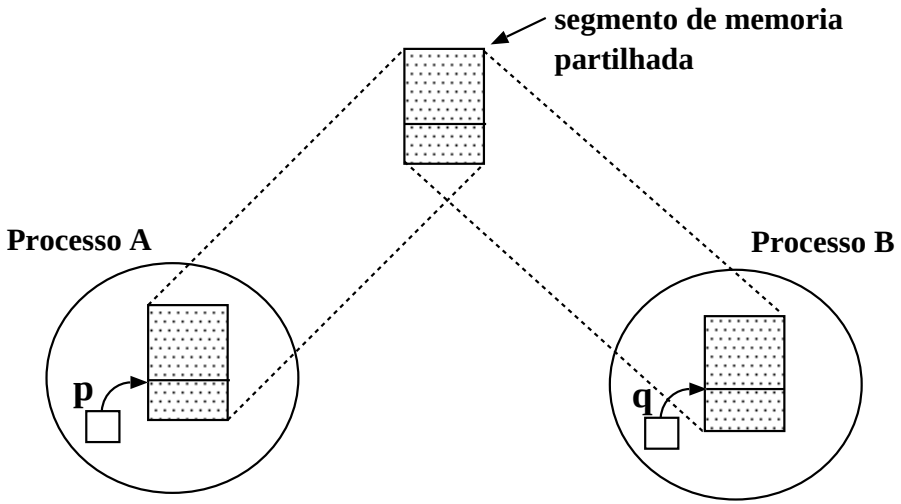
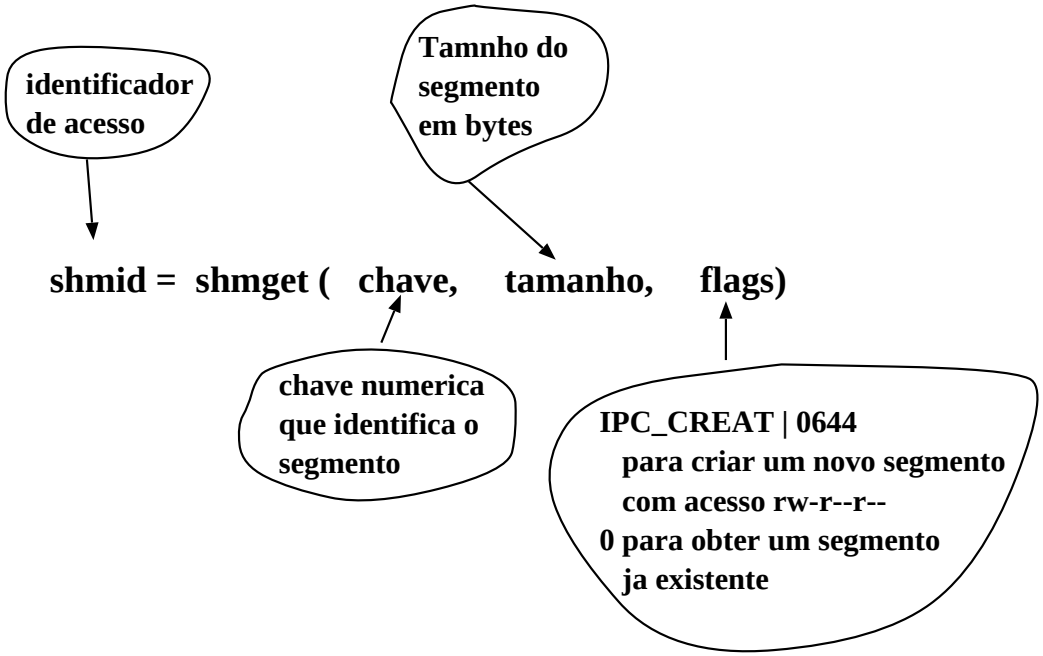
Memória partilhada em Unix SysV

A forma mais geral de comunicação entre processos é através de memória partilhada.

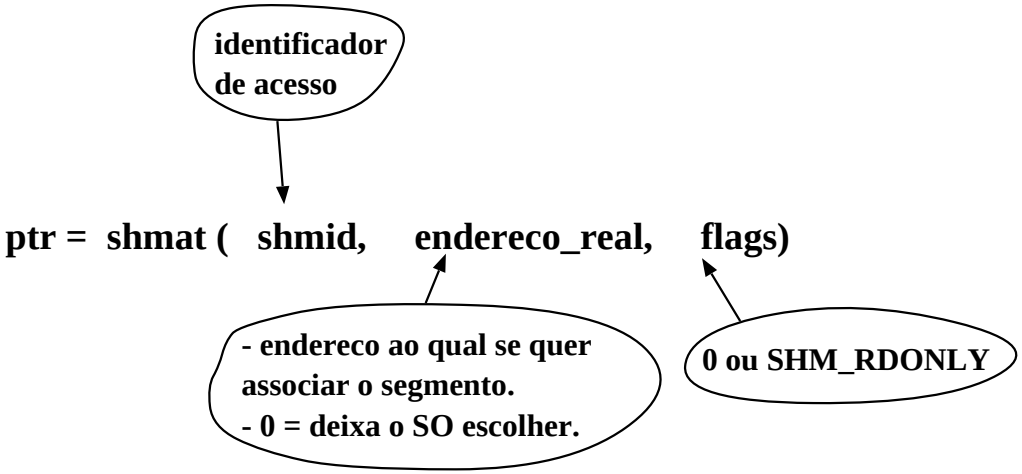


→ Criação de um segmento de memória partilhada:


identificador de acesso
Tamnho do segmento em bytes
shmid = shmget (chave, tamanho, flags)
chave numerica que identifica o segmento
IPC_CREAT | 0644
para criar um novo segmento com acesso rw-r--r--
0 para obter um segmento ja existente

Memória partilhada (cont.)

→ Ligar um segmento de memória partilhada a um endereço virtual no espaço de endereçamento do processo:



→ Desligar o segmento de memória partilhada do espaço do processo:

`int shmdt(ptr)` – o segmento deixa de ficar associado ao endereço local ao processo, `ptr`.

→ Remover o segmento de memória partilhada:

`int shmctl(shmid, cmd, buffer)` – caso `cmd` seja `IPC_RMID`, remove o segmento de memória partilhada do sistema.

→ O número de segmentos de memória partilhada que é possível criar é limitado. O SO fornece dois comandos para lidar com segmentos:

- `ipcs` – vêr que segmentos estão a ser usados;
- `ipcrm` – remover um segmento.

Memória partilhada (cont.): `ipcs`

```
------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status
0x00000000 0          root       777        139264     2
0x00000000 65537      ines       600        393216     2          dest
0x00000000 98306      root       777        4096       2
0x00000000 131075     root       777        4096       2
0x00000000 163844     ines       600        393216     2          dest
0x00000000 196613     ines       600        393216     2          dest
0x00000000 229382     root       777        4096       2
0x00000000 262151     root       777        4096       2
0x00000000 294920     root       777        4096       2
```

Exemplo: shmget() e shmat()

Exemplo de 2 programas (um servidor e outro cliente) que comunicam através de memória partilhada.

```
1  #include <sys/ipc.h>
2  #include <sys/shm.h>
3  #define SHMSZ  27
4  main() {
5    char c, *shm, *s;
6    int chave= 5678, shmid;
7
8    shmid= shmget(chave, SHMSZ, (IPC_CREAT|0666));
9    shm= (char *)shmat(shmid, NULL, 0);
10
11   s= shm;                  /* escreve info em memória */
12   for (c='a'; c<='z'; c++)
13     *s++= c;
14   *s= '\0';                /* espera até que outro proc.
15                               altere o 1o. char em memória */
16   while (*shm != '*')
17     sleep(1);
18   shmdt(shmid);  /* liberta segmento */
19   exit(0);
20 }
```

Exemplo shmget() e shmat() (cont.)

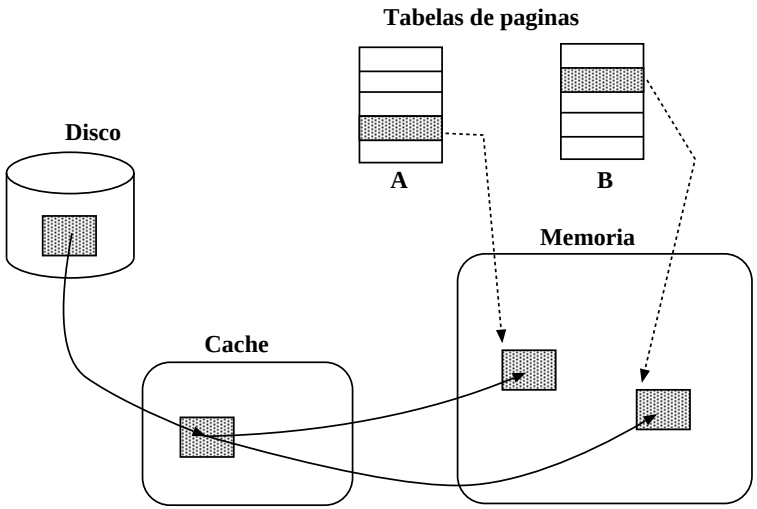
Programa para ler da memória partilhada:

```
1  #include <sys/ipc.h>
2  #include <sys/shm.h>
3  #define SHMSZ  27
4  main()
5  {
6    char c, *shm, *s;
7    int chave= 5678, shmid;
8
9    shmid= shmget(chave, SHMSZ, 0666);
10   shm= (char *)shmat(shmid, NULL, 0);
11
12   for (s=shm; *s!='\0'; s++) /* lê da memória partilhada*/
13     putchar(*s);
14   putchar('\n');
15   *shm='*';          /* alterar o 1o. caracter em memória */
16   exit(0);
17 }
```

Mapeamento de Memória em Ficheiros

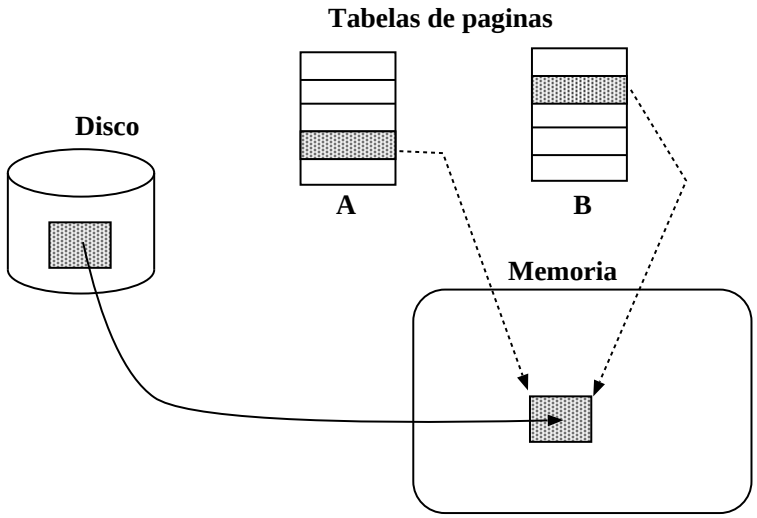
• Modo tradicional de acesso a ficheiros em Unix: open + (read ou write ou lseek) + close.

Dois processos mapeiam no seu espaço uma cópia da mesma página de um ficheiro:



Mapeamento de Memória em Ficheiros

Dois processos mapeiam a mesma página no seu espaço:



Esta página pode ser mapeada como partilhada ou como privada. Neste caso não há garantia de atomicidade na escrita e leitura do ficheiro (como acontece com read e write).

Função mmap()

```
aptr= mmap(endereço,tam,prot,flags,fd,offset)
```

onde,

- `aptr` é o endereço onde está colocado o mapeamento.
- `endereço` sugere um endereço em memória para o mapeamento. `0` ou `NULL`: o sistema escolhe.
- `tam` é o tamanho em bytes.
- `prot` é `PROT_READ` ou `PROT_WRITE`
- `flags`: `MAP_SHARED`
- `fd` é o descritor do ficheiro (é necessário abrir o ficheiro antes!)
- `offset` deslocamento no ficheiro onde começar o mapeamento.

Atenção aos passos para cada processo:

- obter o descritor do ficheiro
- fazer o mmap que retorna um apontador
- ler e escrever sobre o ficheiro através do apontador.

Exemplo com mmap(): rank-sort

```
1  #define N 6 #define SIZE 4*1024 int a[N]; int *b;
2  main() {                                    void putInPlace(int i) {
3    int childpid, status, fd, i;                int t, j, rank;
4    printf("Vector A:\n");                      t = a[i];
5    for(i=0; i<N; i++) {                        rank = 0;
6      a[i]= N-i-1; printf("%d ",a[i]);          for (j=0; j<N; j++)
7    }                                             if (t>a[j]) rank++;
8    printf("\n");                               b[rank] = t; exit(0); }
9    fd= open("tmp.mmap",O_RDWR);  /* abre o ficheiro          */
10   lseek(fd,SIZE,SEEK_SET);      /* truque para garantir     */
11   write(fd,"",1);               /* tamanho do fich é SIZE */
12   b=(int *)mmap(0, (N+1)*sizeof(int),
13           PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
14   for(i=0; i< N; i++)
15     if ((childpid = fork()) == 0)
16       putInPlace(i);
17   for(i=0; i<N; i++)
18     wait(&status);
19   printf("Vector B:\n");
20   for(i=0; i<N; i++)
21     printf("%d ",b[i]);
22   printf("\n");
23 }
```