Semáforo

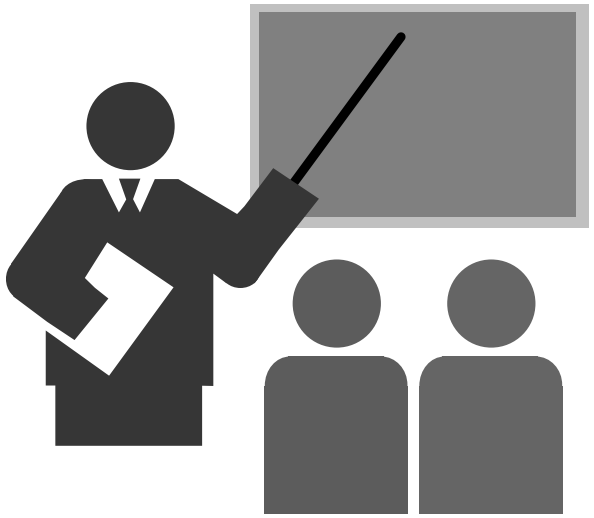
Volnys Borges Bernal
volnys@lsi.usp.br

Depto. de Eng. de Sistemas Eletrônicos
Escola Politécnica da USP

Semáforo

- **Sumário**
  - **Semáforo**
  - **Problema produtor-consumidor usando semáforo**
  - **Semáforo binário**
  - **Interface pthreads para semáforo**

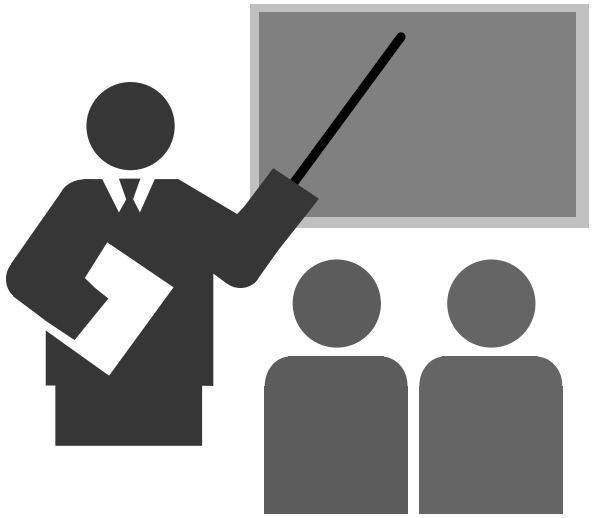
Semáforo



Semáforo

- **Método de sincronização que permite a contagem de recursos disponíveis**
- **Primitivas**
  - **Up(semaforo)**
    - Incrementa o contador do semáforo.
    - Se existirem entidades bloqueadas neste semáforo, uma delas é desbloqueada
  - **Down(semaforo)**
    - Decrementa o semáforo
    - Se o resultado for menor que zero, a entidade fica bloqueada neste semáforo.
  - **Init(semaforo,valor)**
- **Estas primitivas são garantidamente atômicas (indivisíveis)**
- **O semáforo deve ser iniciado com um valor inteiro, geralmente associado à quantidade de recursos disponíveis.**

Problema do produtor-consumidor com semáforo



Problema do produtor-consumidor

- **O problema do produtor consumidor possuir 3 necessidades de sincronização na espera por recursos:**

1. **Uso da região crítica**
   - Tanto produtor quanto consumidor pode, eventualmente, precisar esperar para entrar na região crítica
   - Quantidade de recursos disponíveis no início = 1
     - No início, somente 1 thread pode usar a região crítica
2. **Espera por itens na fila**
   - Consumidor espera por itens na fila quando fila está vazia
   - Quantidade de recursos disponíveis no início = 0
     - No início não existe nenhum item na fila
3. **Espera por slots livres**
   - Produtor espera por slots livres quando fila está cheia
   - Quantidade de recursos disponíveis no início = N (tamanho da fila)
     - No início existe N slots livres na fila

Problema do produtor-consumidor

- Cada necessidade de sincronização pode ser representada por um semáforo

- São necessários 3 semáforos:
  - Região crítica → bloqueia entidade (produtor ou consumidor) caso a região crítica esteja ocupada
  - Itens na fila → bloqueia consumidor caso fila esteja vazia
  - Slots livres → bloqueia produtor caso fila esteja cheia

- Quantidade de recursos disponíveis inicialmente:
  - Região crítica → 1 (permitido 1 thread por vez)
  - Itens na fila → 0 (inicialmente nenhum item na fila)
  - Slots livres → N (N=tamanho da fila)

Problema do produtor-consumidor

```
semaforo mutex = 1;
semafore itens = 0;
semafore slots = N;
```

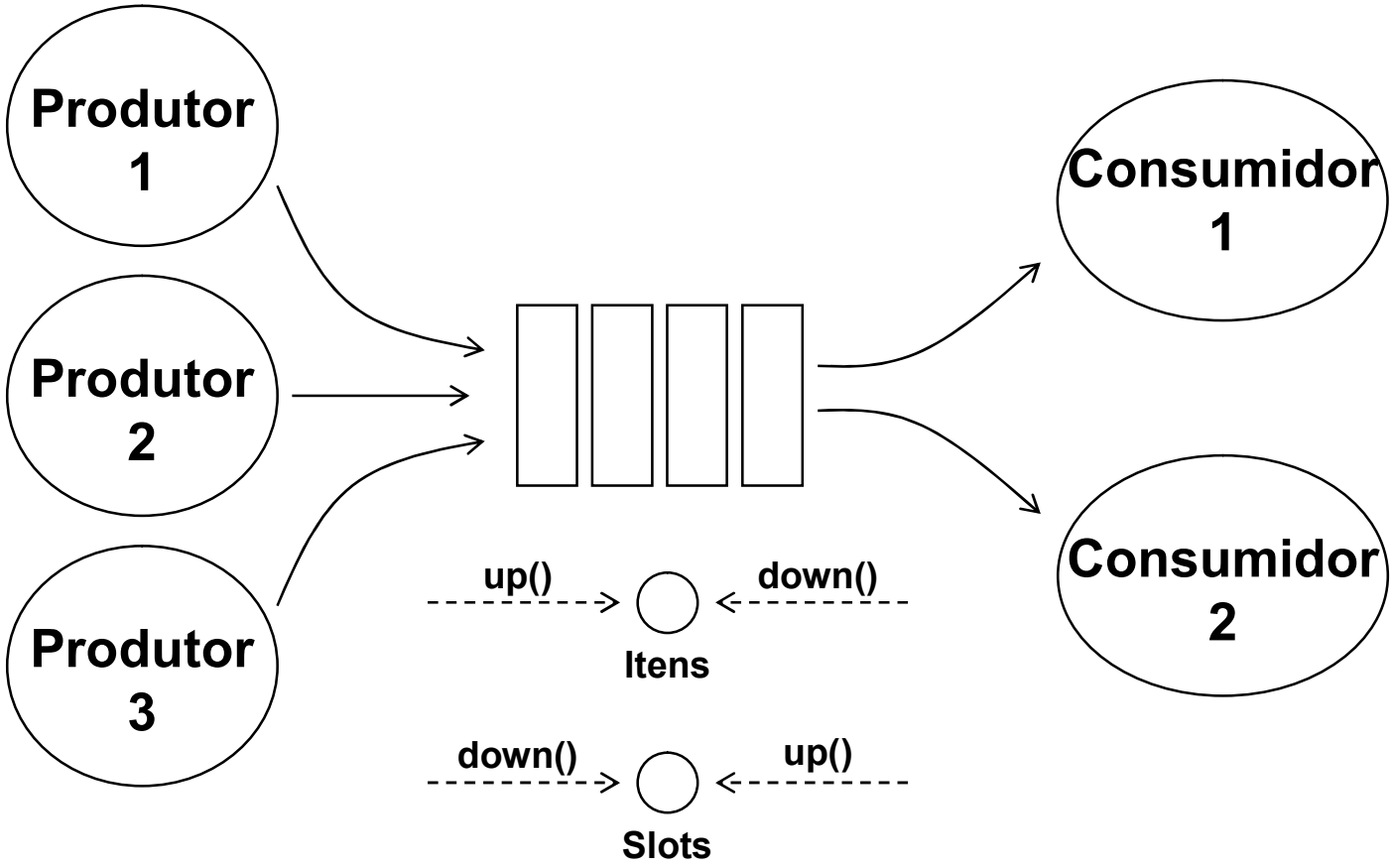
Produtor

```
Repetir
    Produzir(E);
    Down(slots);
    Down(mutex);
    InserirFila(F,E);
    Up(mutex);
    Up(itens);
```

Consumidor

```
Repetir
    Down(itens);
    Down(mutex);
    E=RetirarFila(F);
    Up(mutex);
    Up(slots);
    Processar(E);
```

Problema do produtor-consumidor



Exercício

(1) O que ocorre caso seja invertida a ordem de utilização dos semáforos no exercício anterior, ou seja:

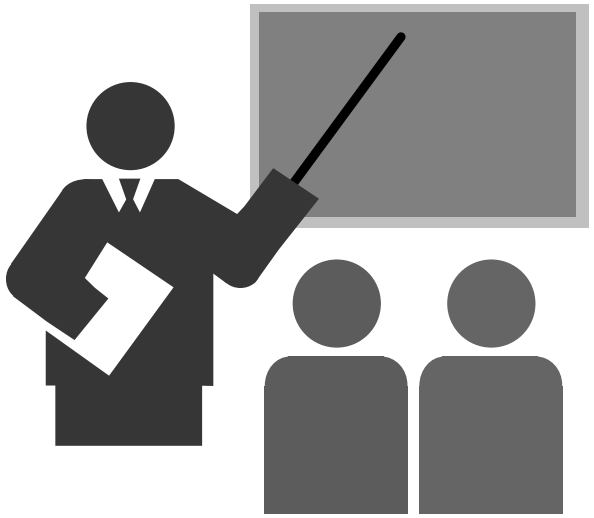
Produtor

```
Repetir
    Produzir(E);
    Down(mutex);
    Down(slots);
    InserirFila(F,E);
    Up(itens);
    Up(mutex);
```

Consumidor

```
Repetir
    Down(mutex);
    Down(itens);
    E=RetirarFila(F);
    Up(slots);
    Up(mutex);
    Processar(E);
```

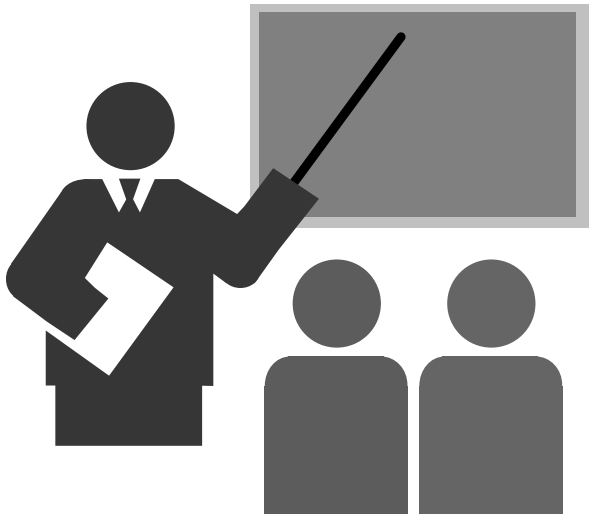
Semáforo Binário



Semáforo Binário

- Caso particular de semáforo no qual é iniciado com valor 1 e cujo valor nunca ultrapassa 1
- Pode ser utilizado para implementação de exclusão mútua:
  - lock() == down(semaforo_binário)
  - unlock() == up(semaforo_binário)

Interface pthreads para semáforo



Interface pthreads para semáforo

- **Tipos de dados**

Tipo	Descrição
`sem_t`	Representa o tipo de um semáforo.

Interface pthreads para semáforo

- **Primitivas**

Primitiva	Descrição
`sem_init`	Iniciação da variável de um semáforo.
`sem_wait`	Down. Decrementa o semáforo. Se o valor resultante for menor que zero a entidade de processamento é bloqueada.
`sem_trywait`	Variante de Down. Caso o valor do semáforo seja iqual ou menor que zero, retorna. Caso contrário, decrementa o semáforo.
`sem_post`	Up. Incrementa o semáforo. Se existirem entidades de processamento bloqueadas neste semáforo, uma delas é desbloqueada.
`sem_get_value`	Retorna o contador do semáforo.
`sem_destroy`	Destrói um semáforo.

Interface pthreads para semáforo

- **Sintaxe das primitivas**

```
#include <semaphore.h>

int  sem_init    (sem_t *sem, int pshared, unsigned int value)
int  sem_wait    (sem_t *sem)
int  sem_trywait (sem_t *sem)
int  sem_post    (sem_t *sem)
int  sem_getvalue(sem_t *sem, int *sval)
int  sem_destroy (sem_t *sem)
```

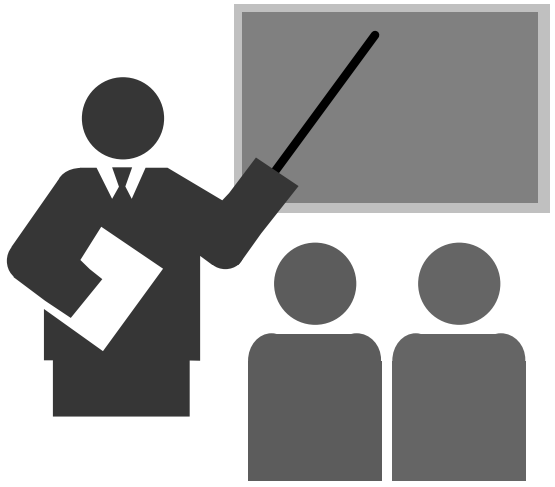
Interface pthreads para semáforo

- **Exemplo de uso:**

```
#include <semaphore.h>

...
sem_t slots;
...
status = sem_init(&slots,0,10);
...
status = sem_wait(&slots);
...
status = sem_post(&slots);
...
```

Referências Bibliográficas



Referências Bibliográficas

- ANDREW S. TANENBAUM; Sistemas Operacionais Modernos. Prentice-Hall.
  - Capítulo 2

- ANDREW S. TANENBAUM; Sistemas Operacionais. Prentice-Hall.
  - Capítulo 2