Threads

Visão Geral

Modelos Multithreading

Detalhes de Threading

Pthreads

Threads no Solaris 2

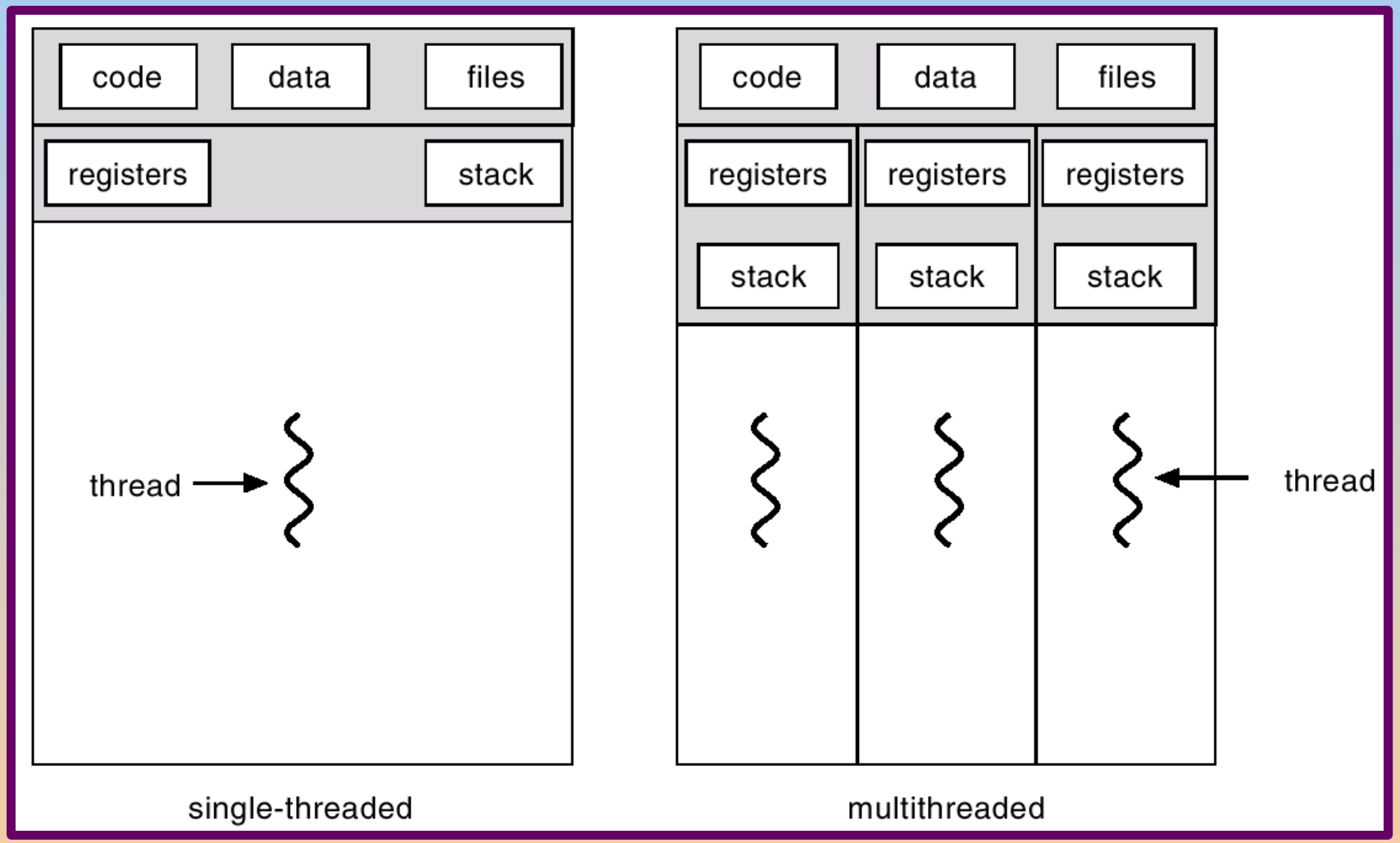
Threads no Windows 2000

Threads no Linux

Threads em Java

Gisele S. Craveiro

Processos simples e com Multithreading



Vantagens de Multithreading

- Delegação de tarefas entre threads
- Compartilhamento de recursos
- Economia de tempo para lançar novas tarefas

Threads do Usuário (User Threads)

- Gerenciamento de threads é feito por bibliotecas sendo executadas no nível usuário

- Exemplos

  - Pthreads POSIX
  - *C-threads* Mach
  - Threads Solaris

Threads de Núcleo(Kernel Threads)

- Suportados pelo kernel

- Exemplos
  - Windows XP/2000
  - Solaris
  - MacOS X
  - Linux

Modelos de Multithreading

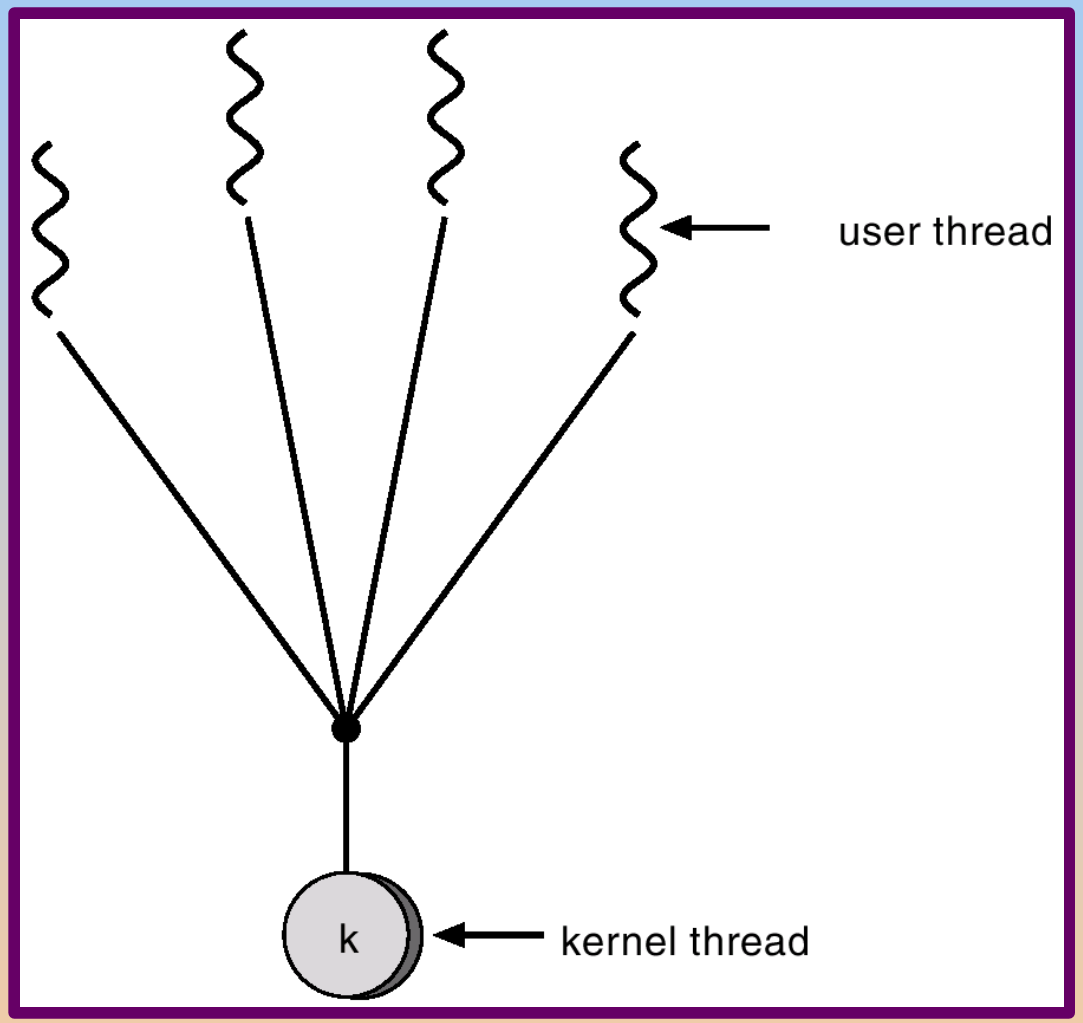
- Muitos-para-um(Many-to-One)
- Um-para-um(One-to-One)
- Muitos-para-Muitos(Many-to-Many)

Muitos-para-um

- Muitos threads do nível usuário mapeados em um único thread de núcleo .

- Usado em sistema que não suportam vários threads de núcleo .

Modelo Muitos-para-um



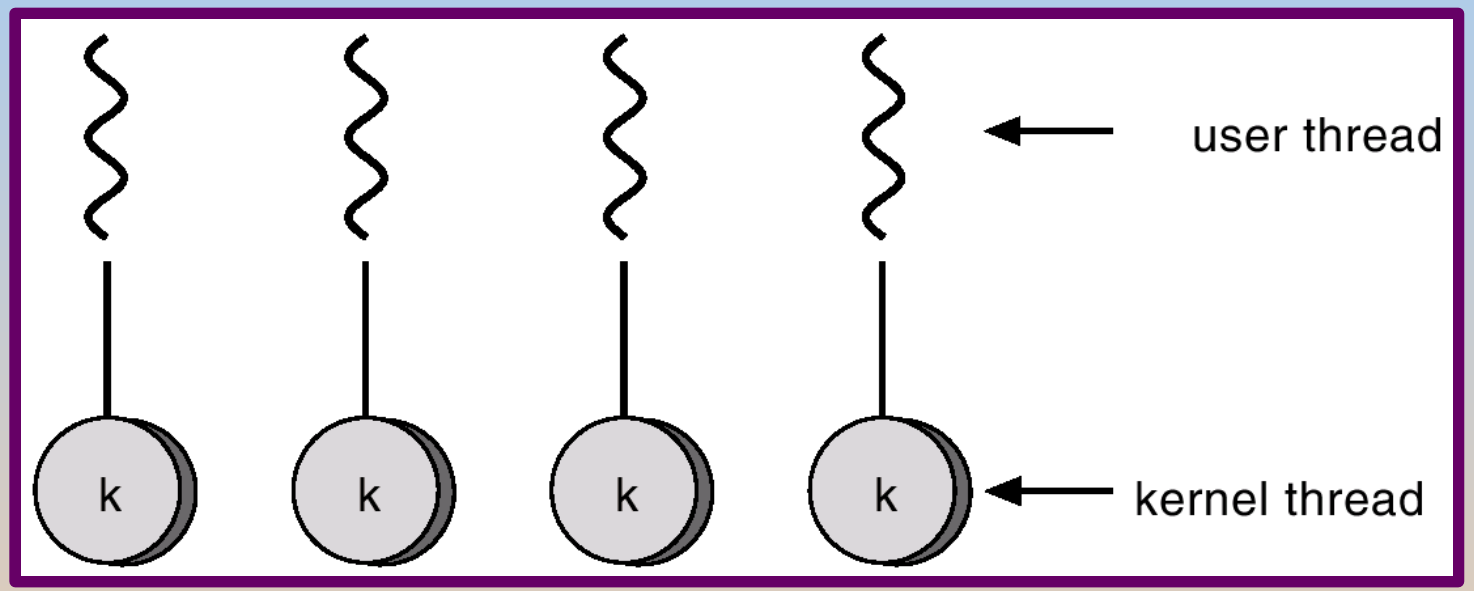
Um-para-um

- Cada thread do nível usuário é mapeado para um thread de núcleo

- Exemplos

  - Windows XP/2000
  - Linux

Modelo um-para-um



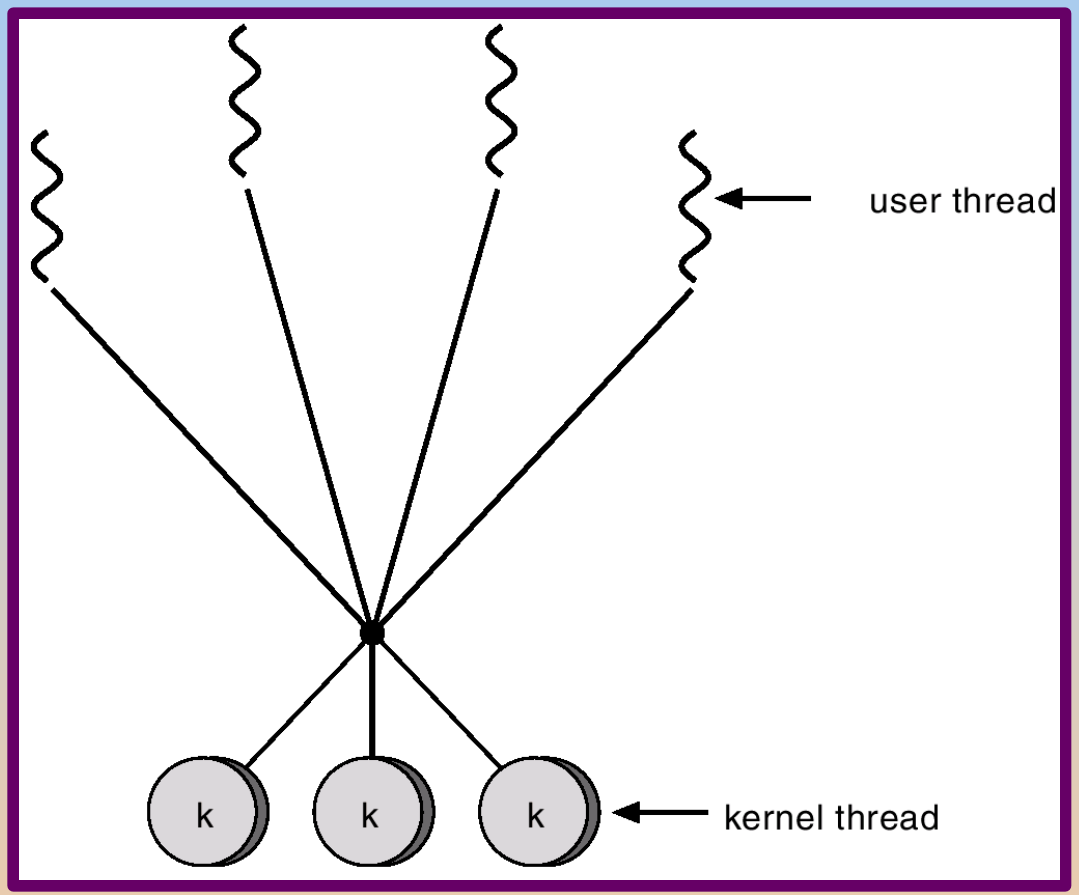
Modelo muitos-para-muitos

- Permite que vários threads de nível usuário ser mapeados em vários threads de núcleo
- Permite o sistema operacional criar um número suficiente de threads de núcleo
- Exemplos:

  Solaris 2

  Windows NT/2000 com o pacote *ThreadFiber*

Modelo muitos-para-muitos



Detalhes envolvidos no mecanismo de Threads

- Semântica das systems calls fork() e exec()
- Cancelamento de Thread
- Manipulação de sinais
- Áreas de comunicação entre Threads (Thread pools)
- Dados específicos de um Thread

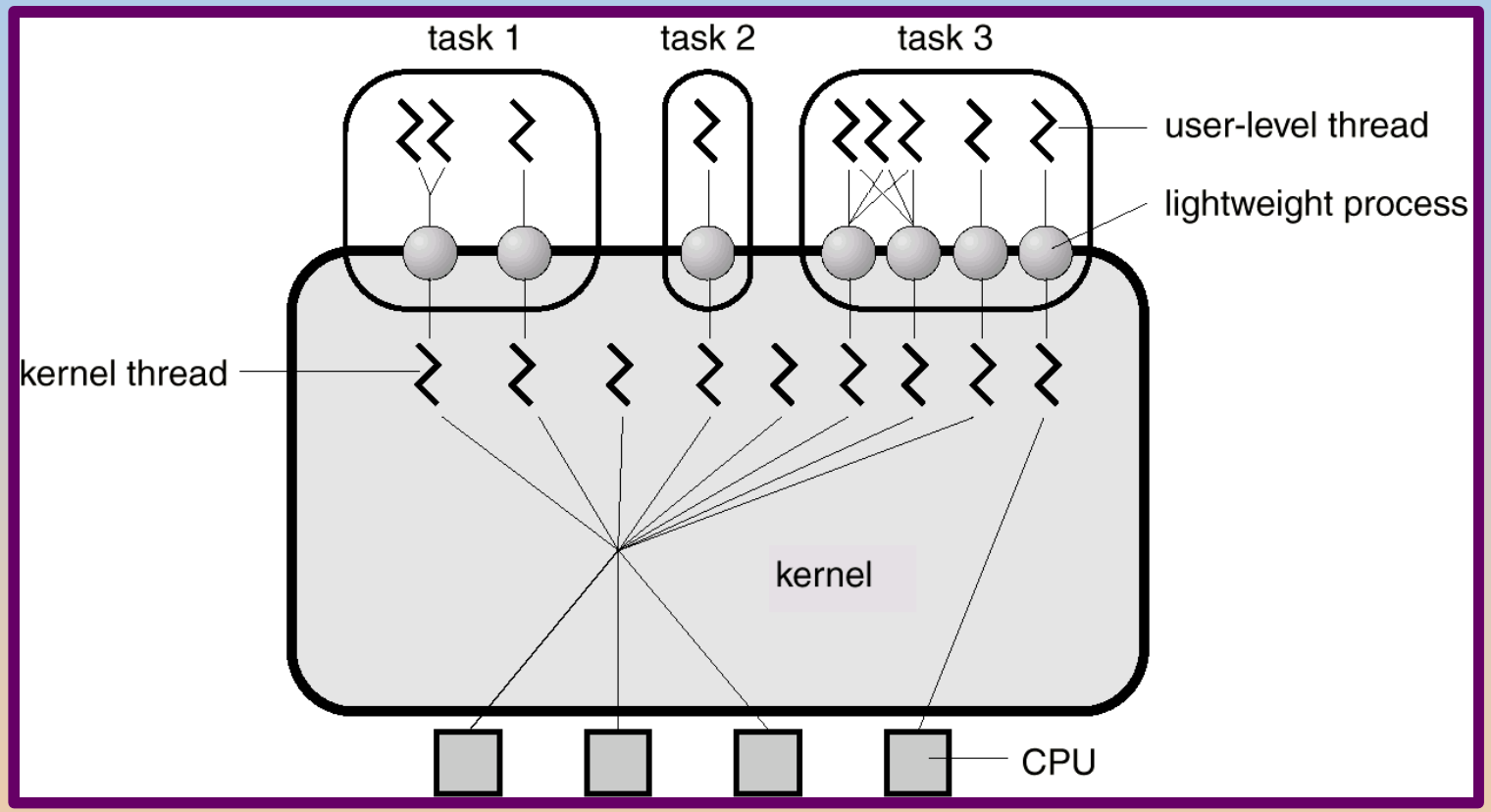
Pthreads

- Um padrão POSIX (IEEE 1003.1c) para criação e sincronização de threads .
- O padrão especifica o comportamento da biblioteca de threads. A implementação fica por conta da biblioteca.
- Padrão muito comum em sistemas operacionais UNIX.

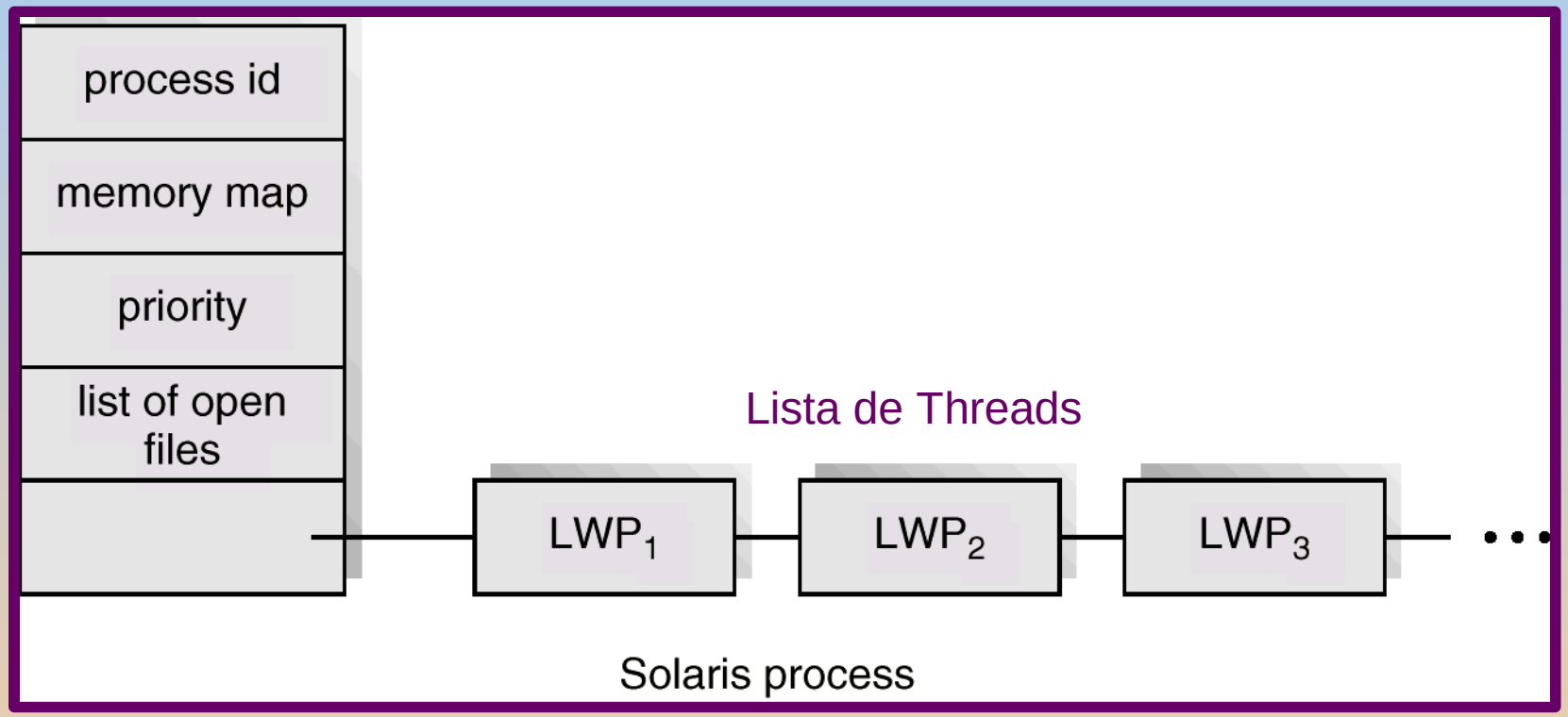
Exemplo em Pthread

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

int sum; /* this data is shared by the thread(s) */
void *runner(void *param); /* the thread */

main(int argc, char *argv[])
{
pthread_t tid; /*the thread identifier */
pthread_attr_t attr; /* set of attributes for the thread */

/* get the default attributes */
pthread_attr_init(&attr);

/* create the thread */
pthread_create(&tid,&attr,runner,argv[1]);

/* now wait for the thread to exit */
pthread_join(tid,NULL);

printf("sum = %d\n",sum);
}

/**
 * The thread will begin control in this function
 */
void *runner(void *param)
{
int upper = atoi(param);
int i;
sum = 0;

   if (upper > 0)
     for (i = 1; i <= upper; i++)
                  sum += i;
pthread_exit(0);
}
```

Threads no Solaris 2



Processos no Solaris



Threads no Windows XP

- Implementa o mapeamento um-para-um
- Cada thread contém:
  - um thread id
  - conjunto de registradores
  - pilhas de execução separadas para threads de usuário e de núcleo
  - área de armazenamento de dados privada

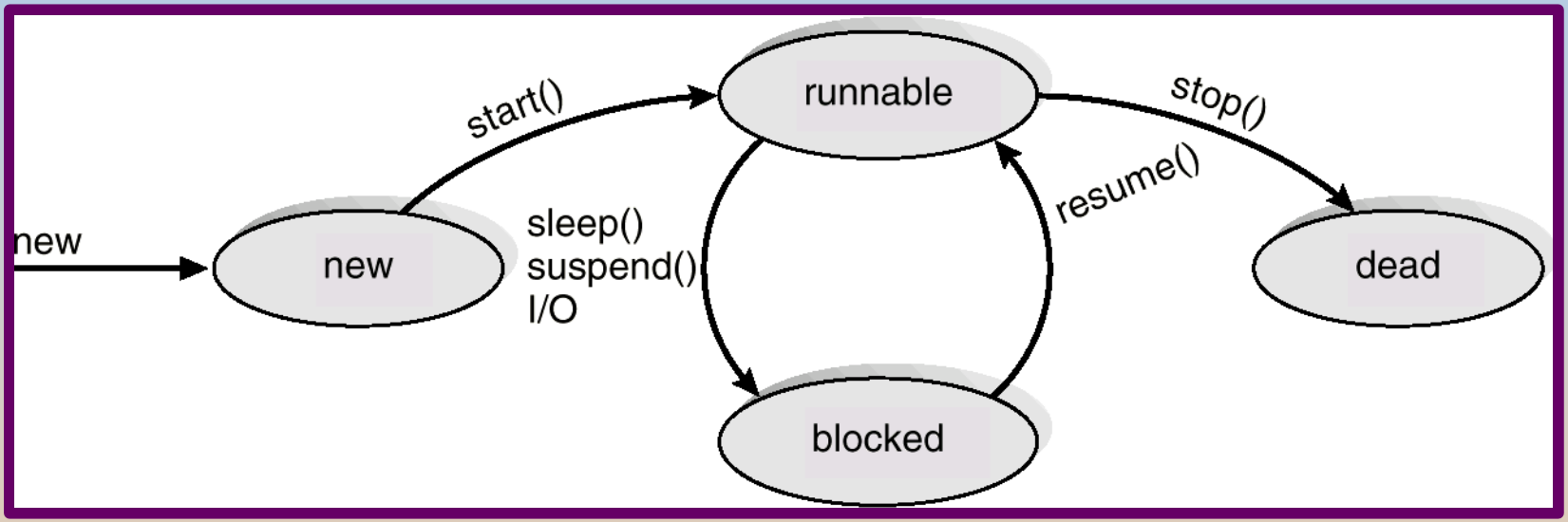
Threads Linux

- Linux denomina threads de tarefas(tasks)
- Criação de threads é feita através da system call clone()
- Clone() permite que uma tarefa-filha compartilhe o espaço de endereçamento do tarefa-pai(processo) .

Threads Java

- Threads Java podem ser criados por:
  - Extensão da classe Thread ou
  - Implementando a interface Runnable
- Threads Java são geranciados pela Java Virtual Machine(JVM).

Estados de Threads Java



Atividade extra-classe

- Analisar o código do produtor-consumidor multi-threaded apresentando no livro texto
- Implementá-lo e testá-lo