Introdução

O compartilhamento de uma região de memória entre dois ou mais processos (executando programas) corresponde a maneira mais rápida deles efetuarem uma troca de dados. A zona de memória compartilhada (denominada segmento de memória compartilhada) é utilizada por cada um dos processos como se ela fosse um espaço de endereçamento que pertencesse a cada um dos programas. Em outras palavras, o compartilhamento de memória permite aos processos de trabalhar sob um espaço de endereçamento comum em memória virtual. Em conseqüência, este mecanismo é dependente da forma de gerenciamento da memória; isto significa que as funcionalidades deste tipo de comunicação interprocessos são fortemente ligadas ao tipo de arquitetura (máquina) sobre a qual a implementação é realizada.

Princípio da memória compartilhada

Um processo pode criar um segmento de memória compartilhada e suas estruturas de controle através da função `shmget()`. Durante essa criação, os processos devem definir: as permissões de acesso ao segmento de memória compartilhada; o tamanho de bytes do segmento e; a possibilidade de especificar que a forma de acesso de um processo ao segmento será apenas em modo leitura. Para poder ler e escrever nessa zona de memória, é necessário estar de posse do identificador (ID) de memória comum, chamado `shmid`. Este identificador é fornecido pelo sistema (durante a chamada da função `shmget()`) para todo processo que fornece a chave associada ao segmento. Após a criação de um segmento de memória compartilhada, duas operações poderão ser executadas por um processo:

- acoplamento (*attachment*) ao segmento de memória compartilhada, através da função `shmat()`;
- desacoplamento da memória compartilhada, utilizando a função `shmdt()`.

O acoplamento à memória compartilhada permite ao processo de se associar ao segmento de memória: ele recupera, executando `shmat()`, um ponteiro apontando para o início da zona de memória que ele pode utilizar, assim como todos os outros ponteiros para leitura e escrita no segmento.

O desacoplamento da memória compartilhada permite ao processo de se desassociar de um segmento quando ele não desejar mais utilizá-lo. Após esta operação, o processo perde a possibilidade de ler ou escrever neste segmento de memória compartilhada.

A Função `shmget()`

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

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

Valor de retorno: o identificador do segmento de memória compartilhada `shmid`, ou -1 em caso de erro.

Esta função é encarregada de buscar o elemento especificado (pela chave de acesso `key`) na estrutura `shmid_ds` e, caso esse elemento não exista, de criar um novo segmento de memória compartilhada, com tamanho em bytes igual a `size`. Além da chave de acesso `key` e do tamanho do segmento (`size`), um terceiro argumento (`shmflg`) é empregado para definir os direitos de acesso ao segmento criado.

O argumento `key` pode conter os seguintes valores:

- IPC_PRIVATE (=0): indicando que a zona de memória não tem chave de acesso, e que somente o processo proprietário tem acesso ao segmento de memória compartilhada.
- o valor desejado para a chave de acesso do segmento de memória. Observe que para a geração de uma chave única no sistema deve-se utilizar a função `ftok` apresentada na seção 1.6

O argumento `shmflg` é bastante semelhante ao `semflg` utilizado para semáforos (ver seção 5.3. Este flag corresponde à combinação de diferentes constantes pré-definidas através do operador lógico OU). O argumento `shmflg` permite assim a especificação dos direitos de acesso ao segmento de memória compartilhada criado. As possíveis constantes a serem combinadas são: `IPC_CREAT`, `IPC_EXCL` similares àquelas dos semáforos e, `SHM_R` (=0400) e `SHM_W`(=200) que dão o direito de leitura e escrita ao segmento. Note que a combinação destas útimas constantes pode ser igualmente representada pelo octal 0600.

Existe muita semelhança entre os direitos de acesso aos segmentos criados e aos arquivos no sistema `UNIX` através da noção de direitos de leitura e escrita para o usuário, para o grupo e para outros. O número octal definido de maneira similar àquela mostrada em 1.4.2 pode ser utilizado.

Estrutura associada a uma memória comum: `shmid_ds`

Quando um novo segmento de memória é criado, as permissões de acesso definidas pelo parâmetro `shmflg` são copiadas no membro `shm_perm` da estrutura `shmid_ds` que define efetivamente o segmento. A estrutura `shmid_ds` é mostrada a seguir:

```
struct shmid_ds
     {
       struct    ipc_perm shm_perm;  /* operation permissions */
       int  shm_segsz;               /* size of segment (bytes) */
       time_t    shm_atime;          /* last attach time */
       time_t    shm_dtime;          /* last detach time */
       time_t    shm_ctime;          /* last change time */
       unsigned short shm_cpid;      /* pid of creator */
       unsigned short shm_lpid;      /* pid of last operator */
       short     shm_nattch;         /* no. of current attaches */
     };
```

Os campos no membro `shm_perm` são os seguintes:

```
struct ipc_perm
     {
       key_t  key;
       ushort uid;   /* owner euid and egid */
       ushort gid;
       ushort cuid;  /* creator euid and egid */
       ushort cgid;
       ushort mode;  /* lower 9 bits of shmflg */
       ushort seq;   /* sequence number */
     };
```

Como criar um segmento de memória compartilhada

O procedimento é exatamente o mesmo que aquele empregaado para gerar um conjunto de semáforos (seção 5.3). As seguintes regras gerais devem ser entretanto observadas:

- `key` deve contar um valor identificando o segmento (diferente de `IPC_PRIVATE` = 0);
- `shmflg` deve conter os direitos de acesso desejadas para o segmento, e ainda a constante `IPC_CREAT`.
- Se deseja-se testar a existência ou não de um segmento associado a uma chave especificada, deve-se adicionar (OU lógico) a constante `IPC_EXCL` ao argumento `shmflg`. A chamada `shmget` irá falhar se esse segmento existir.

Note finalmente que durante a criação do segmento de memória compartilhada, um certo número de membros da estrutura `shmid_ds` serão também inicializados (por exemplo, o proprietário, os modos de acesso, a data de criação, etc). Faça `man shmget` para maiores detalhes.

Exemplo de utilização de `shmget`

Este programa cria um segmento de memória compartilhada associado à chave 123.

```
/* fichier test_shmget.c */
/* exemplo de utilizacap de shmget() */

#include <errno.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>

#define KEY 123

int main()
{
  int shmid ;  /* identificador da memoria comum */
  int size = 1024 ;
  char *path="nome_de_arquivo_existente" ;

     if (( shmid = shmget(ftok(path,(key_t)KEY), size,
                   IPC_CREAT|IPC_EXCL|SHM_R|SHM_W)) == -1) {
          perror("Erro no shmget") ;
          exit(1) ;
     }
     printf("Identificador do segmento: %d \n",shmid) ;
     printf("Este segmento e associado a chave unica: %d\n",
                   ftok(path,(key_t)KEY)) ;
     exit(0);
}
```

Resultado da execução:

Lançando duas vezes a execução do programa, tem-se o seguinte resultado:

```
euler:~> test_shmget
Identificador do segmento: 36096
Este segmento e associado a chave unica: 2063804629
euler:~> ipcs -m

------ Shared Memory Segments --------
```

```
key       shmid     owner     perms     bytes     nattch    status
0x7b0328d5 36096     saibel    600       1024      0

euler:~> test_shmget
Erro no shmget: File exists
```

A Função `shmctl()`

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

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

Valor de retorno: 0 em caso de sucesso, senão -1.

A função `semctl()` é utilizada para examinar e modificar as informações relativas ao segmento de memória compartilhada. De maneira intuitiva, ela permite ao usuário de receber informações relativas ao segmento, de setar o proprietário ou grupo, de especificar as permissões de acesso, e finalmente, de destruir o segmento.

A função utiliza três argumentos: um identificador do segmento de memória compartilhada `shmid`, um parâmetro de comando `cmd` e um ponteiro para uma estrutura do tipo `shmid_ds` associada pelo sistema ao segmento de memória onde a operação será executada.

O argumento `cmd` pode conter os seguintes valores:

- `IPC_RMID (0)`: O segmento de memória será destruído. O usuário deve ser o proprietário, o criador, ou o super-usuário para realizar esta operação; todas as outras operações em curso sobre esse segmento irão falhar;
- `IPC_SET (1)`: dá ao identificador do grupo, ao identificador do usuário, e aos direitos de acesso, os valores contidos no campo `shm_perm` da estrutura apontada por `buf`; a hora da modificação é também atualizada (membro `shm_ctime`);
- `IPC_STAT (2)`: é usada para copiar a informação sobre a memória compartilhada no buffer `buf`;

O super usuário pode ainda evitar ou permitir o *swap* do segmento de memória compartilhada usando os valores `SHM_LOCK (3)` (evitar o *swap* e `SHM_UNLOCK (4)` (permitir o *swap*).

Exemplo:

Neste exemplo, supõe-se que o segmento de memória compartilhada tem a chave de acesso 123 utilizada no exemplo anterior:

```
                /* arquivo test_shmctl.c */

#include <errno.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#define KEY 123

struct shmid_ds buf ;
```

```
int main()
{
  char *path = "nome_de_arquivo_existente" ;
  int shmid ;
  int size = 1024 ;

  /* recuperacao do identificador do segmento associado a chave 123 */
     if (( shmid = shmget(ftok(path,(key_t)KEY),size,0)) == -1 ) {
          perror ("Erro shmget()") ;
          exit(1) ;
     }
      /* recuperacao das informacoes realtivas ao segmento */
     if ( shmctl(shmid,IPC_STAT,&buf) == -1){
          perror("Erro shmctl()") ;
          exit(1) ;
     }
     printf("ESTADO DO SEGMENTO DE MEMORIA COMPARTILHADA %d\n",shmid) ;
     printf("ID do usuario proprietario: %d\n",buf.shm_perm.uid) ;
     printf("ID do grupo do proprietario: %d\n",buf.shm_perm.gid) ;
     printf("ID do usuario criador: %d\n",buf.shm_perm.cuid) ;
     printf("ID do grupo criador: %d\n",buf.shm_perm.cgid) ;
     printf("Modo de acesso: %d\n",buf.shm_perm.mode) ;
     printf("Tamanho da zona de memoria: %d\n",buf.shm_segsz) ;
     printf("pid do criador: %d\n",buf.shm_cpid) ;
     printf("pid  (ultima operacao): %d\n",buf.shm_lpid) ;

     /* destruicao do segmento */
     if ((shmctl(shmid, IPC_RMID, NULL)) == -1){
          perror("Erro shmctl()") ;
          exit(1) ;
     }
     exit(0);
}
```

Resultado da execução

```
euler:~/> test_shmctl
ESTADO DO SEGMENTO DE MEMORIA COMPARTILHADA 35968
ID do usuario proprietario: 1145
ID do grupo do proprietario: 1000
ID do usuario criador: 1145
ID do grupo criador: 1000
Modo de acesso: 384
Tamanho da zona de memoria: 1024
pid do criador: 930
pid  (ultima operacao): 0
euler:~> ipcs -m

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status
```

Função `shmat()`

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

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

Valor de retorno: endereço do segmento de memória compartilhada, ou -1 em caso de erro.

Antes que o processo possa utilizar um segmento de memória criado por outro processo, ele deve inicialmente se acoplar a esse segmento. É exatamente a função `shmat()` que faz esse papel. Ela "acopla" *(*attaches) o segmento de memória compartilhada identificado por `shmid` ao segmento de dados do processo que a chamou. A função exige três argumentos: o identificador do segmento `shmid`, um ponteiro `shmaddr` especificando o endereço de acoplamento e um conjunto de flags, `shmflg`.

O endereço de acoplamento é especificado através dos dois últimos parâmetros `shmaddr` e `shmflg`:

- Se `shmaddr` é 0, o segmento é acoplado ao primeiro endereço possível determinado pelo sistema (caso mais comum na prática);
- Se `shmaddr` não é 0, observa-se então `shmflg`:
    - Se o flag `SHM_RND` não está posicionado em `shmflg`, o acoplamento ocorre no endereço especificado por `shmaddr`;
    - Se `SHM_RND` está posicionado em `shmflg`, o acoplamento ocorre no endereço especificado pelo menor inteiro resultante da divisão de `shmaddr` por `SHMLBA`, uma constante pré-definida do sistema em `<sys/shm.h>` associada ao tamanho da página na memória física.

Observações:

Quando a função `shmat` é chamada, o sistema verifica se existe espaço suficiente no espaço de endereçamento da memória virtual do processo ao qual deve ser acoplado o segmento de memória compartilhada. Se esta não for o caso, um código de erro será retornado. Note ainda que não existe efetivamente uma cópia da zona de memória, mais simplesmente um redirecionamento do endereçamento para o segmento de memória que está sendo compartilhado.

Exemplo:

Suponha que um segmento de memória compartilhada tenha sido criado anteriormente através do programa `test_shmget`. O programa `test_shmat` vai reacoplar um processo ao segmento e escrever na memória comum, uma cadeia de caracteres. O programa `test_shmat2` irá então se acoplar à mesma zona de memória e ler então seu conteúdo. O programa `test_shmctl` irá então obter informações sobre o segmento de memória antes de destruí-lo.

```
                /* arquivo test_shmat.c */

/*
 * exemplo de utilizacao de shmat()
 * escrita num segmento de memoria compartilhada
 */

#include <errno.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>

#define KEY 123
#define KEY 123
#define MSG "Mensagem escrita na memoria comum"

int main()
{
```

```
  int shmid ;  /* identificador da memoria comum */
  int size = 1024 ;
  char *path="nome_de_arquivo_existente" ;
  char *mem ;
  int flag = 0;

     /*
     * recuperacao do shmid
     */
     if (( shmid = shmget(ftok(path,(key_t)KEY), size,0)) == -1) {
          perror("Erro no shmget") ;
          exit(1) ;
     }
     printf("Sou o processo com pid: %d \n",getpid()) ;
     printf("Identificador do segmento recuperado: %d \n",shmid) ;
     printf("Este segmento e associado a chave unica: %d\n",
                   ftok(path,(key_t)KEY)) ;
    /*
     * acoplamento do processo a zona de memoria
     * recuperacao do pornteiro sobre a area de memoria comum
     */
     if ((mem = shmat (shmid, 0, flag)) == (char*)-1){
          perror("acoplamento impossivel") ;
          exit (1) ;
     }

    /*
     * escrita na zona de memoria compartilhada
     */
     strcpy(mem,MSG);
     exit(0);
}

                /* fichier test_shmat2.c */

/*
 * programa para ler o conteudo de um segmento de memoria
 * compartilhada que foi preenchido anteriormente por outro processo
 */

#include <errno.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>

#define KEY 123

int main()
{
  int shmid ;  /* identificateur de la memoire commune */
  int size = 1000 ;
  char *path="nome_de_arquivo_existente" ;
  char *mem ;
  int flag = 0 ;

      /*
     * recuperacao do shmid
     */
     if (( shmid = shmget(ftok(path,(key_t)KEY), size,0)) == -1) {
          perror("Erro no shmget") ;
          exit(1) ;
```

```
     }
     printf("Sou o processo com pid: %d \n",getpid()) ;
     printf("Identificador do segmento recuperado: %d \n",shmid) ;
     printf("Este segmento e associado a chave unica: %d\n",
                  ftok(path,(key_t)KEY)) ;
    /*
     * acoplamento do processo a zona de memoria
     * recuperacao do pornteiro sobre a area de memoria comum
     */
     if ((mem = shmat (shmid, 0, flag)) == (char*)-1){
          perror("acoplamento impossivel") ;
          exit (1) ;
     }
    /*
     * tratamento do conteudo do segmento
     */
     printf("leitura do segmento de memoria compartilhada:\n");
     printf("\t==>%s\n",mem) ;
     exit(0);
}
```

Resultado da execução:

```
euler:~/> test_shmget
Identificador do segmento: 41600
Este segmento e associado a chave unica: 2063804629
euler:~/> test_shmat
Sou o processo com pid: 1250
Identificador do segmento recuperado: 41600
Este segmento e associado a chave unica: 2063804629
euler:~/> test_shmat2
Sou o processo com pid: 1251
Identificador do segmento recuperado: 41600
Este segmento e associado a chave unica: 2063804629
leitura do segmento de memoria compartilhada:
        ==>Mensagem escrita na memoria comum
euler:~/> test_shmctl
ESTADO DO SEGMENTO DE MEMORIA COMPARTILHADA 41600
ID do usuario proprietario: 1145
ID do grupo do proprietario: 1000
ID do usuario criador: 1145
ID do grupo criador: 1000
Modo de acesso: 384
Tamanho da zona de memoria: 1024
pid do criador: 1249
pid  (ultima operacao): 1251
```

Note que após o lançamento em seqüência dos programas, o processo com `pid` = 1249, correspondente à execução de `test_shmget` cria o segmento de memória. Depois, esse segmento será acessado por dois processos, sendo que o último é aquele com `pid` = 1251, correspondente à execução de `test_shmat2`.

Função `shmdt()`

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

   int shmdt ( const void *shmaddr)
```

Valor de retorno: 0 em caso de sucesso e -1 em caso de erro.

A função `shmdt()` desacopla (*detaches*)o segmento de memória compartilhada especificado pelo endereço `shmaddr` do espaço de endereçamento processo que a chama. Obviamente, o segmento desacoplado deve ser um dentre os segmentos previamente acoplados pelo processo usando `shmat`. Este segmento não poderá mais utilizado pelo processo após a chamada da função.

Exemplo:

```
                /* arquivo test_shmdt.c */
/*
 * este programa permite a leitura do conteudo de um segmento de
 * memoria compartilhada que foi preenchido por algum processo
 * anteriormente. O processo vai se desacoplar do segmento apos
 * a leitura
 */

#include <errno.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>

#define KEY 123
#define MSG "Mensagem escrita na memoria comum"

int main()
{
  int shmid ;  /* identificador da memoria comum */
  int size = 1024 ;
  char *path="nome_de_arquivo_existente" ;
  char *mem ;
  int flag = 0;

     /*
     * recuperacao do shmid
     */
     if (( shmid = shmget(ftok(path,(key_t)KEY), size,0)) == -1) {
          perror("Erro no shmget") ;
          exit(1) ;
     }
     printf("Sou o processo com pid: %d \n",getpid()) ;
     printf("Identificador do segmento recuperado: %d \n",shmid) ;
     printf("Este segmento e associado a chave unica: %d\n",
                   ftok(path,(key_t)KEY)) ;
    /*
     * acoplamento do processo a zona de memoria
     * recuperacao do ponteiro sobre a zona de memoria comum
     */
     if ((mem = shmat (shmid, 0, flag)) == (char*)-1){
          perror("acoplamento impossivel") ;
          exit (1) ;
     }
```

```
    /*
     * tratamento do conteudo do segmento
     */
     printf("leitura do segmento de memoria compartilhada:\n");
     printf("\t==>%s\n",mem) ;
    /*
     * desacoplamento do segmento
     */
     if (shmdt(mem)== -1){
          perror("acoplamento impossivel") ;
          exit(1) ;
     }
    /*
     * destruicao do segmento
     */
     if ( shmctl(shmid, IPC_RMID,0) == -1){
          perror("destruicao impossivel") ;
          exit(1) ;
     }
     exit(0);
}
```

Resultado da execução:

Após o lançamento em seqüência dos programas `test_shmget` (para criar um segmento de memória compartilhada), `test_shmat` (para acoplar um processo ao segmento e escrever uma mensagem na zona de memória compartilhada) e `test_shmdt` (para ler o conteúdo, desacoplar e destruir o segmento de memória compartilhada, tem-se a seguinte saída na janela de execução:

```
euler:~/> test_shmget
Identificador do segmento: 43136
Este segmento e associado a chave unica: 2063804629
euler:~/> test_shmat
Sou o processo com pid: 788
Identificador do segmento recuperado: 43136
Este segmento e associado a chave unica: 2063804629
euler:~/> test_shmdt
Sou o processo com pid: 789
Identificador do segmento recuperado: 43136
Este segmento e associado a chave unica: 2063804629
leitura do segmento de memoria compartilhada:
        ==>Mensagem escrita na memoria comum
```