Implementações de Mutex em software

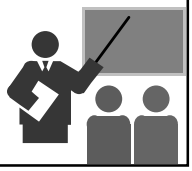
Volnys Borges Bernal
volnys@lsi.usp.br

Departamento de Sistemas Eletrônicos
Escola Politécnica da USP

Tópicos

- Alternativas para implementação de Exclusão Mútua (mutex) em software:
  - Alternância obrigatória
  - Solução de Peterson

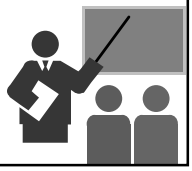
Implementação de Mutex em software



Implementação de Mutex

- As alternativas para implementação de Exclusão Mútua em software não funcionam
- Exemplos de soluções em software:
  - Alternância obrigatória
  - Solução de Peterson

Mutex implementado em software: Alternância Obrigatória



Alternância Obrigatória

- Objetivo
  - Implementação de exclusão mútua
- Descrição
  - Alterna o acesso à região crítica entre duas entidades
  - Totalmente em software
  - Utiliza espera ociosa
- Desvantagem:
  - Viola requisito #3 (Nenhuma entidade fora da região crítica pode ter a exclusividade desta)
  - Válida para somente duas entidades (processos/threads)
  - Utiliza espera ociosa

Alternância Obrigatória

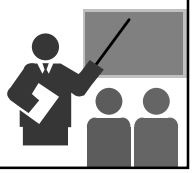
Entidade 0:

```
...
while (TRUE)
    {
    realiza outras atividades

    // lock()
    while (turn!=0);
    ...
    região_critica
    ...
    // unlock()
    turn=1;
    }
```

Entidade 1:

```
...
while (TRUE)
    {
    realiza outras atividades

    // lock()
    while (turn!=1);
    ...
    região_critica
    ...
    // unlock()
    turn=0;
    }
```


Mutex implementado em software: Solução de Peterson



Solução de Peterson

- **Objetivo**
  - **Implementação de exclusão mútua**
- **Autoria**
  - **Publicada por G. L. Peterson em 1981**
  - **Baseada em uma solução do matemático holandês T. Dekker**
- **Descrição**
  - **Totalmente em software**
  - **Utiliza espera ociosa**
- **Desvantagem**
  - **Exemplo mostrado a seguir é válido somente para 2 entidades**

Solução de Peterson

```
int turn;                       // Duas entidades:
int interested[2];              //     Entidade 0
                                //     Entidade 1
void lock(int me)
    {
    int me;
    int other;

    other = (me + 1) mod 2;
    interested[me] = TRUE;
    turn = me;
    while (turn != me && interested[other] == TRUE);
    }

void unlock(int me)
    {
    interested[me] = FALSE;
    }
```