Processos com Memória Compartilhada

Programação Concorrente e Paralela – 2013.2

August 28, 2013

Execução de um programa

- sequência de ações *atômicas* ou indivisíveis
  - instruções de máquina
- ações fazem processo transitar entre *estados*
  - variáveis globais
  - pilha de execução

Execução de um Programa

- organização

processo: espaço de endereçamento isolado

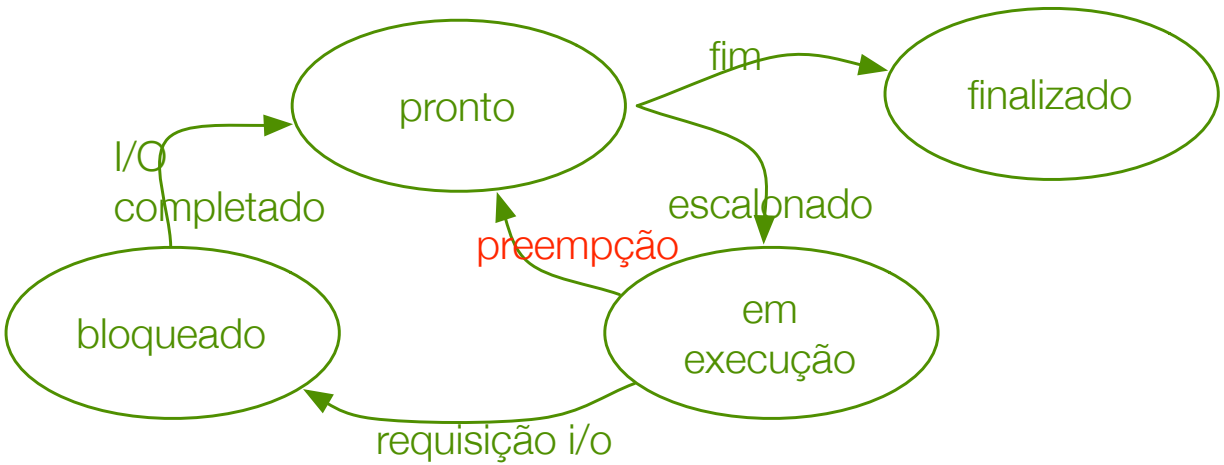
- variáveis globais, arquivos, signals, etc

thread: pilha de execução

- system-level e library-level
- acesso a memmória compartilhada

muitas vezes usaremos os termos de forma intercambiável

- escalonamento



Eventos e Estados

- uma execução específica pode ser vista como uma *história* (sequência de eventos? sequência de estados?)

  $s_0 \rightarrow s_1 \rightarrow \ldots \rightarrow s_n$
- *eventos* são instantâneos
- *estados* vão mudando com ocorrência de eventos
  - programas concorrentes: número enorme de possíveis histórias
  - sincronização: corte de histórias indesejáveis

Seções Críticas

- conceito clássico em sistemas de memória compartilhada
- seção crítica: trecho de código que apenas uma thread pode estar executando em determinado momento

```
public class Counter {
  private long value;
  private Lock lock;
  public long getAndIncrement() {
    int temp = value;
    value = value + 1;
    return temp;
  }
}
```

Garantia de Exclusão Mútua

- protocolos de entrada e saída de regiões críticas

```
public class Counter {
  private long value;
  private Lock lock;
  public long getAndIncrement() {
   lock.lock();
   try {
    int temp = value;
    value = value + 1;
   } finally {
     lock.unlock();
   }
   return temp;
  }
}
```

- se executado em threads 0 e 1: $RC_0 \rightarrow RC_1$ ou $RC_1 \rightarrow RC_0$

Exclusão Mútua – Propriedades

- garantia da exclusão mútua
- ausência de deadlock
- entrada em algum momento (ausência de starvation)
- ausência de esperas desnecessárias
  - thread não espera quando não há competição pela RC

Exclusão Mútua – Propriedades

- garantia da exclusão mútua
  - correção
- ausência de deadlock
  - safety
- entrada em algum momento (ausência de starvation)
  - liveness
- ausência de esperas desnecessárias
  - nível de concorrência

Classes de Propriedades

- *safety*
  - programa nunca entra em estado ruim
- *liveness*
  - programa em algum momento entra em estado bom

Soluções de exclusão mútua utilizando espera ocupada

- estudo clássico
- alguma utilidade com múltiplos processadores
- algoritmos interessantes (!)
  - complexidade: caso com 2 threads é o mais inteligível

  - Peterson
  - Filtros
  - Padaria

Soluções de exclusão mútua utilizando espera ocupada

- estudo clássico
- alguma utilidade com múltiplos processadores
- algoritmos interessantes (!)
  - complexidade: caso com 2 threads é o mais inteligível