpid;    /* pid of last shmop() */
  pid_t    shm_cpid;    /* pid of creator */
  ulong    shm_nattch;  /* number of current attaches */
  ulong    shm_cnattch; /* used only for shminfo */
  time_t   shm_atime;   /* last-attach time */
  time_t   shm_dtime;   /* last-detach time */
  time_t   shm_ctime;   /* last-change time */
};
```


Limiti di sistema

Name	Description	Typical Value
SHMMAX	The maximum size in bytes of a shared memory segment.	131,072
SHMMIN	The minimum size in bytes of a shared memory segment.	1
SHMMNI	The maximum number of shared memory segments, systemwide.	100
SHMSEG	The maximum number of shared memory segments, per process.	6

Funzione shmget

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

int shmget(key_t key, int size, int flag );
```

Restituisce: ID del segmento di memoria condivisa se OK,
-1 in caso di errore

shmget

- Quando si crea un nuovo segmento alcuni dei membri di shmid_ds sono settati
  - si inizializza ipc_perm con i permessi specificati in *flag*
  - shm_lpid, shm_nattach, shm_atime sono settati a 0
  - shm_ctime è settato all'istante di creazione

Es: shmid = shmget (key, 1000, IPC_CREAT|IPC_EXCL|0666);

- Quando ci si riferisce ad una memoria condivisa esistente si deve mettere size=0

Es: shmid = shmget (key, 0, 0);

Funzione shmctl

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

int shmctl(int shmid, int cmd, struct shmid_ds *buf );
```

Restituisce: 0 se OK
-1 in caso di errore

argomento *cmd*

- *cmd* rappresenta il comando da eseguire su *shmid*

IPC_STAT = recupera la struttura shmid_ds e la conserva nella struttura puntata da *buf*

IPC_SET = setta il campo shm_perm in shmid_ds secondo quanto contenuto nella struttura puntata da *buf*

IPC_RMID = rimuove la memoria condivisa dal sistema

SHM_LOCK = locca la memoria condivisa; può essere eseguito solo da superuser

SHM_UNLOCK = unlocca la memoria condivisa; può essere eseguito solo da superuser

Funzione shmat

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

void *shmat(int shmid, void *addr, int flag );
```

- Il processo che chiama shmat attach la memoria condivisa al suo spazio degli indirizzi (al primo indirizzo utile se *addr* =0; in genere è così)
- Se *flag*=SHM_RDONLY si potrà solo leggere il segmento
- Restituisce il puntatore al segmento se OK, -1 in caso di errore

Funzione shmdt

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

int shmdt(void *addr);
```

- Deattach la memoria condivisa (nota che questo non rimuove la memoria e le sue strutture dal sistema; la rimozione può essere fatta solo con shmctl ed il comando IPC_RMID)
- Restituisce 0 se OK, -1 in caso di errore

esempio d'uso

```
shmid= shmget(IPC_PRIVATE,SEGSIZE,IPC_CREAT|0666);
char *segptr;

segptr=shmat(shmid, 0,SHM_RDONLY);

      \\si utilizza la memoria condivisa

shmdt(segptr)        \\ non rimuove l'id e la struttura dati associata dal
                     \\ sistema, semplicemente non è più a disposizione

shmctl(shmid, IPC_RMID,0); \\ rimuove la shared memory (di solito il
                           \\server)
```


esempio: scrivi su memoria condivisa

```
#include <sys/ipc.h>
#include <stdio.h>
#define SHMSIZE 1000
int main(int argc, char* argv[]) {
    key_t chiaveipc;
    int tmp, shmid;
    char* shptr;
    char buffer[SHMSIZE];
    if (argc != 2) { printf("Mi serve la chiave IPC\n"); exit(1); }
    chiaveipc = atoi(argv[1]);
    printf("Chiave IPC e' %d.\n",chiaveipc);
    shmid=shmget(chiaveipc,SHMSIZE,IPC_CREAT|0600);
```

esempio: scrivi su memoria condivisa

```
shptr = (char*)shmat(shmid,0,0);

   printf("Immetti una stringa:\n
   fgets(buffer,SHMSIZE,stdin);
   strncpy((char*) shptr,buffer,SHMSIZE);
   printf("Fatto.\n");

 shmdt(shptr);
   exit(0);
}
```


esempio: leggi da memoria condivisa

```
#include <sys/ipc.h>
#define SHMSIZE 100
int main(int argc, char* argv[]) {
    key_t chiaveipc;
    int shmid;
    char* shptr;
    char buffer[SHMSIZE];
    if (argc != 2) { printf("Mi serve la chiave IPC\n"); exit(1);}
    chiaveipc = atoi(argv[1]);
    printf("Chiave IPC e' %d.\n",chiaveipc);
    shmid=shmget(chiaveipc,0,0);
```


esempio: leggi da memoria condivisa

```
 shptr = (char*)shmat(shmid,0,0);

   strncpy(buffer,shptr,SHMSIZE);
   buffer[100]=0;
   printf("Nella memoria condivisa c'e':\n%s\n",buffer);

 shmdt(shptr);
   exit(0);
}
```


esempio 2: uso memoria condivisa

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

int main( ) {
    key_t my_key=1234;
     int* ptr;

    shid=shmget(my_key,sizeof(int),IPC_CREAT|0666);

     ptr=shmat(shid,0,0);
     *ptr=0;              \\ si può utilizzare lo spazio di memoria come si utilizza
                  \\ in C un qualunque spazio di memoria puntato da un puntatore
     shmdt(ptr);

exit(0);
}
```