Trincos Lógicos: Implementações

Algorítmicas

Hardware

Sistema Operativo

Trincos lógicos: Propriedades

- Exclusão mútua
- Progresso (liveness)
  - Ausência de interblocagem (deadlock)
  - Ausência de míngua (starvation)

Proposta #1

```
int trinco = ABERTO;

fechar() {
  while (trinco == FECHADO) ;
  trinco = FECHADO;
}

abrir() {
  trinco = ABERTO;
}
```

Qual a propriedade que não é garantida?

Proposta #2

```
int vez = 1;

t1_fechar() {
  while (vez == 2) ;
}

t1_abrir() {
  vez = 2;
}
```

```
t2_fechar() {
  while (vez == 1) ;
}

t2_abrir() {
  vez = 1;
}
```

Qual a propriedade que não é garantida?

Proposta #3

```
int t1_quer_entrar = FALSE, t2_quer_entrar = FALSE;

t1_fechar() {
  while (t2_quer_entrar == TRUE) ;
  t1_quer_entrar = TRUE;
}

t1_abrir() {
  t1_quer_entrar = FALSE;
}

/* t2 -> simetrico */
```

Qual a propriedade que não é garantida?

Proposta #4

```
int t1_quer_entrar = FALSE, t2_quer_entrar = FALSE;

t1_fechar() {
  t1_quer_entrar = TRUE;
  while (t2_quer_entrar == TRUE) ;
}

t1_abrir() {
  t1_quer_entrar = FALSE;
}

/* t2 -> simetrico */
```

Porque motivo é garantida a exclusão mútua?

Qual a propriedade que não é garantida?

Algoritmo de Peterson

```
int trinco_t1 = ABERTO;
int trinco_t2 = ABERTO;
int tar_prio = 1;

t1_fechar () {
  A1: trinco_t1 = FECHADO;
  B1: tar_prio = 2;
  C1: while (trinco_t2 == FECHADO && tar_prio == 2);
}

t1_abrir () {trinco_t1 = ABERTO;}

t2_fechar () {
  A1: trinco_t2 = FECHADO;
  B1: tar_prio = 1;
  C1: while (trinco_t1 == FECHADO && tar_prio == 1);
}

t2_abrir () {trinco_t2 = ABERTO;}
```

T1 pretende aceder à secção crítica

T2 é mais prioritária

Algoritmo de Lamport (Bakery)

```
int senha[N];   // Inicializado a 0
int escolha[N];   // Inicializado a FALSE

Fechar (int i) {
  int j;
  escolha[i] = TRUE;
  senha [i] = 1 + maxn(senha);
  escolha[i] = FALSE;

  for (j=0; j<N; j++) {
    if (j==i) continue;
    while (escolha[j]) ;
    while (senha [j] && (senha [j] < senha [i]) ||
           (senha [i] == senha [j] && j < i));
  }
}

Abrir (int i) {senha [i] = 0;}
```

- senha contém o número da senha atribuído à tarefa
- escolha indica se a tarefa está a pretender aceder à secção crítica

•Pi indica que está a escolher a senha
•Escolhe uma senha maior que todas as outras
•Anuncia que escolheu já a senha

•Pi verifica se tem a menor senha de todos os Pj

•Se Pj estiver a escolher uma senha, espera que termine

•Neste ponto, Pj ou já escolheu uma senha, ou ainda não escolheu
  •Se escolheu, Pi vai ver se é menor que a sua
  •Se não escolheu, vai ver a de Pi e escolher uma senha maior

•Se a senha de Pi for menor, Pi entra
•Se as senhas forem iguais, entra o que tiver o menor identificador

Soluções Algorítmicas

- Conclusão:
  - Complexas → Latência
  - Só contemplam **espera activa**
- Solução:
  - Introduzir instruções hardware para facilitar a solução

Soluções com Suporte do Hardware

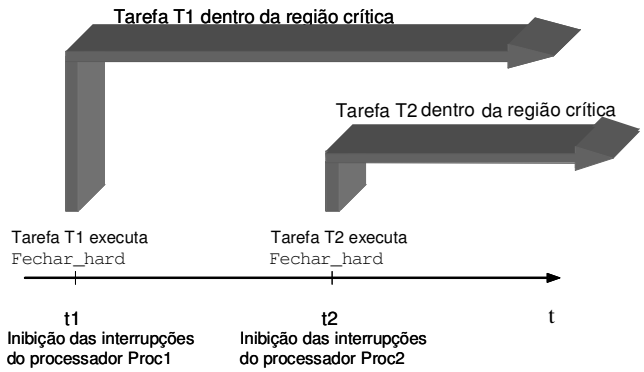
- Abrir( ) e Fechar( ) usam instruções especiais oferecidas pelos processadores:
  - Inibição de interrupções
  - Exchange (xchg no Intel)
  - Test-and-set (cmpxchg no Intel)

Exclusão Mútua com Inibição de Interrupções

```
int mascara;

Fechar_hard () {
  mascara = mascarar_int ();
}

Abrir_hard () {
  repoe_int (mascara);
}
```

- Este mecanismo só deve ser utilizado dentro do sistema operativo em secções críticas de muito curta duração
  - inibição das interrupções impede que se executem serviços de sistema (I/O, etc)
  - se o programa se esquecer de chamar abrir(), as interrupções ficam inibidas e o sistema fica parado
- Não funciona em multiprocessadores

Inibição das Interrupções não Funciona em Multiprocessadores



Test-and-Set

```
ABERTO  EQU 0              ; ABERTO equivale ao valor 0
FECHADO EQU 1              ; FECHADO equivale ao valor 1

Fechar_hard:
L1:     MOV AX, 0          ; AX é inicializado com o valor 0 (ABERTO)
        BTS trinco, AX     ; testa se trinco e AX são iguais
        JC L1              ; a carry flag fica com o valor inicial do trinco
                           ; se carry flag ficou a 1, trinco estava FECHADO
                           ; implica voltar a L1 e tentar de novo
                           ; se carry flag ficou a 0, trinco estava ABERTO
        RET                ; trinco fica a 1 (FECHADO)

Abrir_hard:
        MOV     AX, ABERTO
        MOV     trinco, AX
        RET
```

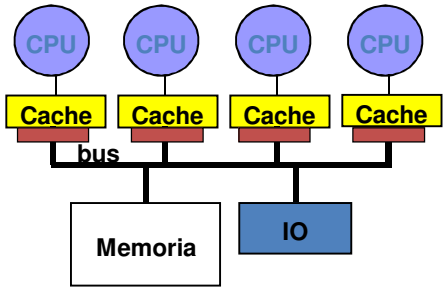

XCHG

```
ABERTO  EQU     0       ; ABERTO equivale ao valor 0
FECHADO EQU     1       ; FECHADO equivale ao valor 1

trinco DW       0       ; trinco com valor inicial ABERTO

Fechar_hard:
        MOV  AX, FECHADO
L1:     XCHG AX, trinco
        CMP  AX, ABERTO ; compara o valor de AX com 0
        JNZ  L1         ; se AX é diferente de zero volta para L1
        RET

Abrir_hard:
        MOV AX, ABERTO
        MOV trinco, AX
        RET
```

- a instrução XCHG assegura que o trinco fica fechado:
  - pois escreveu o valor um (que estava no registo AX) na posição de memória trinco (bus trancado)
- qualquer outra tarefa que posteriormente execute a instrução cmp AX,ABERTO vai detectar que o valor do trinco é um (está fechado)

XCHG em Multiprocessadores

	P1	P2
Instante 1	P1 inicia exchange e tranca o bus	
Instante 2	P1 completa exchange e tranca a secção crítica	P2 tenta fazer exchange mas bloqueia-se a tentar obter o bus
Instante 3	P1 entra secção crítica	P2 verifica que o trinco está trancado e fica em espera activa



Soluções com Suporte do Hardware

- Conclusão:
  - Oferecem os mecanismos básicos para a implementação da exclusão mútua, mas...
  - Algumas não podem ser usadas directamente por programas em modo utilizador
    - e.g., inibição de interrupções
  - Outras só contemplam espera activa
    - e.g., exchange, test-and-set

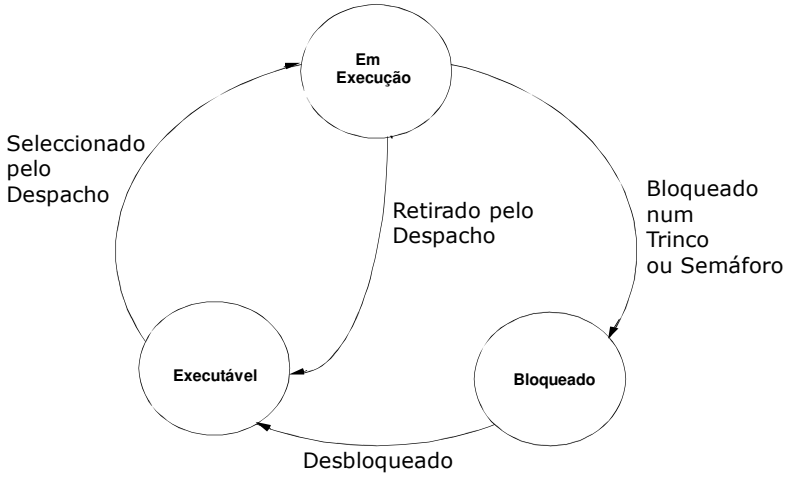
Soluções com Suporte do SO

- Primitivas de sincronização são chamadas ao SO:
  - Software trap (interrupção SW)
  - Comutação para modo núcleo
  - Estruturas de dados e código de sincronização pertence ao núcleo
  - Usa o suporte de hardware (exch. / test-and-set)
- Veremos duas primitivas de sincronização:
  - Trincos
  - Semáforos

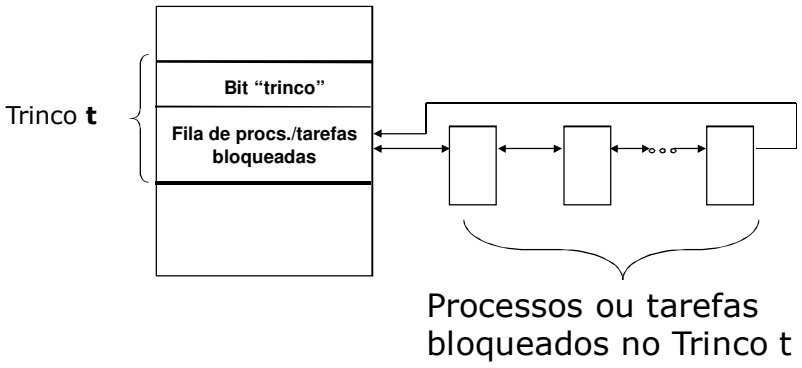
Soluções com Suporte do SO

- Vantagens (intuitivas):
  - se SO souber quais processos estão à espera de secção crítica, nem sequer lhes dá tempo de processador
  - evita-se a espera activa!
- Soluções hardware podem ser usadas, mas pelo código do SO:
  - dentro das chamadas sistema que suportam o abrir(), fechar(), etc.

Diagrama de Estado dos Processos / Tarefas



Estruturas de Dados Associadas aos Trincos



Funções do trinco (*mutex*)

```
t.var=ABERTO; // trinco inicialmente ABERTO
t.numTarefasBloqueadas = 0;

Fechar_mutex (trinco_t t) {

    if (t.var == FECHADO){
        numTarefasBloqueadas++;

        bloqueia_tarefa();
        }
    else {
        t.var = FECHADO;

        }
}
Abrir_mutex (trinco_t t) {

    if (numTarefasBloqueadas > 0)
        {desbloqueia_tarefa();
         numTarefasBloqueadas--;
        }
    else t.var = ABERTO;
```

- Retirar tarefa de execução
- Salvaguardar o seu contexto
- Marcar o seu estado como bloqueada
- Colocar a estrutura de dados que descreve a tarefa na fila de espera associada ao trinco
- Invocar o algoritmo de escalonamento

- Existem tarefas bloqueadas
- Marcar o estado de uma delas como "executável"
- Retirar a estrutura de dados que descreve a tarefa da fila de espera associada ao trinco

- Este programa concorrente está errado!
- É necessário assegurar que variáveis partilhadas são acedidas em exclusão mútua

Funções do trinco (*mutex*)

```
trinco_t = t;
t.var=ABERTO; // trinco inicialmente ABERTO
t.numTarefasBloqueadas = 0;

Fechar_mutex (trinco_t t) {
    Fechar_hard(t);
    if (t.var == FECHADO){
        numTarefasBloqueadas++;
        Abrir_hard(t);
        bloqueia_tarefa();
        }
    else {
        t.var = FECHADO;
        Abrir_hard(t);
        }
}
Abrir_mutex (trinco_t t) {
    Fechar_hard(t);
    if (numTarefasBloqueadas > 0)
        {desbloqueia_tarefa();
         numTarefasBloqueadas--;
        }
    else t.var = ABERTO;
    Abrir_hard(t)
}
```

- É necessário assegurar exclusão mútua no acesso aos atributos do trinco

Qual a diferença entre a exclusão mútua no acesso aos atributos do trinco e a exclusão mútua que o trinco assegura?

Espera activa com e sem suporte do núcleo