**Threads**

# Introdução

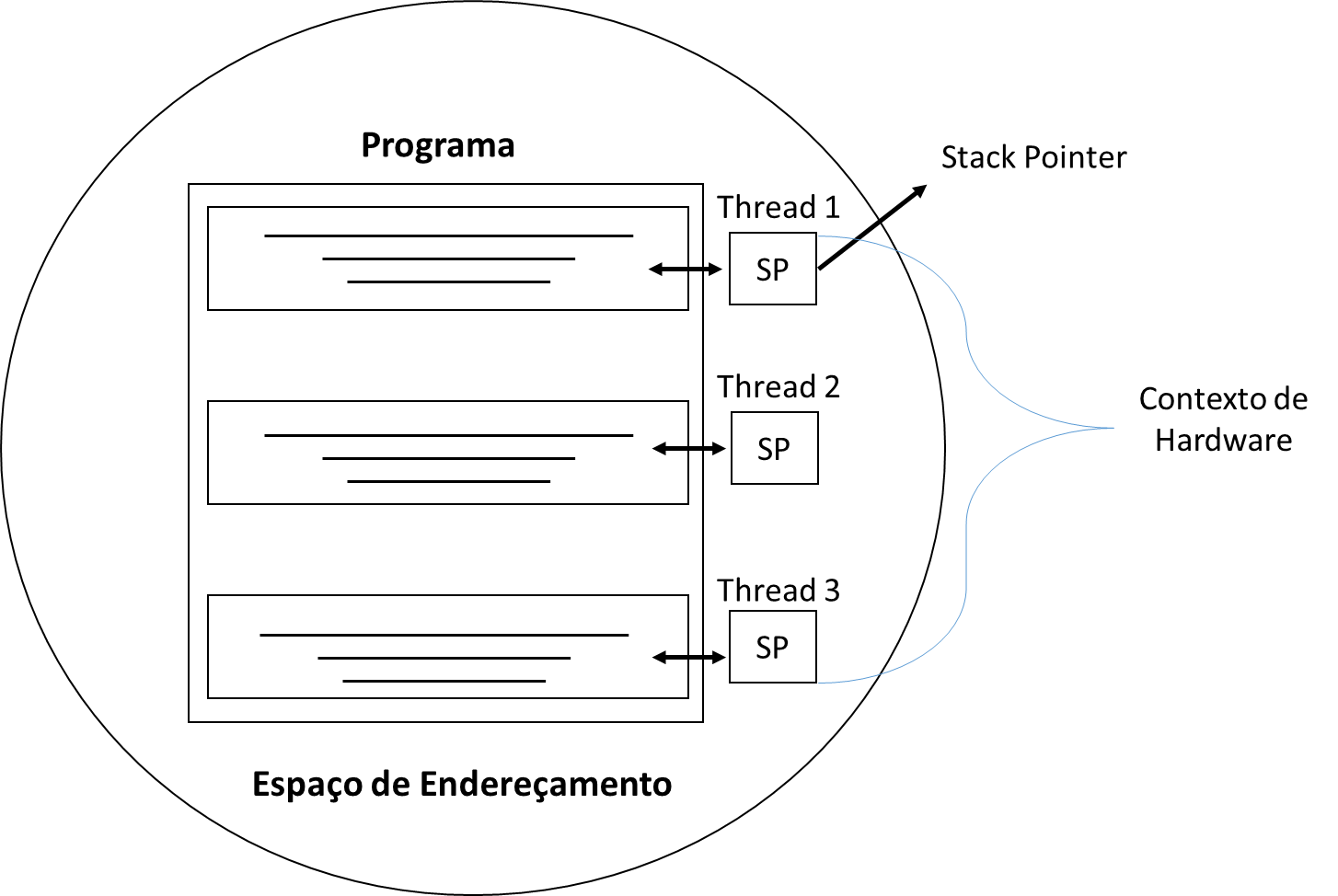
* Até o final da década de 70, sistemas operacionais suportavam apenas processos com uma única thread.
* **1979**: surgimento de processos light-weight onde o espaço de endereçamento era compartilhado com vários programas.
* **1980:** viabilidade com sistema operacional Mach (clara separação entre processo e thread).
* Hoje um processo pode ter partes diferentes de seu código sendo executados em paralelo e como as threads ocupam o mesmo espaço de endereçamento do processo, melhora-se seu desempenho.

# Ambiente MonoThread

* Um processo suporta apenas um programa em seu espaço de endereçamento.
* A concorrência existe apenas entre processo e subprocessos acarretando maior demanda por recursos do sistema principal na criação de novo processo.
* As primeiras versões do Windows são exemplos de sistemas operacionais monothreads.

# Ambiente MultiThread

* Um processo tem uma thread em execução mas pode compartilhar seu espaço de endereçamento com outras threads.
* **Thread:** pode ser definido como uma subrotina de um programa que pode ser executada de forma assíncrona e concorrente dentro de um processo.
* Vantagem desse ambiente é minimizar alocação de recursos e diminuir o overhead (criação, troca e eliminação de processos).
* Threads compartilham o processador da mesma maneira que os processos e são implementados pela TCB (Thread Control Block).



## Exemplo 1

X:= SQRT (1024) + (35.4 \* 023) – (302/7)

Program Expressao

Var x, t1, t2, t3 : Real;

Begin

ParaBegin

T1:= SQRT (1024);

T2:= 35.4 \* 0.23;

T3:= 302/7;

ParaEnd;

X:= T1+T2+T3;

Write (x);

End

## Exemplo 2

Latência entre processos e threads segundo Vahalia (1996)

|  |  |  |
| --- | --- | --- |
| Implementação | Tempo de Criação  (Micro segundos) | Tempo de Sincronização (Micro Segundos) |
| Processo | 1700 | 200 |
| Lightweight | 350 | 390 |
| Thread | 52 | 66 |

# Arquitetura

## Threads em Modo Usuário

Threads em modo usuário (TMU) são implementados pela aplicação e não pelo sistema operacional.

Nesse modo, o S.O não sabe da existência de múltiplos threads, sendo responsabilidade exclusiva da aplicação gerenciá-los.

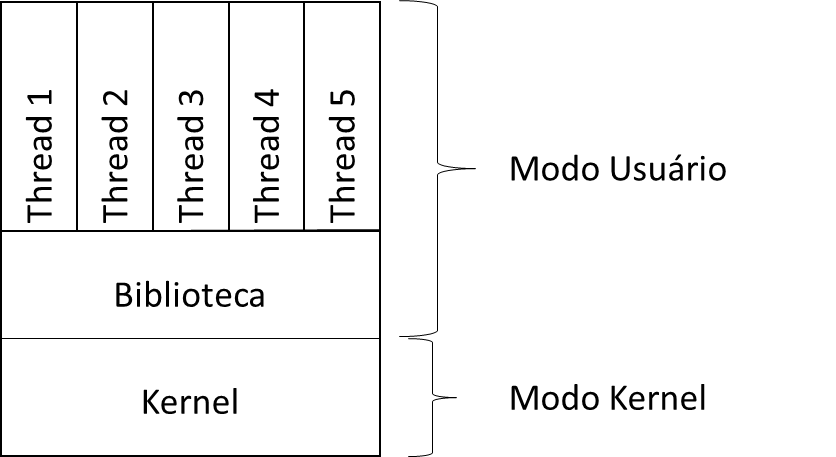
### Vantagens

A vantagem do TMU é a possibilidade de implementar aplicações multithread mesmo em S.O que não suportam threads.

São rápidos por dispensar o acesso ao Kernel do S.O evitando a mudança de modo de acesso (usuário – Kernel – Usuário).

### Desvantagens

A desvantagem é que não é possível que múltiplos threads sejam executados em diferentes CPUs simultaneamente pois o S.O enxerga apenas o processo e não múltiplos threads.



## Threads em Modo Kernel

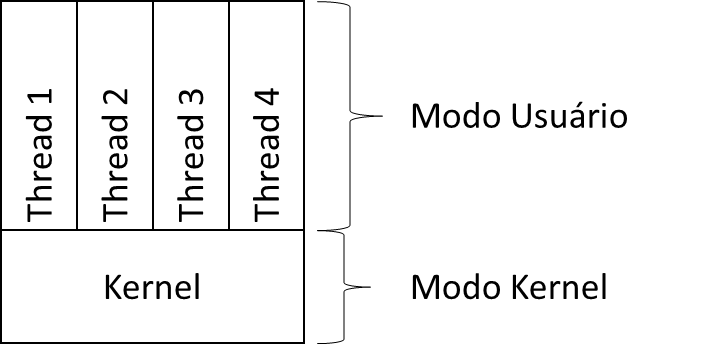
TMK, são implementados diretamente pelo núcleo do S.O que conhece a existência de cada thread e como escaloná-lo.

### Vantagens

No caso de múltiplos processadores, as threads de um mesmo processo podem ser executadas simultaneamente.

### Desvantagens

A desvantagem é seu baixo desempenho pela constante troca de modo de acesso (Usuário – Kernel - Usuário).



|  |  |  |
| --- | --- | --- |
| Implementação | Operação 1  (Micro segundos) | Operação 2  (Micro Segundos) |
| Subprocessos | 11300 | 1840 |
| TMK | 948 | 441 |
| TMU | 34 | 37 |

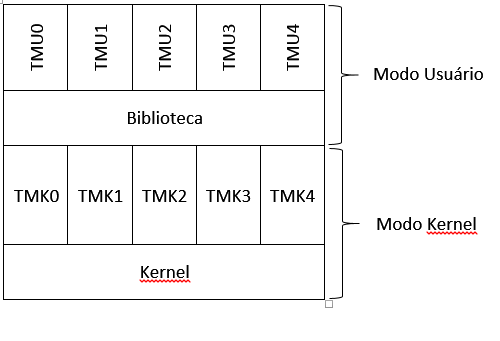
## Threads em Modo Híbrido

A arquitetura de threads em modo híbrido combina as vantagens de threads implementados em Modo Kernel e Modo Usuário.

Um processo pode ter várias TMK e, por sua vez, várias TMU.

O modo Híbrido, apesar de maior flexibilidade, apresenta problemas herdados de ambas implementações.

Quando um TMK realiza uma chamada à rotina do sistema, todos TMU são colocados em modo (estado) de espera.



## Scheduler Activations

Os problemas apresentados no pacote de thread em modo híbrido existem devido à falta de comunicação entre os TMK e TMU. O modelo ideal deveria mesclar as facilidades do TMK com o desempenho do TMU.

O Scheduler Activitions combina o melhor das duas arquiteturas, mas em vez de dividir os threads em modo usuário e Kernel, o núcleo do sistema troca informações com a biblioteca de threads utilizando essa arquitetura.

A maneira de se obter o melhor desempenho é evitar as mudanças de modos de acesso desnecessários. Isso é possível porque a biblioteca em TMK e TMU se comunicam e trabalham de forma cooperativa.

