InstitutoFederal Norte de Minas Gerais

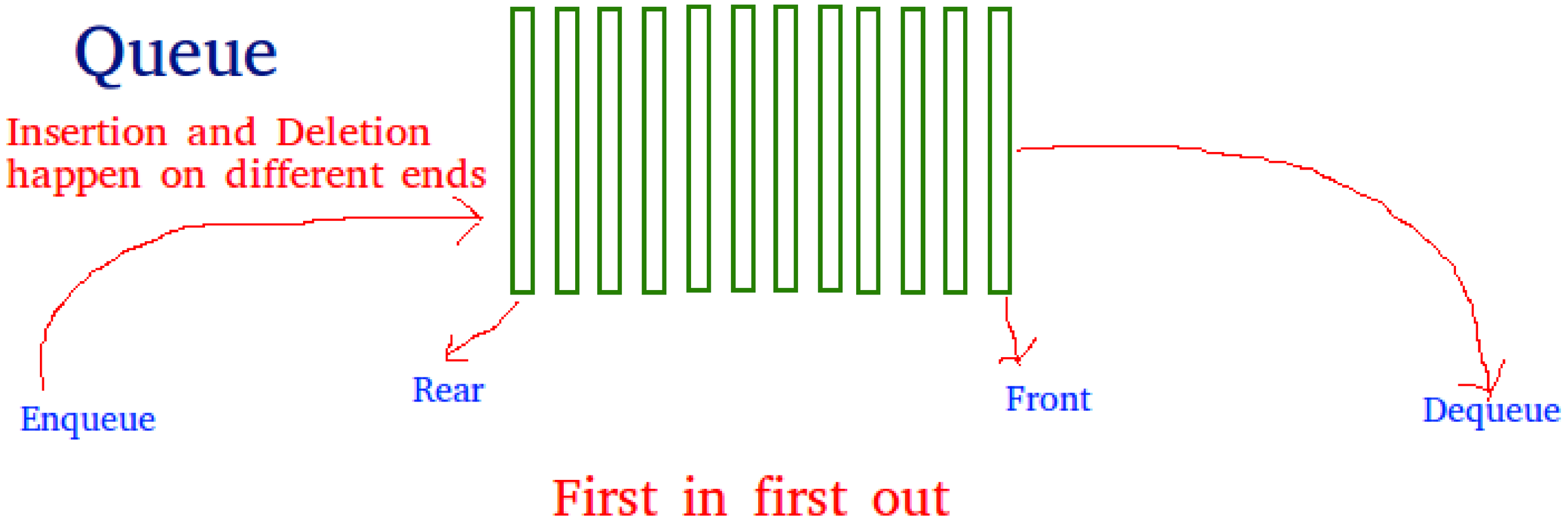
Queue as an interprocess mechanism

Sistemas Distribuidos

Docente:Adriano Antunes Prates

Discente:Arthur Soares Cardoso

Queue


Queue
Insertion and Deletion happen on different ends
Enqueue
Rear
Front
Dequeue
First in first out

Processos

Fonte: Silberschatz (2011. p. 58)

Voltar ao índice

Processo Independente

- Não utiliza comunicação interprocessual

Processo Cooperativo

- Sincroniza ações com outros processos podendo ser afetado por eles.

IPC

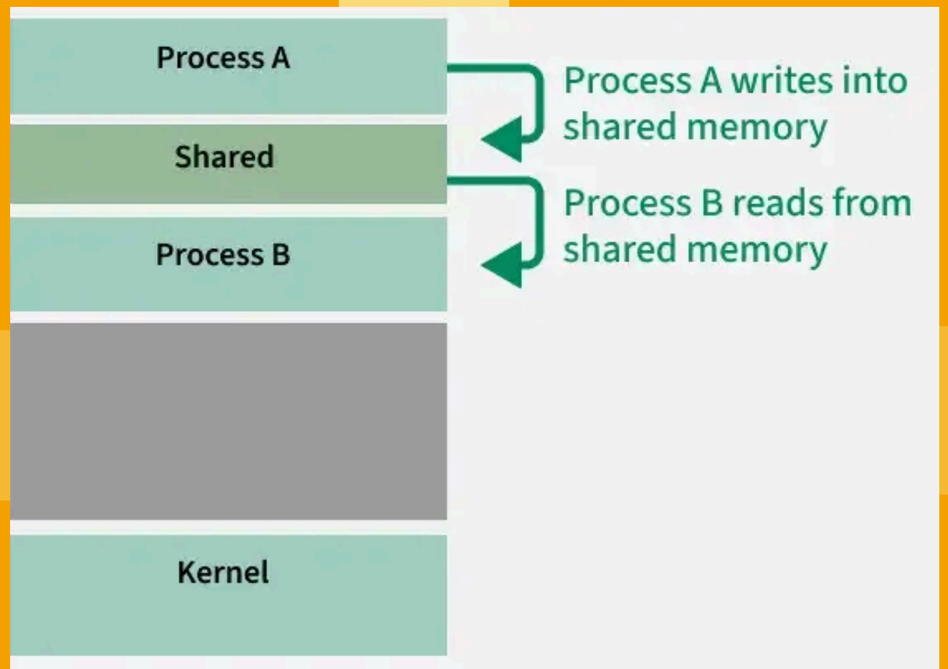
Interprocesss Communication

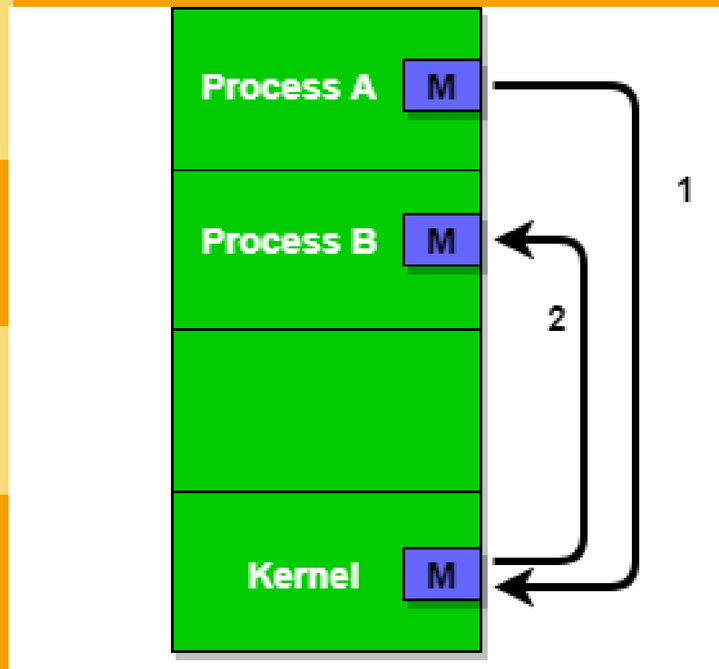
Memória Compartilhada

- Multiplos processos recebem acesso a mesma região de memoria.

Passagem de Mensagen

- Processos se comunicam enviando e recebendo mensagens usando metodos como Sockets ,Message Queues ou pipes.





IPC Using Message Queues

Armazenamento de mensagens:	uma fila armazena as mensagens até um processo receptor a leia.
Comunicação ordenada:	as queues garantem que as mensagens sejam entregues na ordem em que foram enviadas
Comunicação assíncrona:	Um processo de envio pode colocar uma mensagem na fila e o processo de recebimento pode recuperá-la mais tarde.
Desacoplamento de Processos:	os processos nao necessáriamente precisam estar cientes da existência ou estado um do outro.
Priorização:	É possivel tambem implementar filas que processam certas mensagens primeiro baseado no nivel de sua prioridade
Tratamento de erros:	Pode ser implementado dentro das filas mecanismos que garantam entrega das mensagens.

Thread-Safe

Utilizando locks

Python fornce locks a partir da threading.Lock ,protegem seções criticas.

```python
import threading

lock = threading.Lock()

def thread_safe_function():
    with lock:
        # Critical section of code
        # Access and modify shared resources safely
```

Estruturas de dados thread-safe

Python oferece estruturas thread-safe no modulo collections como queue ,Dequeu e Counter

```python
from collections import deque

thread_safe_deque = deque()

# Thread 1
thread_safe_deque.append(1)

# Thread 2
element = thread_safe_deque.pop()
```

Operações atômicas

para garantir atomicidade é preciso usar operações threading como Lock ou Rlock

```python
import threading

lock = threading.Lock()

def thread_safe_function():
    with lock:
        # Critical section of code
        # Access and modify shared resources safely
```

Evitando o estado compartilhado

Um beneficio do threading safety é minimizar ou eliminar o estado de mutavel sempre que possivel,não compartilhando dados e acesso a outras threads

Testando a segurança do thread

Os testes dos aplicativos multithread devem ser obrigatorios se o conceito de thread-safe for adotado,em cenários de simultainedade identificando possiveis bugs e potenciais condições de Corrida.

Diferenças

Queue

Retorna um processo da queue compartilhada usando pipe e metodos semaforo e lock

JoinableQueue

É uma subclass da classe queue adicionando os métodos :

task_done()

indica que o desenesfileiramento foi comcluido,esse metodo vai ser usado nas queue consumidoras indicadno que os processos foram completados.

join()

bloqueia enquanto todos os itens da fila forem pegos e processados.

Sentinelas

Voltar ao índice

O que são

É um valor ou um objeto,que vai ser colocado no final da fila indicando alguma coisa pra os consumidores.Normamelnte recebe o valor None.

Finaliza Processos

O sentinela vai indicar que não vai ter mais dados pra serem processados,pras varias threads que estão consumindo a fila.

Evita Deadlocks

Sem um sentinela pode acontecer de os consumidores esperarem infinitamente por dados resultando em um deadlock.

Obrigado Por Assistir!

DUVIDAS?