FATEC

CENTRO PAULA SOUZA
COMPETÊNCIA EM EDUCAÇÃO PÚBLICA PROFISSIONAL
Faculdade de Tecnologia Prof. Jessen Vidal
São José dos Campos

GOVERNO DO ESTADO DE SÃO PAULO

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Disciplina: Sistemas Operacionais I

Aula 08: Processos P3

Prof. Diogo Branquinho Ramos
diogo.branquinho@fatec.sp.gov.br
São José dos Campos - SP

Roteiro

- Importância da Comunicação
- Problema do Produtor-Consumidor
- Modelos fundamentais
  - Memória compartilhada
  - Passagem de mensagens

Comunicação entre processos – IPC

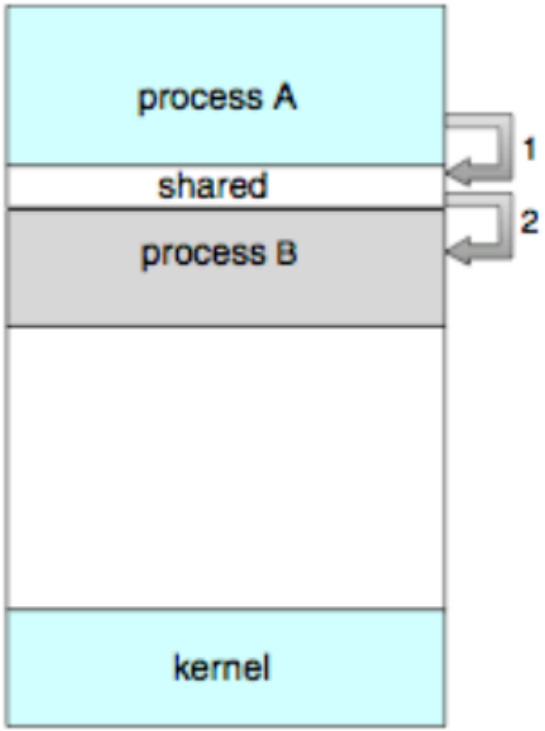
- **Processo independente**
  - Qualquer processo que não compartilhe dados com outro.
    - Não afeta e nem pode ser afetado por outro.
- **Características de processos cooperativos**
  - Podem afetar outros processos em execução.
  - Permitem compartilhamento de informações.
    - Ex.: vários usuários interessados no mesmo arquivo.
  - Usufruem de agilidade na computação.
    - Subdivisão de tarefas.
  - São modulares.
    - Funções em processos separados.

Problema do produtor-consumidor

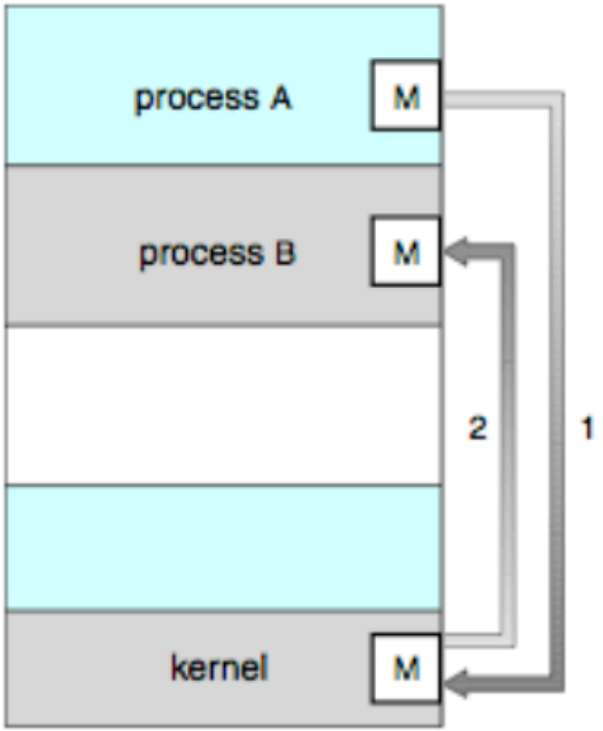
- **Processo produtor**
  - Gera informações para a aplicação como um todo.
- **Processo consumidor**
  - Usa as informações geradas pelo produtor.
- **Exemplo**
  - Ex.: página web consumida pelo browser.

Modelos fundamentais

Memória compartilhada



Passagem de mensagem



Memória compartilhada

- **Definição**
  - Área comum na memória que pode ser acessada por dois processos para comunicação entre eles.
- **Operações: *read/write***
- **Mais velocidade na comunicação**
  - Não há intermediário.
- **Menos seguro**
  - O formato e o local dos dados são determinados pelos processos envolvidos.
  - Os processos devem garantir a proteção de escrita.

Memória compartilhada: buffer vinculado

Produtor-consumidor

```
public class BoundedBuffer implements Buffer
{
    private static final int BUFFER_SIZE = 5;
    private int count; // number of items in the buffer
    private int in; // points to the next free position
    private int out; // points to the next full position
    private Object[] buffer;

    public BoundedBuffer() {
       // buffer is initially empty
       count = 0;
       in = 0;
       out = 0;

       buffer = new Object[BUFFER_SIZE];
    }

    // producers calls this method
    public void insert(Object item) {
       // Figure 3.16
    }

    // consumers calls this method
    public Object remove() {
       // Figure 3.17
    }
}
```

Memória compartilhada: buffer vinculado

Produtor-consumidor

```
public void insert(Object item) {
   while (count == BUFFER_SIZE)
        ; // do nothing -- no free buffers

   // add an item to the buffer
   ++count;
   buffer[in] = item;
   in = (in + 1) % BUFFER_SIZE;
}
```

Memória compartilhada: buffer vinculado

Produtor-consumidor

```
public Object remove() {
   Object item;

   while (count == 0)
        ; // do nothing -- nothing to consume

   // remove an item from the buffer
   --count;
   item = buffer[out];
   out = (out + 1) % BUFFER_SIZE;

   return item;
}
```

Passagem de mensagens

- **Proveem um protocolo para troca de mensagens**
  - Útil para quantidades menores de dados, pois não há conflitos.
  - Mais fácil de implementar.
  - Útil em ambiente distribuído: *hosts* conectados.
- **Operações: necessitam de um enlace**
  - *send*(mensagem)
  - *receive*(mensagem)
- **Tamanho da mensagem**
  - Fixo ou variável

Comunicação direta

- **Processos nomeados explicitamente**
  - ***send***(*P, mensagem*)
    - Envia uma mensagem ao processo P.
  - ***receive***(*Q, mensagem*)
    - Recebe uma mensagem do processo Q.
- **Propriedades do enlace de comunicação**
  - É estabelecido automaticamente em cada par de processos.
  - É associado a exatamente dois processos.
  - Entre cada par de processos, existe exatamente um enlace.

Comunicação indireta

- **Comunicação através de caixas de correio (ou portas)**
  - Cada porta tem um *id* exclusiva e os processos só podem se comunicar se compartilharem uma porta.
- **Primitivas**
  - ***send***(*A, mensagem*) – envia uma mensagem à porta A.
  - ***receive***(*A, mensagem*) – recebe uma mensagem da porta A.
- **Propriedades do enlace de comunicação**
  - Estabelecido se os processos compartilharem uma porta.
  - Pode ser associado a mais de dois processos.
  - Em cada par de processos, pode haver diversos enlaces diferentes: cada enlace associado a uma porta.

Comunicação indireta: dono da porta

Dono da porta

- **Pode ser o processo (P-)**
  - Porta faz parte do espaço de endereços do processo.
  - O P-proprietário (único) só pode receber através desta porta.
  - O P-usuário só pode enviar mensagens a essa porta.
  - A porta dura o tempo do processo.
- **Pode ser o SO**
  - Dura o tempo do SO e independe dos processos do usuário.
- **O SO deve permitir**
  - A criação de uma nova porta;
  - O envio e a recepção de mensagens por meio da porta; e
  - A destruição de uma porta.

Comunicação assíncrona/síncrona

- **Sem bloqueio é considerado assíncrono**
  - Envio sem bloqueio faz com que o emissor envie a mensagem e continue.
  - Recepção sem bloqueio faz com que o receptor receba uma mensagem válida ou nula.
- **Com bloqueio é considerado síncrono**
  - Envio com bloqueio deixa o emissor bloqueado até que a mensagem é recebida.
  - Recepção com bloqueio deixa o receptor bloqueado até que a uma mensagem esteja disponível.

Ocorre um "encontro" (rendezvous)

Buffers

- **Implementações**
  - Capacidade zero: o enlace não pode ter mensagens aguardando.
    - Emissor deve esperar pelo receptor.
  - Capacidade limitada: tamanho finito de n mensagens.
    - Emissor deve esperar se o enlace estiver cheio.
  - Capacidade ilimitada: tamanho infinito.
    - Emissor nunca espera.

Passagem de mensagens

Produtor-consumidor: caixa de correio

```
public class MessageQueue implements Channel
{
    private Vector queue;

    public MessageQueue() {
       queue = new Vector();
    }

    // This implements a nonblocking send
    public void send(Object item) {
       queue.addElement(item);
    }

    // This implements a nonblocking receive
    public Object receive() {
       if (queue.size() == 0)
          return null;
       else
          return queue.remove(0);
    }
}
```

Representa uma conexão aberta para uma entidade: *sockets, I/O,* arquivos...

Buffer ilimitado

Passagem de mensagens

Produtor-consumidor: caixa de correio

- **Produtor**

```
Channel mailBox;

while (true) {
    Date message = new Date();
    mailBox.send(message);
}
```

- **Consumidor**

```
Channel mailBox;

while (true) {
   Date message = (Date) mailBox.receive();
   if (message != null)
      // consume the message
}
```

Sem bloqueio, é preciso avaliar o conteúdo!