Sincronização e Comunicação

Eduardo Ferreira dos Santos

Ciência da Computação
Centro Universitário de Brasília – UniCEUB

Maio, 2017

Sumário

1. Concorrência

2. Sistemas Multiprogramáveis
   - Memória compartilhada
   - Troca de mensagens

UniCEUB
Centro Universitário de Brasília

1. Concorrência

2. Sistemas Multiprogramáveis
   - Memória compartilhada
   - Troca de mensagens

Concorrência

- Paradigma: controlar/restringir o acesso ao recurso em determinado espaço de tempo;
- O controle de acesso aos recursos é realizado através de eventos;
- Eventos inesperados pode causar um desvio inesperado no fluxo de execução.

- Definição [Chagas, 2016]:
  1. O programa perde o uso do processador;
  2. O programa retorna para continuar o processamento;
  3. O estado do programa deve ser idêntico ao do momento em que foi interrompido.
- O programa continua a execução exatemente na instrução seguinte.

Troca de Contexto

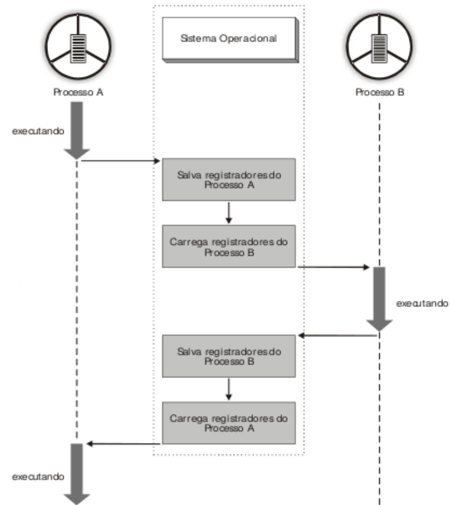

Figura 1.1: Troca de Contexto [Favacho, 2009]

Sincronia

O modelo é dito síncrono porque as saídas do sistema podem ser vistas como sincronizadas com as suas entradas. [FARINES and MELO, 2000]

Uma das características mais importantes encontrada nos modelos síncronos é a rejeição do não determinismo. [FARINES and MELO, 2000, p.105]

Determinismo Para cada estado do programa, existe somente uma possibilidade para a função de transição (próximo estado);

Não determinismo Podem existir várias escolhas para o próximo estado em qualquer ponto.

Estados dos processos

Figura 1.2: Estados dos processos [Chagas, 2016]

Interrupção e Exceção

- Concorrência: controlar/restringir o acesso ao recurso;
- O controle de acesso aos recursos é realizado através de eventos;
- Eventos inesperados pode causar um desvio inesperado no fluxo de execução.

Interrupção Realizada por algum evento externo ao programa, independente da instrução.

- Podem ser geradas por eventos assíncronos;
- Até várias vezes ao mesmo tempo.

Exceção Erro na instrução de algum programa. Ex.: falha de segmentação (*segfault*).

- Sempre gerada por um evento síncrono;
- Depende de instrução do próprio programa;
- Somente um evento de cada vez.

Interrupção

- Geradas por eventos assíncronos;
- Diversos dispositivos podem informar ao processador que estão prontos.

Figura 1.3: Tratamento de Interrupção [Chagas, 2016]

Exceção

- Gerada por um evento síncrono;
- Considerando a mesma entrada, a exceção ocorrerá sempre na mesma instrução.

Figura 1.4: Tratamento de Exceção [Chagas, 2016]

1 Concorrência

2 Sistemas Multiprogramáveis
- Memória compartilhada
- Troca de mensagens

Programação concorrente

- Na programação concorrente existe mais de uma tarefa sendo executada ao mesmo tempo. Ex.: Fatorial
- No caso de múltiplas tarefas é necessário haver comunicação entre elas.

Memória compartilhada As tarefas compartilham área de memória;

Troca de mensagens Sinais trocados entre processos.

Comunicação entre processos

- Gerência de recursos de memória compartilhada: condição de corrida
  - Exclusão mútua;
  - Semáforo;
  - Monitor.
- Comunicação por troca de mensagens: deadlocks
  - Leitura assíncrona;
  - Método *rendezvous*.

1 Concorrência

2 Sistemas Multiprogramáveis
- Memória compartilhada
- Troca de mensagens

Condição de corrida

Situações onde dois ou mais processos estão acessando dados compartilhados, e o resultado final do processamento depende de quem roda quando.

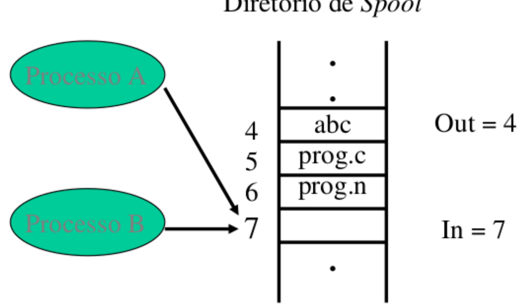

Figura 2.1: Exemplo da condição de corrida

Solução para condição de corrida

- Uma boa solução para a condição de corrida requer quatro condições [Favacho, 2009]:
  1. Dois ou mais processos não podem estar simultaneamente dentro de suas regiões críticas correspondentes;
  2. Nenhuma consideração pode ser feita a respeito da velocidade relativa dos processos, ou a respeito do número de processadores disponíveis no sistema;
  3. Nenhum processo que esteja executando fora de sua região crítica pode bloquear a execução de outro processo;
  4. Nenhum processo pode ser obrigado a esperar indefinidamente para entrar em sua região crítica.

Exclusão mútua [Chagas, 2016]

- Solução: impedir que mais de um processo acesse o dado ao mesmo tempo.
- Deve ser executada somente quando um dos processos estiver acessando o recurso compartilhado;
- A parte do código onde o acesso ao recurso é feito é chamada de região crítica.

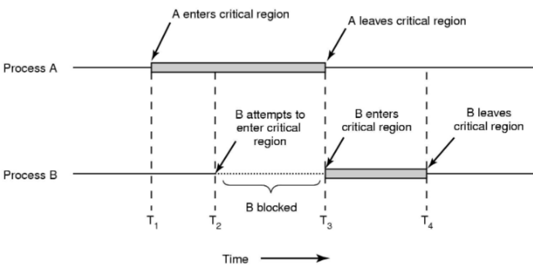

Figura 2.2: Região crítica

Soluções para exclusão mutua I

Inibição das interrupções Inibir as interrupções de cada processo logo após o ingresso na região crítica, habilitando-as novamente após deixá-las.

- Desabilitar interrupções deve ser uma atribuição do kernel;
- Interferir no kernel pode não ser uma boa ideia.

Variáveis de travamento (locks) Utilização de variável única compartilhada (*lock*) que pode assumir 0 ou 1.

- Se dois processos chegam ao mesmo tempo?
- Condição de corrida.

Soluções para exclusão mutua II

Chaveamento obrigatório Utiliza a variável inteira turn [Tanenbaum and Machado Filho, 1995].

- A variável turn indica a vez de quem é de entrar na região crítica;
- Se um dos processos for mais lento que o outro requer a solução estritamente alternada;
- Espera ocupada: teste contínuo do valor esperando por uma mudança.

Listing 1: a

```
while (TRUE) {
  while (turn!=0)    /* çlao */
  critical_region();
  turn = 1;
  non_critical_region();
}
```

Listing 2: b

```
while (TRUE) {
  while (turn!=1)    /* çlao */
  critical_region();
  turn = 0;
  non_critical_region();
}
```

Problema do produtor-consumidor

- Dois processos compartilham um *buffer* de tamanho fixo;
- Um põe a informação dentro do *buffer*: produtor;
- Outro retira a informação do *buffer*: consumidor;
- Problema: produtor quer colocar um item no *buffer*, mas já está cheio;
- Solução: colocar o produtor para dormir (*sleep*) e só acordar quando o consumidor remover um ou mais itens;
- Grande possibilidade de gerar condição de corrida: perda do envio de sinal para acordar (*wakeup*) quando o processo ainda não está dormindo.

Semáforos [Favacho, 2009]

- Baseado em um tipo de variável que possui dois estados: UP e DOWN.
  1. O semáforo fica associado a um recurso compartilhado;
  2. Se o valor da variável semáforo for diferente de zero, nenhum processo está utilizando o recurso; caso contrário,o processo fica impedido do acesso;
  3. Sempre que deseja entrar em sua região crítica, o processo executa uma instrução DOWN;
  4. Se o semáforo for maior que 0, este é decrementado de 1, e o processo que solicitou a operação pode executar sua região crítica;
  5. Entretanto, se uma instrução DOWN é executada em um semáforo cujo valor seja igual a 0, o processo que solicitou a operação ficará no estado de espera;

Semáforos (cont.) [Favacho, 2009]

6. Além disso, o processo que está acessando o recurso, ao sair de sua região crítica, executa uma instrução UP, incrementando o semáforo de 1 e liberando o acesso ao recurso;
7. A verificação do valor do semáforo, a modificação do seu valor e, eventualmente a colocação do processo para dormir são operações atômicas;
8. Operações atômicas são únicas e indivisíveis;
9. Os semáforos aplicados ao problema da exclusão mútua são chamados de mutex (*mutual exclusion*) ou binários, por apenas assumirem os valores 0 e 1.

Implementação [Favacho, 2009]

```
#define N 100
typedef int semaphore
semaphore mutex = 1; /* controla ãregio ícrtica */
semaphore empty = N; /* controla çõposies vazias */
semaphore full = 0; /* controla çõposies ocupadas */
```

```
void producer(void ) {
  while (TRUE) {
    item = produce_item ();
    down (&empty);
    down (& mutex );
    insert_item(item ); /* R_critica
      */
    up(&mutex );
    up(& full );
  }
}
```

```
void consumer(void ) {
  while (TRUE) {
    down (& full );
    down (& mutex );
    item = remove_item (); /* R_crí
        tica */
    up(&mutex );
    up(&empty );
    consume_item(item );
  }
}
```

1 Concorrência

2 Sistemas Multiprogramáveis
- Memória compartilhada
- Troca de mensagens

Comunicação

- Somente entre dois processos;
- Os processos precisam ter a execução sincronizada.

Figura 2.3: Troca de mensagens [Chagas, 2016]

Classificação [Chagas, 2016]

Direta Comunicação entre dois processos, cujos nomes devem ser explicitados;

Indireta Utiliza uma área (ou *buffer*) onde as mensagens devem ser colocadas.

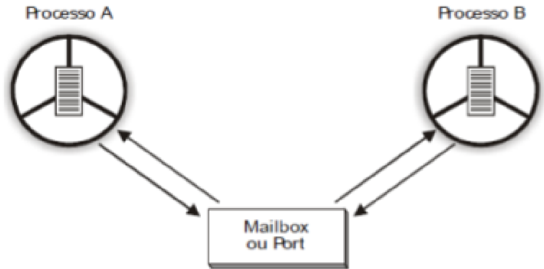

Figura 2.4: Troca de mensagens [Chagas, 2016]

Chagas, F. (2016).
Notas de aula do Prof. Fernando Chagas.

FARINES, J. M. and MELO, R. (2000).
Sistemas de Tempo Real, volume 1.
IME-USP.

Favacho, A. (2009).
Notas de aula da Profa. Aletéia Favacho.

Tanenbaum, A. S. and Machado Filho, N. (1995).
Sistemas operacionais modernos, volume 3.
Prentice-Hall.

UniCEUB
Centro Universitário de Brasília

OBRIGADO!!!
PERGUNTAS???