Sistemas Operativos

LEI - 2019/2020

:: Threads ::

Escola Superior de Tecnologia de Setúbal - IPS

Conteúdos

- Noção de threads
- APIs de threads (Pthreads, Java threads, Win threads)
- Estratégias implícitas de threading
- Multithreading

Thread

Unidade básica de utilização do CPU ("linha" de execução)

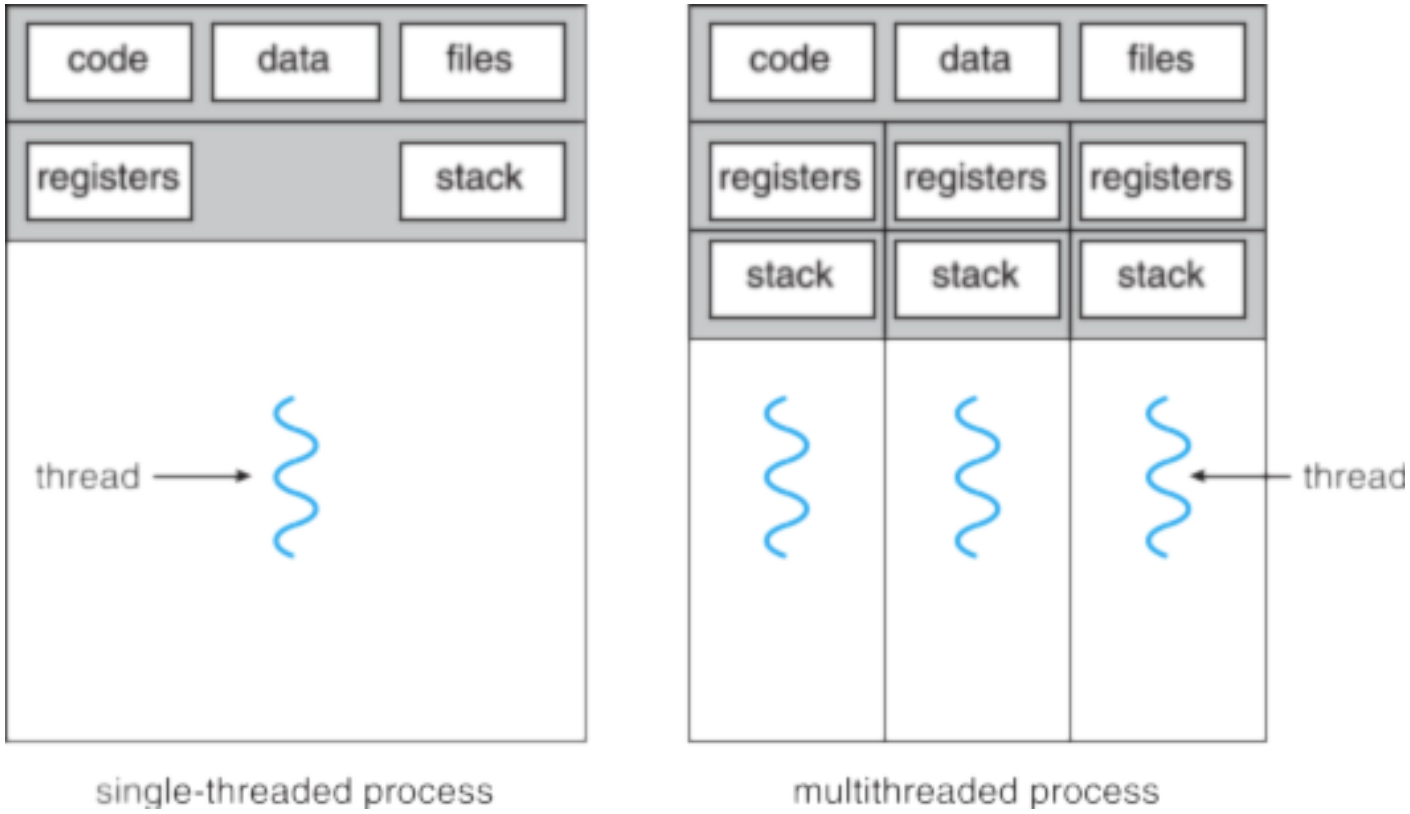
Contém:

- threadID
- Program counter / registos
- Stack

Partilha com outras threads do mesmo processo:

- Código
- Variáveis globais
- memória heap do processo

Processos single-thread vs multithread



Motivação

Processos podem necessitar paralelismo interno, com acesso:

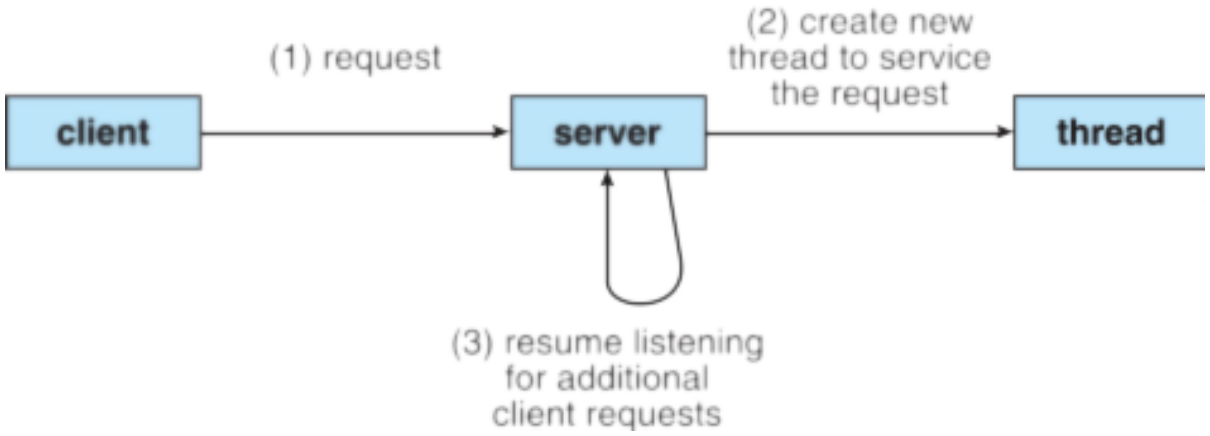
- aos mesmos dados
- aos mesmos ficheiros
- aos mesmos recursos do processo

Ex: Servidor web multithreaded

Servidor single-threaded só aceitaria um cliente de cada vez..

Com threads:

- Espera por ligações de clientes
- Aceita uma ligação e cria uma thread para tratar do pedido
- Volta a esperar por novos clientes
- A thread criada trata, paralelamente, dos pedidos do cliente

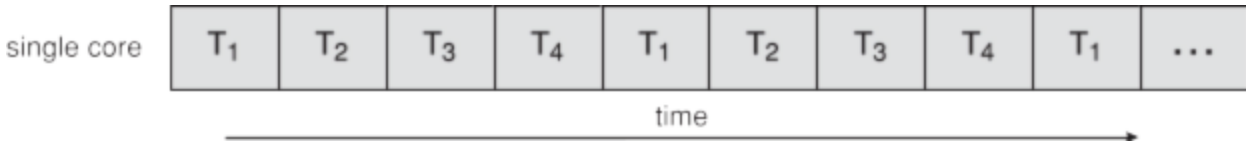


Benefícios

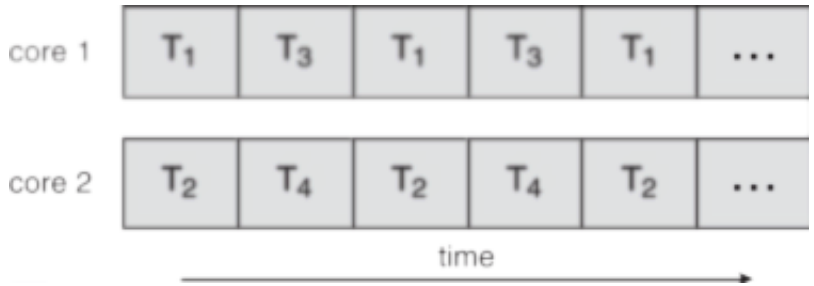
- **Capacidade de resposta:** partes de uma aplicação podem bloquear, e outras manterem-se activas (ex: UI)
- **Partilha de recursos:** paralelismo e partilha de informação entre threads acedendo aos recursos dentro do processo
- **Economia:** mais económico que criar processos novos e context-switching é menor (~5X)
- **Escalabilidade:** funciona bem em arquitecturas multicore

Programação single vs multicore

Execução concorrencial num sistema single-core



Execução concorrencial num sistema multicore



Lei de Amdahl

Identificar ganhos de performance adicionando N cores.

S - Componente série (%)

N - Número de cores

$$speedup \leq \frac{1}{S + \frac{(1-S)}{N}}$$

Ex: 40% da aplicação é em série => ganhos máximos de ~2.5X

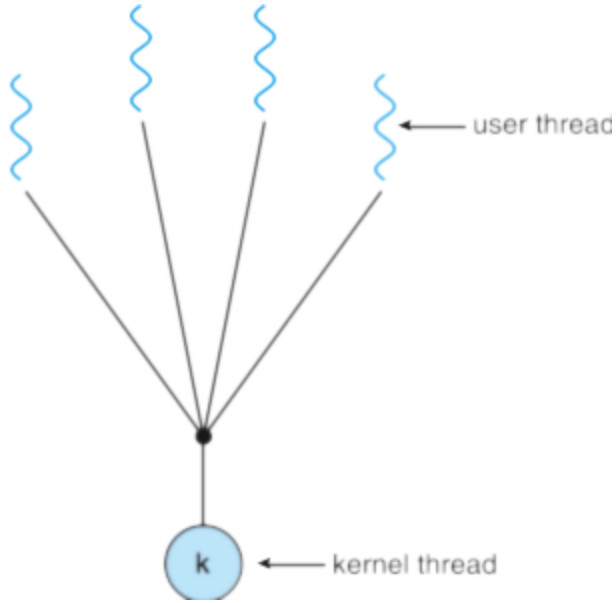
User vs kernel threads

User threads: gestão feita pelo software em user mode

Kernel threads: gestão feita pelo kernel

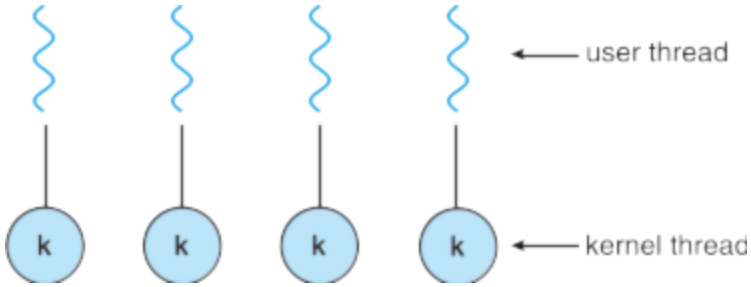
💻 Relação entre kernel threads e user threads?

Modelo many-to-one



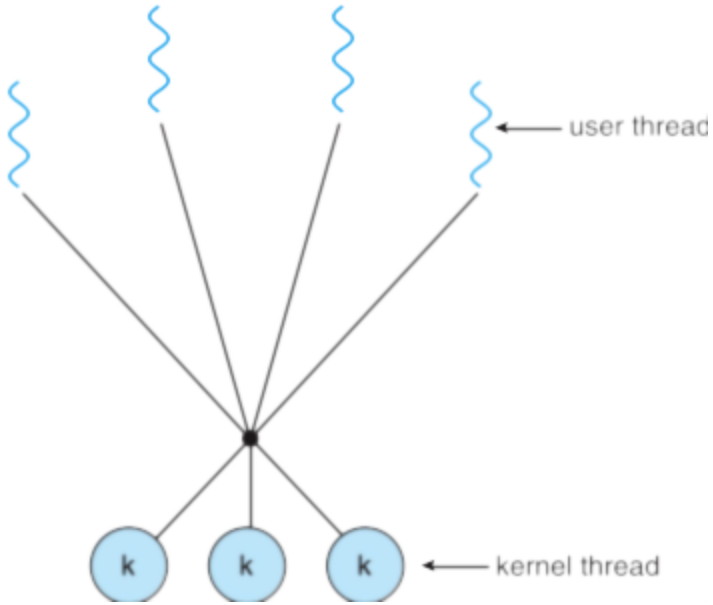
💻 Ex: "green threads"..

Modelo one-to-one



💻 Ex: Linux, Windows..

Modelo many-to-many



💻 Ex: Goroutines..

Biblioteca pthreads

```c
#include <stdio.h>
#include <pthread.h>

void *thread(void *params) {
    printf("%s\n", (char *) params);
    return 0;
}

int main()
{
    // Create threads
    pthread_t thread1, thread2;
    // Start threads
    pthread_create(&thread1, NULL, *thread, "Thread 1");
    pthread_create(&thread2, NULL, *thread, "Thread 2");
    // Wait for threads to finish
    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);
    return 0;
}
```

Java threads

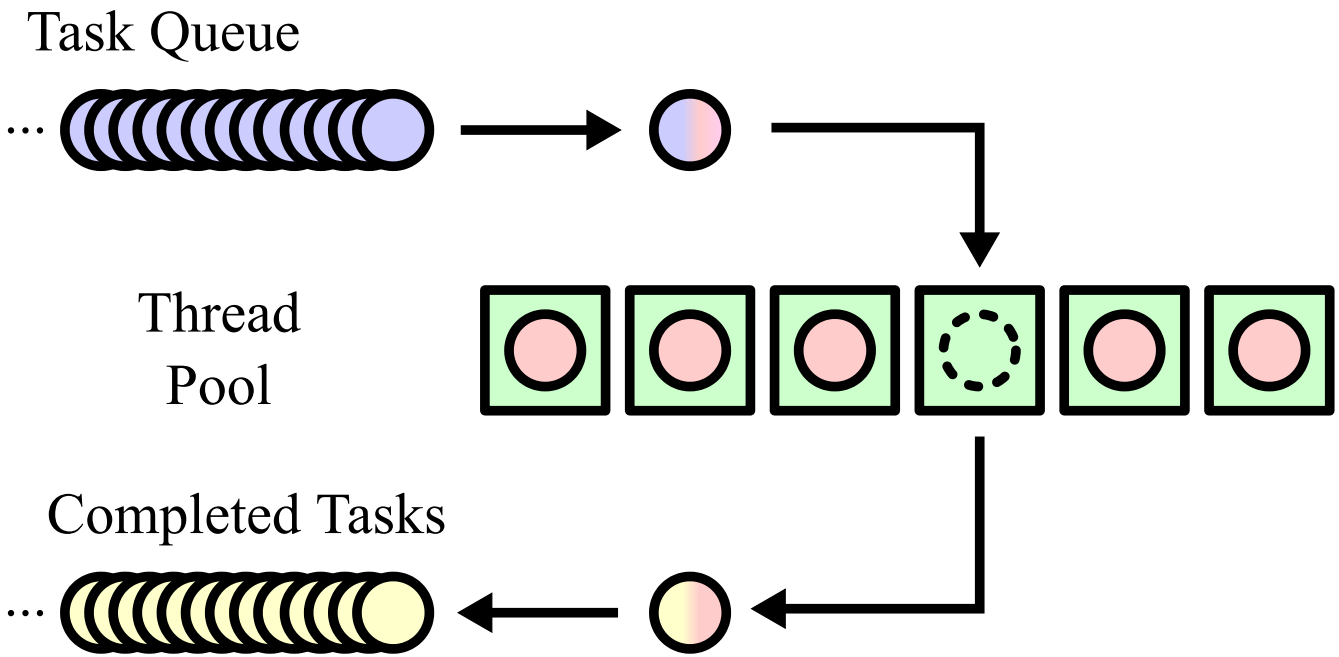
```
public class SimpleThread extends Thread {

    @Override
    public void run() {
        for (int i=0; i<5; i++) {
            System.out.println(this.getName() + ": " + i);
        }
    }

    public static void main(String[] args) {
        SimpleThread thread0 = new SimpleThread();
        SimpleThread thread1 = new SimpleThread();
        thread0.start();
        thread1.start();
    }
}
```

Threads implícitas

- Aplicações com centenas ou milhares de threads podem ser difíceis de gerir.
- Transferir a criação e gestão de threads para compiladores ou bibliotecas...

Thread pools

Tarefas vão utilizando threads criadas inicialmente..



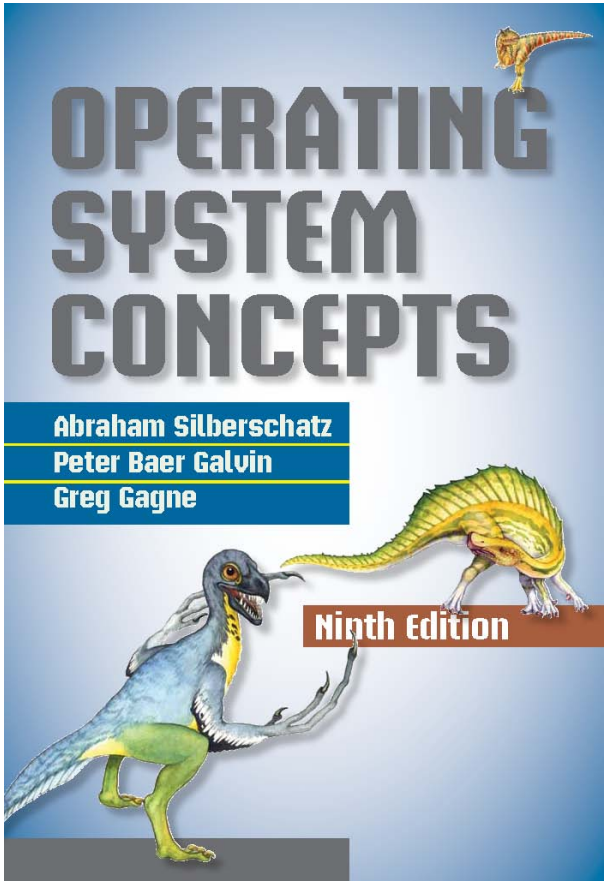
OpenMP

```c
#include <omp.h>
#include <stdio.h>

int main()
{
    // Sequential code

    #pragma omp parallel
    {
        printf("Hello from parallel code!\n");
    }

    // Sequential code
    return 0;
}
```

Quiz...

Sumário

- Threads são linhas de execução dentro de um processo
- Permite partilhar recursos do processo e paralelismo
- Existem diferentes modelos de threads
- Existem várias bibliotecas de utilização de threads



Ler capítulo 4...