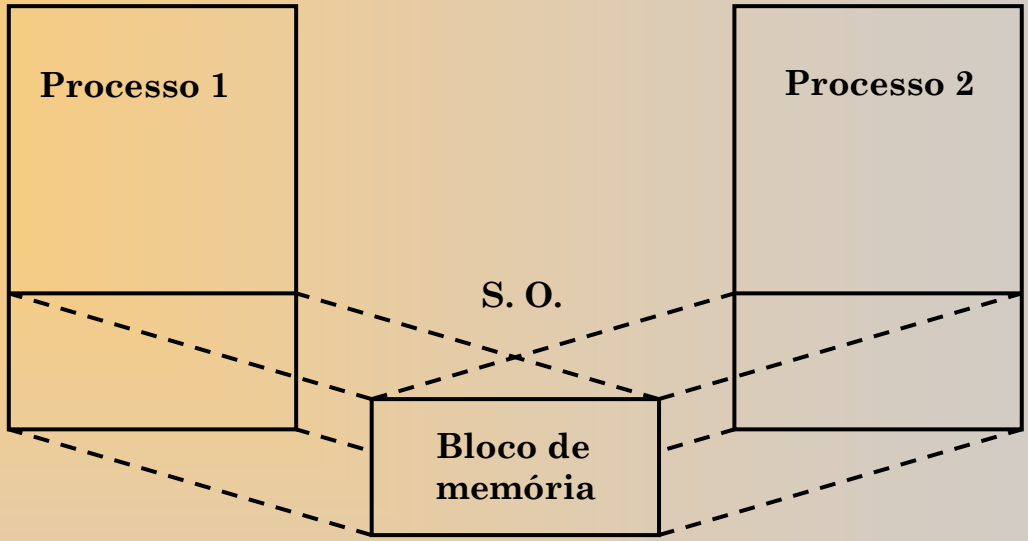
Programação de sistema UNIX

Memória partilhada

Criação e acesso a um bloco (System V)




```
#include <sys/mman.h>
#include <sys/stat.h>
#include <fcntl.h>

int shm_open(char *name, int oflags, mode_t mode);
```

cria/abre um bloco de memória partilhado com nome e seu descritor

-1 erro, >=0 descritor

O_RDONLY, O_RDWR, O_CREAT, O_EXCL

```
#include <unistd.h>

int ftruncate(int fd, off_t len);
```

definir tamanho do bloco

tamanho em bytes

```
void *mmap(void *addr, size_t len, int prot, int flags, int fd, off_t offset);
```

NULL

PROT_READ

PROT_WRITE

=0

MAP_SHARED

endereço mapeado ou MAP_FAILED (erro)

mapear bloco no espaço do processo

Desassociar e destruir um bloco

```
#include <sys/mman.h>

int munmap(void *addr, size_t len);
```

-1, erro

desassocia o bloco de memória com endereço *addr*

***len* é o tamanho em bytes e deve coincidir com o mapeado**

```
int shm_unlink(char* name);
```

Liberta o bloco de memória com nome *name*.
Só acontece quando todos os processos o tiverem desassociado (com munmap( ))

É muito importante destruir os os blocos de memória partilhada:
a terminação dos processos não o faz

Exemplo

Processo 1

```
int main(void)
{
    int *i1, *i2, *i3, smfd, sz;
    char *flag;

    sz = 3*sizeof(int)+sizeof(char);
    smfd = shm_open("/shm1", O_RDWR|O_CREAT,
                            0600);
    ftruncate(smfd, sz);
    i1 = (int *) mmap(NULL, sz, PROT_READ|
            PROT_WRITE, MAP_SHARED, smfd, 0);
    i2 = i1 + 1;
    i3 = i2 + 1;
    flag = (char *) (i3 + 1);
    *i1 = 20;
    *i2 = 30;
    *flag = 1;
    while (*flag != 2) { }
    printf("resultado = %d\n", *i3);
    munmap(i1, sz);
    shm_unlink("/shm1");
    return 0;
}
```

Processo 2

```
int main(void)
{
    int *i1, *i2, *i3, smfd, sz;
    char *flag;

    sz = 3*sizeof(int)+sizeof(char);
    smfd = shm_open("/shm1", O_RDWR,
                            0600);
    i1 = (int *) mmap(mmap(NULL, sz,
                PROT_READ|PROT_WRITE,
                MAP_SHARED, smfd, 0);
    i2 = i1 + 1;
    i3 = i2 + 1;
    flag = (char *) (i3 + 1);
    while (*flag != 1) { }
    *i3 = *i1  *  *i2;
    *flag = 2;
    munmap(i1);
    return 0;
}
```