6. Sincronização

Cooperação entre processos…

Sincronia:

- Ordem de operações
- Espera pela conclusão
- Recurso compartilhado e controle de acesso exclusivo ou simultâneo

Sincronização baseada em:

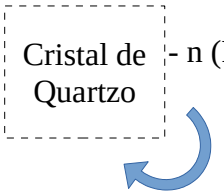
- Tempo real (absoluto) => Relógio físico
- Ordem relativa => Relógio lógico

Alerta! Uso da hora local ao invés da hora do servidor

```
UPDATE …
SET Usuario = :Usuario,
    DataHora = :DataHora
WHERE …
```

É possível sincronizar todos os relógios em um sistema distribuído?

Clock/Timer



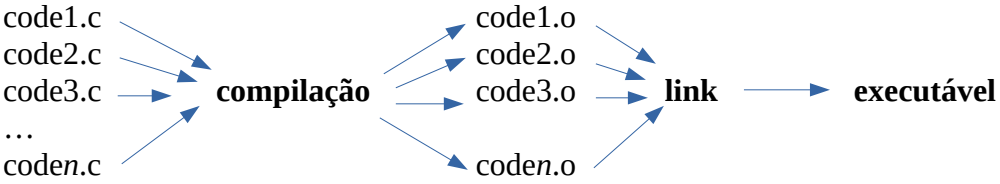
- n (Registrador)

Oscilações => n – 1
12, 11, 10, …, 0 => Ciclo => Interrupção => Procedimento => RAM/CMOS (time++)

Mesmo com relógios em perfeita sincronia, com o passar do tempo a sincronização é perdida devido a diferenças nas oscilações dos cristais de quartzo.

A hora UTC também sofre ajustes anuais (cerca de 1s) para se manter em sincronia com a rotação da Terra e seu posicionamento na órbita solar.

Desafio!



code*k*.c é recompilado somente se code*k*.o for mais antigo, ou seja, data(code*k*.o) < data(code*k*.c).

Você consegue pensar em uma solução que independa do relógio da máquina?

Relógio lógico de Lamport

Conceitos iniciais…

`a → b` O evento `a` ocorre antes do evento `b`.

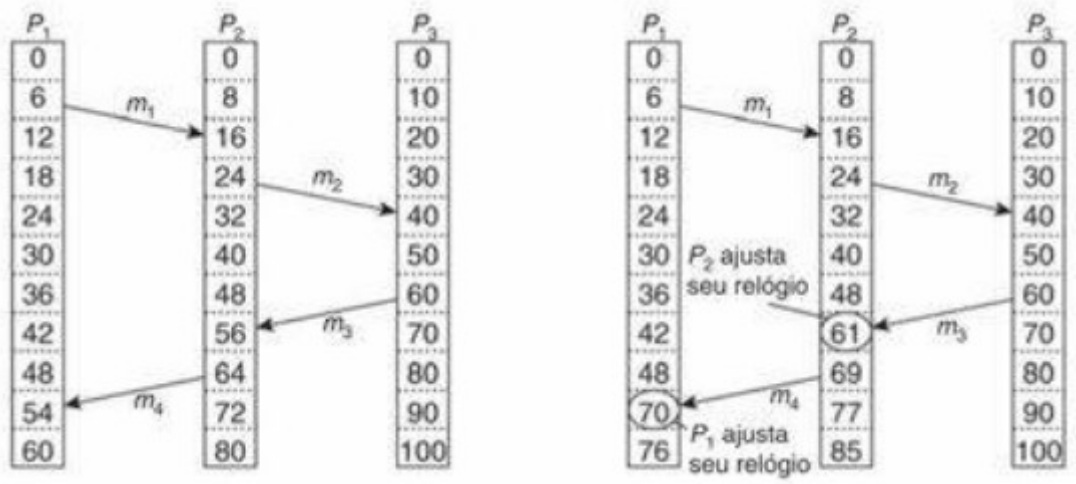
Exemplo: `a` e `b` são "disparados" por um mesmo processo, sendo que este dispara `a` e somente depois dispara `b`.

Exemplo: `a` representa um evento de uma mensagem enviada por um processo e `b` representa um evento que corresponde a essa mensagem sendo recebida por outro processo.

`a → b` e `b → c` implica `a → c`, ou seja, relação transitiva.

`a` e `b` são concorrentes, ou seja, não se pode afirmar que `a → b` ou `b → a`, pois ambos ocorrem em processos diferentes que não trocam mensagens.

Objetivo: Garantir que se `a → b`, então `C(a) < C(b)`. Onde `C` (clock) é o instante em que ocorre o evento. Não necessariamente ligado ao relógio real. Tipicamente um contador local.



Exclusão mútua

1. Soluções baseadas em tokens

O processo que detém o token pode, se desejar, acessar o recurso ou simplesmente passar o token adiante.

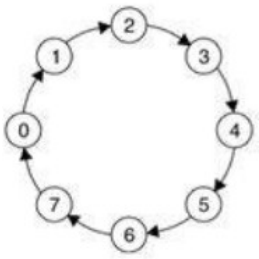
Vantagens:

- Evita starvation (inanição)
- Evita deadlock (impasse)

Desvantagem:

Se o processo que detém o token falha, um "complicado" procedimento para disponibilizar um novo token deve ter início.

Exemplo: Algoritmo Token Ring



2. Soluções baseadas em solicitação de permissão

Algoritmo centralizado: O processo que deseja acesso a um recurso compartilhado deve requisitar ao coordenador (servidor responsável pelo uso do recurso) permissão de acesso.

Desvantagens:

- Single point of failure
- O coordenador pode se tornar um gargalo de desempenho

Algoritmo descentralizado: O processo que deseja acesso a um recurso compartilhado deve enviar um pedido de acesso a todos os coordenadores responsáveis pelo recurso. Supondo n coordenadores, o requisitante espera apenas por uma resposta majoritária m: $m > n / 2$.

Desvantagem:

Em caso de alta concorrência, pode haver problema de inanição.

Algoritmos de eleição

Dado um conjunto de nós em um sistema distribuído, como eleger um líder?

Estratégia comumente adotada: Selecionar o processo com o maior ID.

Observação: Todos os processos devem concordar com o líder escolhido, mesmo quando alguns nós estão inativos.

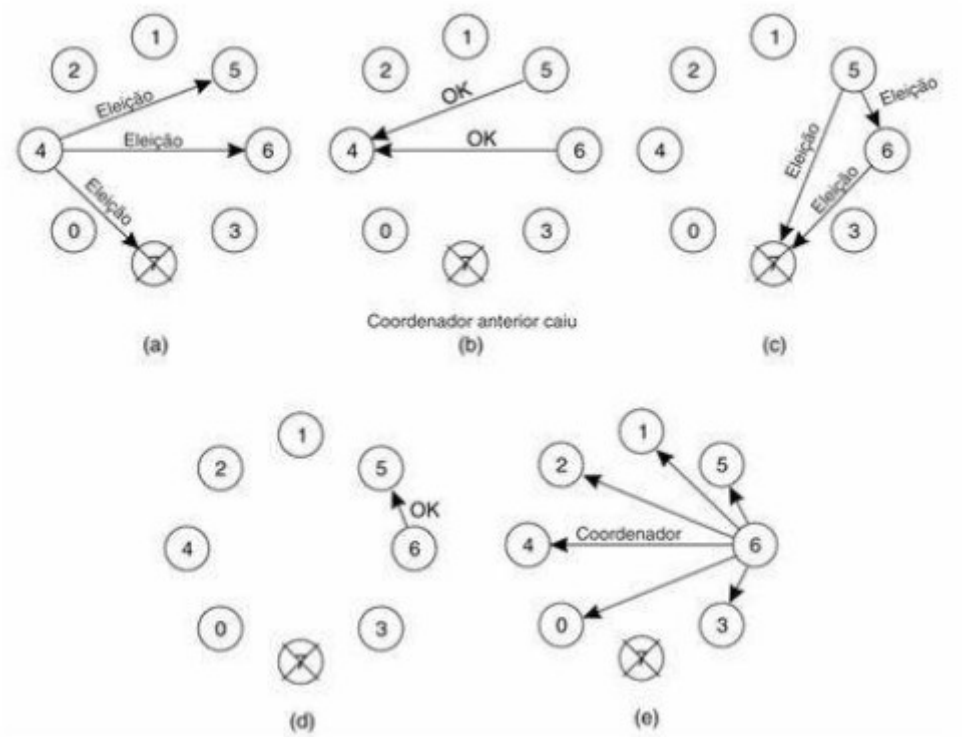
Algoritmo do valentão

Quando um processo P "percebe" que o líder atual não está mais respondendo, P inicia uma eleição:

- P envia uma mensagem "ELEIÇÃO" a todos os processos com ID maior que o seu.
- Se nenhum destinatário responder, P vence a eleição e se torna o líder.
- Se um processo com ID maior responder, P desiste da eleição.

Quando uma mensagem "ELEIÇÃO" chega a um destinatário, este responde com uma mensagem "OK" informando ao processo P que ele irá assumir o controle (Valentão) iniciando sua própria candidatura, caso ainda não o tenha feito.

O processo que vence a eleição comunica a todos os nós que agora ele é o líder.



Algoritmo do anel

Quando um processo P “percebe” que o líder atual não está mais respondendo, P inicia uma eleição:

- P envia uma mensagem “ELEIÇÃO + [P_{ID}]” a seu sucessor. Caso o sucessor não esteja ativo, P submete a mensagem ao sucessor seguinte na cadeia.
- Cada nó que recebe a mensagem adiciona a si mesmo como candidato inserindo seu próprio ID na mensagem: “ELEIÇÃO + [P_{ID}, Q_{ID}, …]”.
- Quando o nó que iniciou a eleição (P) recebe a mensagem de volta após um ciclo, este reconhece seu próprio ID na cadeia e envia novamente uma mensagem, desta vez passando apenas pelos sucessores identificados na cadeia e comunicando o novo líder eleito: o maior ID presente no ciclo.