Sistemas Operacionais
Aula Prática 6

Prof. Samuel Ferraz

UFMS

17/10/14

Tópicos

1. Monitores
   - Introdução
   - Definição e exemplos
2. Variáveis de Condição
   - Introdução e Definição
   - Exemplo 1
   - Exemplo 2
   - Exemplo 3

Introdução

- Em aulas anteriores, vimos a importância do uso de semáforos;
- Também estamos (cansados) de ver a solução do problema do produtor-consumidor com semáforos;

Introdução

- Em aulas anteriores, vimos a importância do uso de semáforos;
- Também estamos (cansados) de ver a solução do problema do produtor-consumidor com semáforos;

```
#define N 100
Semaforo mutex = 1;
Semaforo empty = N;
Sematoro full = 0
```

Introdução

- Em aulas anteriores, vimos a importância do uso de semáforos;
- Também estamos (cansados) de ver a solução do problema do produtor-consumido com semáforos;

```
void produtor() {
     int item;
     while(TRUE) {
          item = produce_item();
          down(&empty);
          down(&mutex);
          insert_item_into_buffer(item);
          up(&mutex);
          up(&full);
     }
}
```

Introdução

- Em aulas anteriores, vimos a importância do uso de semáforos;
- Também estamos (cansados) de ver a solução do problema do produtor-consumido com semáforos;

```
void consumidor() {
     int item;
     while(TRUE) {
          down(&full);
          down(&mutex);
          item = remove_item_from_buffer(item);
          up(&mutex);
          up(&empty);
     }
}
```

Introdução

- Ok, então ainda há algo de novo para aprender no tópico de sincronização de threads?

Introdução

- Ok, então ainda há algo de novo para aprender no tópico de sincronização de threads?
- Não...

Introdução

- Ok, então ainda há algo de novo para aprender no tópico de sincronização de threads?
- Não...
- e SIM!

Introdução

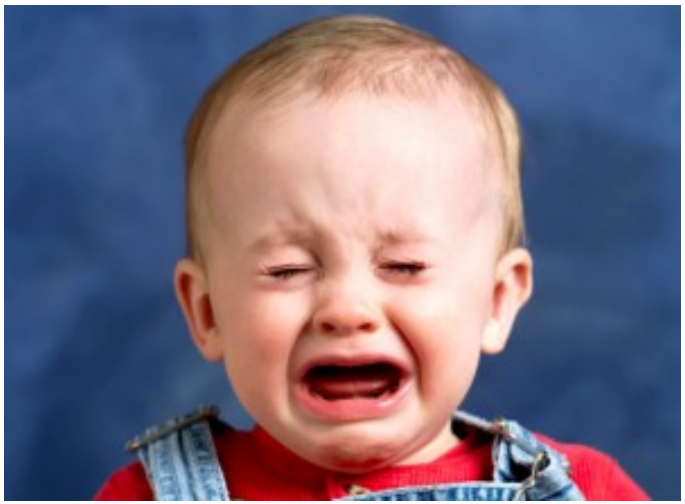
- Ok, então ainda há algo de novo para aprender no tópico de sincronização de threads?
- Não...
- e SIM!
- Semáforos resolvem boa parte dos nossos problemas de sincronização, mas seu uso em determinadas situações é limitado e pode ser perigoso.
- Adivinha quem está de volta para ilustrar essa situação?

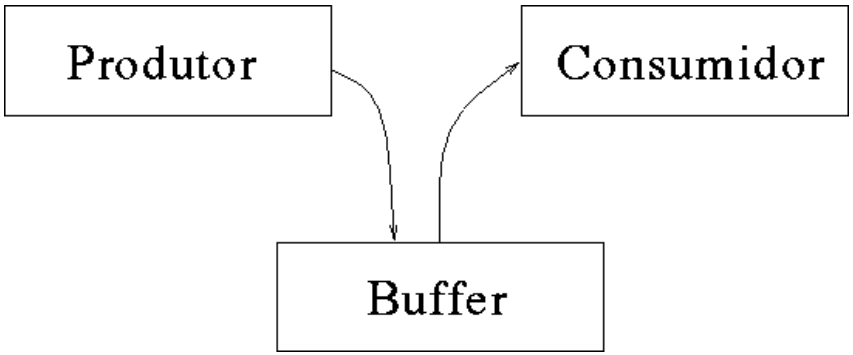
Introdução

- Adivinha quem está de volta para ilustrar essa situação?

Introdução

- Adivinha quem está de volta para ilustrar essa situação?





Possíveis erros ao usar semáforos

- Inverter a ordem correta das chamadas.

```
void produtor() {
     int item;
     while(TRUE) {
          down(&mutex);
          down(&full);
          item = remove_item_from_buffer(item);
          up(&mutex);
          up(&empty);
     }
}
```

Possíveis erros ao usar semáforos

- Esquecer de fazer uma das chamadas.

```
void produtor() {
     int item;
     while(TRUE) {
          down(&full);
          down(&mutex);
          item = remove_item_from_buffer(item);
          up(&mutex);
     }
}
```

Possíveis erros ao usar semáforos

- Se confundir na hora de fazer uma das chamadas.

```
void produtor() {
     int item;
     while(TRUE) {
          down(&full);
          down(&mutex);
          item = remove_item_from_buffer(item);
          up(&mutex);
          up(&full);
     }
}
```

Monitores

- Por essas e outras razões, existem os *monitores*;
- Abstração de alto nível, fornecida por uma linguagem de programação;
- Fazem com que determinados métodos sejam executados com exclusão mútua;
- Exemplo em Java.

Introdução

- Em determinadas situações, pode ser interessante mandar uma thread dormir e depois acordá-la quando uma condição for satisfeita;
- Isso acontece em um monitor através do uso implícito de variáveis de condição;
- Como em C/C++ não há o conceito de monitores, quando queremos que uma thread durma ou acorde apenas quando determinados eventos acontecem, utilizamos variáveis de condição.

Exemplo 1

- Exemplo: implementação do *pthread_join*;
  - Uma *thread* pai quer esperar a *thread* filha fazer algo, para então continuar o seu trabalho;

Exemplo 1

- Exemplo: implementação do *pthread_join*;
  - Uma *thread* pai quer esperar a *thread* filha fazer algo, para então continuar o seu trabalho;
- Intuição: *join_1.c*;

Exemplo 1

- Exemplo: implementação do *pthread_join*;
  - Uma *thread* pai quer esperar a *thread* filha fazer algo, para então continuar o seu trabalho;
- Intuição: *join_1.c*;
- Solução inicial: *join_2.c*.

Exemplo 1

- Exemplo: implementação do *pthread_join*;
  - Uma *thread* pai quer esperar a *thread* filha fazer algo, para então continuar o seu trabalho;
- Intuição: *join_1.c*;
- Solução inicial: *join_2.c*.
  - Correta?

Exemplo 1

- Exemplo: implementação do *pthread_join*;
  - Uma *thread* pai quer esperar a *thread* filha fazer algo, para então continuar o seu trabalho;
- Intuição: *join_1.c*;
- Solução inicial: *join_2.c*.
  - Correta?
  - Recomendável?

Exemplo 1

- Exemplo: implementação do *pthread_join*;
  - Uma *thread* pai quer esperar a *thread* filha fazer algo, para então continuar o seu trabalho;
- Intuição: *join_1.c*;
- Solução inicial: *join_2.c*.
  - Correta?
  - Recomendável?
- Próxima solução: *join_3.c*.

Exemplo 1

- Exemplo: implementação do *pthread_join*;
  - Uma *thread* pai quer esperar a *thread* filha fazer algo, para então continuar o seu trabalho;
- Intuição: *join_1.c*;
- Solução inicial: *join_2.c*.
  - Correta?
  - Recomendável?
- Próxima solução: *join_3.c*.
  - Correta?

Exemplo 1

- Exemplo: implementação do *pthread_join*;
  - Uma *thread* pai quer esperar a *thread* filha fazer algo, para então continuar o seu trabalho;
- Intuição: *join_1.c*;
- Solução inicial: *join_2.c*.
  - Correta?
  - Recomendável?
- Próxima solução: *join_3.c*.
  - Correta?
  - Recomendável?

Exemplo 1

- Exemplo: implementação do *pthread_join*;
  - Uma *thread* pai quer esperar a *thread* filha fazer algo, para então continuar o seu trabalho;
- Intuição: *join_1.c*;
- Solução inicial: *join_2.c*.
  - Correta?
  - Recomendável?
- Próxima solução: *join_3.c*.
  - Correta?
  - Recomendável?
  - Por qual motivo usar *while* e não *if*?

Exemplo 1

- Exemplo: implementação do *pthread_join*;
  - Uma *thread* pai quer esperar a *thread* filha fazer algo, para então continuar o seu trabalho;

Exemplo 1

- Exemplo: implementação do *pthread_join*;
  - Uma *thread* pai quer esperar a *thread* filha fazer algo, para então continuar o seu trabalho;
- Próxima solução: *join_4.c*.

Exemplo 1

- Exemplo: implementação do *pthread_join*;
  - Uma *thread* pai quer esperar a *thread* filha fazer algo, para então continuar o seu trabalho;
- Próxima solução: *join_4.c*.
  - Correta?

Exemplo 1

- Exemplo: implementação do *pthread_join*;
  - Uma *thread* pai quer esperar a *thread* filha fazer algo, para então continuar o seu trabalho;
- Próxima solução: *join_4.c*.
  - Correta?
- Próxima solução: *join_5.c*.

Exemplo 1

- Exemplo: implementação do *pthread_join*;
  - Uma *thread* pai quer esperar a *thread* filha fazer algo, para então continuar o seu trabalho;
- Próxima solução: *join_4.c*.
  - Correta?
- Próxima solução: *join_5.c*.
  - Correta?

Exemplo 2

- Adivinha quem está de volta?

Exemplo 2

- Adivinha quem está de volta?

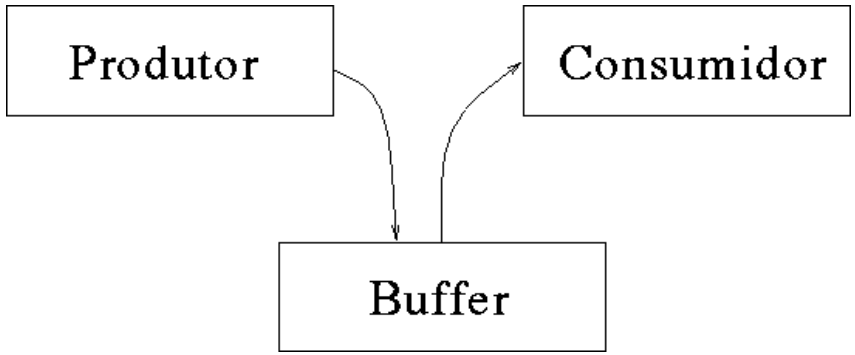
Exemplo 2

- Adivinha quem está de volta?

- Não, ainda não...

Exemplo 2

- Adivinha quem está de volta?



Exemplo 2

- Vejamos algumas implementações.
- Intuição: *pc_1.c*;

Exemplo 2

- Vejamos algumas implementações.
- Intuição: *pc_1.c*;
- Solução inicial: *pc_2.c*.
  - Correta?

Exemplo 2

- Vejamos algumas implementações.
- Intuição: *pc_1.c*;
- Solução inicial: *pc_2.c*.
  - Correta?
  - Apenas para uma *thread* produtora e uma consumidora.

Exemplo 2

- Vejamos algumas implementações.
- Intuição: *pc_1.c*;
- Solução inicial: *pc_2.c*.
  - Correta?
  - Apenas para uma *thread* produtora e uma consumidora.
  - Exemplo: duas *threads* consumidoras e uma produtora;

Exemplo 2

- Vejamos algumas implementações.
- Intuição: *pc_1.c*;
- Solução inicial: *pc_2.c*.
  - Correta?
  - Apenas para uma *thread* produtora e uma consumidora.
  - Exemplo: duas *threads* consumidoras e uma produtora;
  - Sequência:
    - consumidora1 tenta consumir e dorme;
    - produtora1 produz e gera sinal;
    - consumidora2 executa e consome (mas não estava dormindo);
    - consumidora1 executa e consome (não há elemento!).

Exemplo 2

- Vejamos algumas implementações.
- Próxima solução: *pc_3.c*.
  - Correta?

Exemplo 2

- Vejamos algumas implementações.
- Próxima solução: *pc_3.c*.
  - Correta?
  - Apenas para uma *thread* produtora e uma consumidora.

Exemplo 2

- Vejamos algumas implementações.
- Próxima solução: *pc_3.c*.
  - Correta?
  - Apenas para uma *thread* produtora e uma consumidora.
  - Exemplo: duas *threads* consumidoras e uma produtora;

Exemplo 2

- Vejamos algumas implementações.
- Próxima solução: *pc_3.c*.
  - Correta?
  - Apenas para uma *thread* produtora e uma consumidora.
  - Exemplo: duas *threads* consumidoras e uma produtora;
  - Sequência:
    - consumidora1 tenta consumir e dorme;
    - consumidora2 tenta consumir e dorme;
    - produtora1 produz, gera sinal e dorme;
    - consumidora1 acorda, consome e gera sinal;
    - consumidora2 acorda, não consome e dorme;
    - As três *threads* estão dormindo. Por qual motivo?

Exemplo 2

- Vejamos algumas implementações.
- Próxima solução: *pc_4.c*.
  - Correta?

Exemplo 2

- Vejamos algumas implementações.
- Próxima solução: *pc_4.c*.
  - Correta?
  - Sim! Mas para buffer de tamanho 1;
  - Sacada: variáveis de condição específicas!

Exemplo 2

- Vejamos algumas implementações.
- Próxima solução: *pc_4.c*.
  - Correta?
  - Sim! Mas para buffer de tamanho 1;
  - Sacada: variáveis de condição específicas!
- Próxima solução: *pc_5.c*.
  - Correta?

Exemplo 2

- Vejamos algumas implementações.
- Próxima solução: *pc_4.c*.
  - Correta?
  - Sim! Mas para buffer de tamanho 1;
  - Sacada: variáveis de condição específicas!
- Próxima solução: *pc_5.c*.
  - Correta?
  - Sim! Sem restrições.

Exemplo 3

- Alocação de memória: *malloc.c*;

Exemplo 3

- Alocação de memória: *malloc.c*;
- Solução: *pthread_cond_broadcast*.