MODELLO SHARED MEMORY

Il modello a memoria condivisa consente a processi diversi di scambiare informazioni usando l'area condivisa. L'area viene trattata come parte dello spazio di indirizzamento di ogni processo.
Ha però bisogno di sincronizzazione.

DETACH

Se un processo termina o fa una exceve(); , si distacca dalla shared memory (non si chiude).

COMM. PADRE-FIGLIO

I figli ereditano il segmento del padre con shared mem.

OPERAZIONI

Le operazioni disponibili sono:

- **shmget** :
  Se non esiste → crea la shared memory.
  Se esiste → ottiene l'id della shared memory (non si collega)
- **shmat** :
  Effettua l'attach, ovvero collega la shared memory all'indirizzamento logico del processo.
  Ritorna un puntatore all'area della SM (inizio segmento)
- **shmdt** :
  Effettua il detach della memoria dal processo.
- **shmctl** :
  Consente eliminazione/gestione della shared memory.

SHMGET

Crea la shared memory e ritorna l'ID

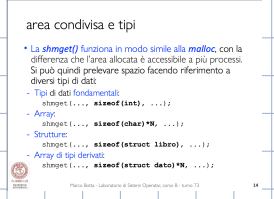
```
#include <sys/types.h>  /* For portability */
#include <sys/shm.h>

int shmget(key_t key, size_t size, int shmflg);

        Returns shared memory segment identifier on
                                success, or -1 on error
```

- key: La chiave generabile con IPC_PRIVATE, ftok()...
- size: Definisce la dimensione del segmento in Bytes.
  Se in collegamento, deve essere ≤ alla dimensione.

Si può usare sizeof() come la malloc() per allocare tipi definiti.



- flg: Specifica dei flag aggiuntivi come:
  - IPC_CREAT: Se non c'è, crea
  - IPC_EXCL: Se esiste già, errore

e permessi

dipende da EEXIST

SHMAT >

```
#include <sys/types.h> /* For portability */
#include <sys/shm.h>

void *shmat(int shmid, const void *shmaddr, int shmflg);

     Returns address at which shared memory is attached on
                          success, or (void *) -1 on error
```

Effettua l'attach della memoria cond:

- shmid:
  L'ID della shared memory a cui collegarsi.
- shmaddr:
  Consente di specificare l'indirizzo a cui collegarsi. È consigliato impostarlo a null per lasciar fare al kernel.
- shmflg:
  Consente di specificare flag aggiuntivi:
  - SHM_RDONLY: Imposta l'attach in sola lettura
  - SHM_REMAP: Rimpiazza i mapping in shmaddr.

Il valore di ritorno è un void* che può essere castato come la malloc();

SHMDT >

```
#include <sys/types.h> /* For portability */
#include <sys/shm.h>

int shmdt(const void *shmaddr);

                          Returns 0 on success, or -1 on error
```

Effettua la detach del processo.

- shmaddr: l'area di memoria da staccare, dovrebbe essere generata da una shmat();

La detach è opzionale

```
00 int main(int argc, char** argv) {
01   int shmid_1, shmid_2, return_val;
02   char *stringa_1, *stringa_2;
03   char msg[] = "ciao a tutti!";
04
05   shmid_1 = shmget(MYKEY, sizeof(char)*SHMSZ, IPC_CREAT|0666);
06   stringa_1 = (char *)shmat(shmid_1,NULL,0);
07
08   snprintf(stringa_1, sizeof(msg), "%s", msg);
09   return_val = shmdt(stringa_1);
10
11   // un altro processo che si attacchi alla stessa area
12   // di memoria condivisa potrà leggerne il contenuto
13   shmid_2 = shmget(MYKEY, sizeof(char)*SHMSZ, 0);
14
15   stringa_2 = (char *)shmat(shmid_2,NULL,0);
16   printf("%s\n",stringa_2); // leggo il contenuto della SM
17
18   shmctl(shmid_2, IPC_RMID,0);
19   exit(EXIT_SUCCESS);
20 }
```

Marco Botta - Laboratorio di Sistemi Operativi, corso B - turno T3 22

SHMCTL >

```
#include <sys/types.h>  /* For portability */
#include <sys/shm.h>

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

                          Returns 0 on success, or -1 on error
```

Consente di gestire la shared mem:

- shmid : L'ID della shared mem.
- cmd : L'operazione da eseguire:
  - IPC_STAT : copia la control struct in buf
  - IPC_SET : imposta i dati dalla control struct in buf.
  - IPC_RMID : Elimina la struct con l'ID.
- buf : Il buffer usato nei cmd

```
struct shmid_ds {
  struct ipc_perm shm_perm; /*Ownership e permissions */
  size_t shm_segsz; /* Size of segment in bytes */
  time_t shm_atime; /* Time of last shmat() */
  time_t shm_dtime; /* Time of last shmdt() */
  time_t shm_ctime; /* Time of last change */
  pid_t shm_cpid; /* PID of last creator */
  pid_t shm_lpid; /* PID of last shmat() / shmdt() */
  shmatt_t shm_nattch; /* Number of currently attached
                          processes */
}
```