Programação Concorrente e Paralela

Exclusão Mútua

2016

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

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
- idéias básicas (que não funcionam)
  - variável `vez`
  - variável `chave`

Soluções de exclusão mútua utilizando espera ocupada

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

Exclusão Mútua – Propriedades

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

Exclusão mútua: algoritmo de Peterson

```
public void lock() {
 flag[i] = true;
 victim  = i;
 while (flag[j] && victim == i) {};
}
public void unlock() {
 flag[i] = false;
}
```

Art of Multiprocessor Programming

Exclusão Mútua

```
public void lock(0) {
  flag[A] = true;
  victim  = A;
  while (flag[B] && victim == A)
     {};
```

```
public void lock(0) {
  flag[B] = true;
  victim  = B;
  while (flag[A] && victim == B)
     {};
```

funciona?

- vamos supor, por contradição, que ambos estão na seção crítica

... funciona? prova por contradição

- do código:
  - $write_A(flag[A]=true) \rightarrow write_A(victim=A) \rightarrow read_A(flag[B]) \rightarrow read_A(victim) \rightarrow CS_A$
  - $write_B(flag[B]=true) \rightarrow write_B(victim=B) \rightarrow read_B(flag[A]) \rightarrow read_B(victim) \rightarrow CS_B$
- vamos assumir:
  - $write_B(victim=B) \rightarrow write_A(victim=A)$
- como A entrou na RC:
  - $write_A(victim=A) \rightarrow read_A(flag[B] == false)$

- mas então
  - $write_B(flag[B]=true) \rightarrow write_B(victim=B) \rightarrow write_A(victim=A) \rightarrow read_A(flag[B] == false)$

```
public void lock(1) {
  flag[A] = true;
  victim  = A;
  while (flag[B] && victim == A)  {};
```

Exclusão Mútua: problemas!

- soluções baseadas em variáveis dependem de ordenação que pode se perder na compilação...

```
public void lock(0) {
 victim  = 0;  [2]
 flag[0] = true; [3]
 while (flag[1] && victim == 0)  [4]
    {};
```

```
public void lock(1) {
 victim  = 1; [1]
 flag[1] = true; [5]
 while (flag[0] && victim == 1) [6]
   {};
```

Test & Set

Test & Set