# Memória Compartilhada

Laboratório de Sistemas Operacionais

Prof. MSc. João Tavares

# Introdução

- Três modelos de comunicação
  - Filas de mensagens
  - Semáforos
  - Memória compartilhada

- Duas APIs
  - XSI IPC → tradicional, inspirada na API do System V
  * Utiliza namespace independente e utilitários específicos
  * Amplamente disponível; utilizada em aplicações clássicas como o servidor X11
  - POSIX IPC → sugestão para novas aplicações
  * Utiliza namespace e utilitários do sistema de arquivos
  * Não disponível no Linux antes do kernel 2.6

# POSIX Shared Memory

- Comunicação através de segmentos de memória que são compartilhados entre processos

- **Segmento de Memória Compartilhada**
  - Semelhante a um array que existe fora do processo
  - Possui um nome e permissões como um arquivo
  - Possui um tamanho
  - Pode ser mapeado (tornado visível) por vários processos simultaneamente
  * Processos modificam/leêm a memória diretamente, sem intervenção do kernel
  * Alterações realizadas por um processo ficam imediatamente disponíveis a todos os demais processos

# Memória compartilhada

- Atrativo → forma mais rápida e simples de IPC
  - Evita múltiplas cópias da informação em buffers internos do SO
  - Basta escrever no segmento e todos os demais processos já verão a nova informação

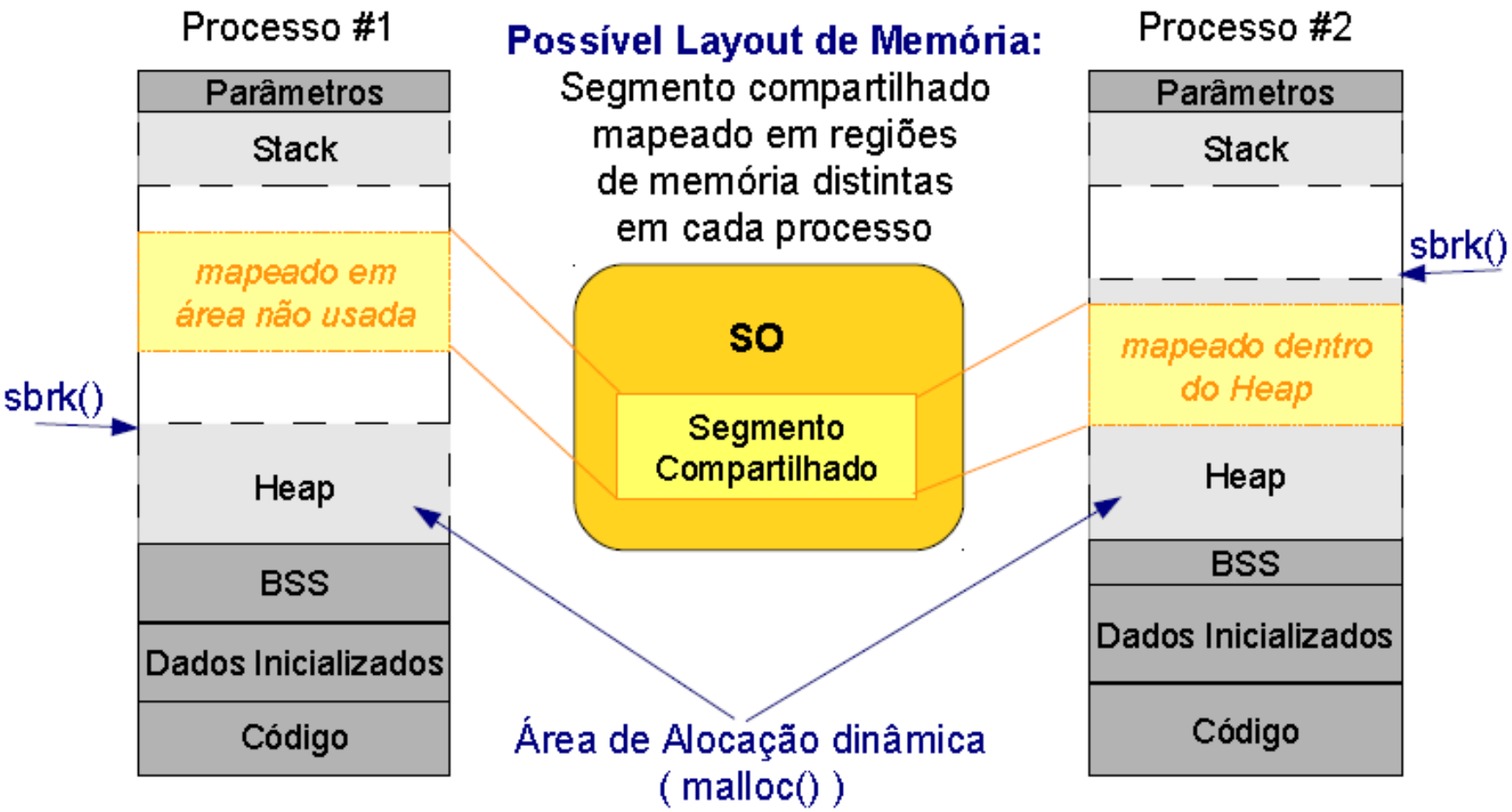
- Problema → quando o dado está pronto para o uso?
  - Não provê sincronização automática!
  - Tipicamente, usa-se outro mecanismo de IPC para coordenação do acesso ao segmento compartilhado
  * Ex.: semáforos, sinais, mensagens

# POSIX Shared Memory

- **Características gerais**:
  - Nomes globais, descritores locais
  - Persistência de sistema em memória RAM

- **Para se comunicarem, os processos precisam**:
  - Ter credencial compatível com as ACLs (permissões) configuradas para a SHM
  - Obter descritor para SHM:
  * A partir do nome; ou
  * Por heranca do processo pai
  - Definir um protocolo de acesso e formato de dados
  - Definir estratégia de sincronização → usualmente usa-se semáforos

# Mapeando um segmento de SHM



# Mapeando um segmento de SHM

- **Arquivos Mapeados em Memória** → suporte através do **subsistema de memória virtual**
  - Vincula região (páginas) de memória do processo com o conteúdo de um arquivo
  - Aplicação acessa a região onde o arquivo está mapeado normalmente (ex. ponteiros, arrays)
  - SO sincroniza conteúdo da região de memória com o arquivo automaticamente
  * **page-fault** dispara carga sob demanda do arquivo
- POSIX SHM faz uso desse suporte para sua implementação
  - Porém, sem um arquivo real em disco!
  - Segmento de SHM representado com arquivo virtual

# Visão geral  da API POSIX SHM

- Operações específicas da API:
  - **shm_open** → cria ou obtém descritor para shm
  - **shm_unlink** → remove segmento de shm
  - Ligar programa com a biblioteca POSIX RT (**-l rt**)

- Para o resto, utiliza operações gerais já existentes na API de arquivos:
  - **close** → libera descritor
  - **ftruncate** → usado para definir o tamanho
  - **mmap / munmap** → controlam mapeamento de no espaço de memória virtual do processo
  - **fstat** → consulta informações
  - **fchown**, **fchmod** → modifica dono e ACLs

# shm_open()

API

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

- Obtém um descritor para um segmento de memória compartilhada, opcionamente, criando-o caso não exista
  - **name** → semelhante a um nome absoluto de arquivo, deve iniciar com '/'. Ex.: “/nome”
  - **oflags** → o que será feito com a shm (O_RDONLY, O_RDWR)
    * Opcional: O_CREAT, O_EXCL, O_TRUNC
  - **mode** → permissões a serem atribuídas se O_CREAT, senao 0
- **Retorna**: descritor ou -1 em caso de erro
- Após criação, **ftruncate**(fd,size) usado na sequência para alterar o tamanho do segmento.

# mmap()

API

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

- Torna um segmento de SHM visível dentro do espaço de endereçamento virtual do processo
  - ***addr** → endereço onde deverá ser mapeado ou NULL para deixar o SO escolher
  - **len** → tamanho desejado da área mapeada
  - **prot** → PROT_READ | PROT_WRITE, deve ser compartível com o especificado no **shm_open()**
  - **flags** → **MAP_SHARED**
  - **fd** → descritor para shm
  - **offset** → início (dentro da shm) da área que será mapeada
- **Retorna**: endereço da SHM, ou MAP_FAILED

# munmap()

API

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

- Cancela um mapeamento previamente realizado com mmap()
  - ***addr** → onde inicia a área a ser “des-mapeada”
  - **len** → tamanho da área a ser “des-mapeada”

- É possivel “des-mapear” seletivamente partes da SHM mapeada pelo processo
  - Escolher valores para **addr** e **len** que não sejam o endereço inicial e tamanho total da área mapeada
  - Mas esse não é o caso mais comum

- **Retorna**: 0 se sucesso, ou -1 em caso de fracasso

# close()

API

```
int close(shmfd);
```

- Libera descritor do segmento de memória compatilhada
- Pode ser chamado logo depois o **mmap**()
- Raciocínio semelhante a arquivos
  - Limite de número de descritores por processo
  - Herança de descritores em fork()/exec()
- **Retorna**: 0 se sucesso, ou -1 em caso de erro

# shm_unlink()

API

```
int shm_unlink(name);
```

- Solicita a destruição de um segmento de memória compartilhada
  - **name** → nome segmento de SHM a ser removido

- Memória só é de fato liberada depois que todos os processos cancelarem os mapeamentos existentes (**munmap**())

- **Retorna**: 0 se sucesso, ou -1 em caso de erro

# Exemplo

```
struct data {
   int a; char b[10]; long c;
};
const int N = 4096*2;

int main() {
   int fd;
   struct data *addr;

   fd = shm_open("/xyz", O_RDWR | O_CREAT, 0600);
   ftruncate(fd, N);
   addr = mmap(NULL, N, PROT_READ|PROT_WRITE, MAP_SHARED, fd, 0);
   close(fd);
   addr->a = 5;
   addr->b[3] = 'C';
   addr->c = 0xffff;
   munmap(addr, N);
}
```

# Leituras complementares

- STEVENS, W.R. Advanced Programming in the UNIX Environment. 2nd. Ed., Addison Wesley, 2005.

- Man pages
  - Referentes a cada uma das funções abordadas
  - Overview de POSIX Shared Memory
    * man 7 shm_overview

- Livro: Advanced Linux Programming
  http://www.advancedlinuxprogramming.com/alp-folder

- Na web: System Software Unix IPC API
  http://jan.newmarch.name/ssw/ipc/unix.html

# Referências Bibliográficas

- Material originalmente elaborado por Prof. Cristiano Costa. Material autorizado e cedido pelo autor. Revisado e atualizado por Prof. Luciano Cavalheiro e posteriormente pelo Prof. João Tavares.