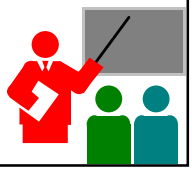
Semáforo, monitor e troca de mensagens

Volnys Borges Bernal
volnys@lsi.usp.br
http://www.lsi.usp.br/~volnys

Sumário

- **Mecanismos de sincronização e comunicação:**
  - **Semáforo**
    - Semáforo
    - Semáforo binário
    - Interface pthreads para semáforo
  - **Monitor**
  - **Troca de mensagens**

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.**
- **Normalmente, este valor está associado à quantidade de recursos disponíveis.**

Semáforo

- **Problema do produtor-consumidor**
  - **Necessita 3 semáforos**
    - 1 para garantir a exclusão mútua
    - 1 para bloquear os produtores
      - quando a fila estiver cheia (sem slots disponíveis)
    - 1 para bloquear os consumidores
      - quando a fila estiver vazia (sem itens na fila)

Semáforo

- **Problema do produtor-consumidor**

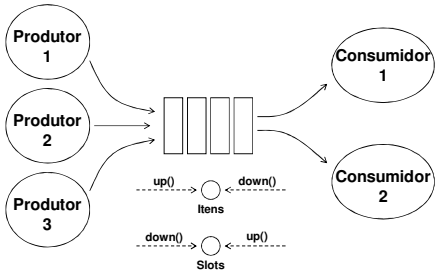
```
semaforo mutex = 1;
semafore slots = N;
semafore itens = 0;
```

```
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);
```

Semáforo

- Problema do produtor-consumidor

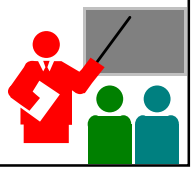


Exercício

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

```
Produtor                    Consumidor
   Repetir                     Repetir
      Produzir(E);                Down(mutex);
      Down(mutex);                Down(itens);
      Down(slots);                E=RetirarFila(F);
      InserirFila(F,E);           Up(slots);
      Up(itens);                  Up(mutex);
      Up(mutex);                  Processar(E);
```

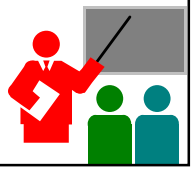

Semáforo Binário



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



Semáforo

- Pthreads
  - Tipos de dados

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

 13

Semáforo

- Pthreads
  - 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 igual 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.

 14

Semáforo

- Pthreads - sintaxe

```
#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)
```

 15

Semáforo

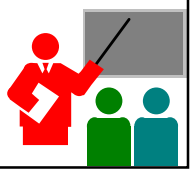
- Exemplo de uso

```
#include <semaphore.h>

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

 16

Monitor



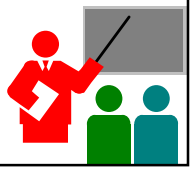
 17

Monitor

- Conjunto de funções agrupadas em um pacote especial, como uma caixa preta.
- As estruturas internas manipuladas pela funções do monitor não são visíveis.
- Existe um mecanismo de sincronização de alto nível que garante que somente uma entidade (processo ou thread) pode adentrar no monitor em um determinado momento.

 18

Troca de mensagem



Troca de mensagem

- **Mecanismo muito utilizado para sincronização e comunicação entre processos**
- **Primitivas**
  - **Send(destinatário, mensagme)**
  - **Receive(remetente, mensagem)**
- **Tipos de primitivas**
  - **Síncrona**
  - **Assíncrona**

Troca de mensagem

- **Tipos de primitivas**
  - **Síncrona**
    - Send() bloqueante
      - Entidade é bloqueada até a entidade parceira ativar o Receive()
    - Receive() bloqueante
      - Entidade é bloqueado até a entidade parceira ativar Send()
    - Não necessita de buffers temporários
  - **Assíncrona**
    - Send() não é bloqueante
    - Receive() é bloqueante
    - Necessita de gerenciamento de buffers temporários