BCC 362 – Sistemas Distribuídos

Joubert de Castro Lima – joubertlima@gmail.com
Professor Adjunto – DECOM

UFOP

Como o Sistema Operacional lida com o paralelismo???

Ele cria o conceito de

processo

PROCESSO: possui o código do programador + contador do programa + o conteúdo dos registradores + pilha (usada para dados temporários) + seção de dados para as variáveis globais

PROCESSO NÃO É O SEU CÓDIGO

O SEU CÓDIGO É UMA ENTIDADE PASSIVA (UM ARQUIVO APENAS!!!)

UM PROCESSO É ATIVO, ESTÁ EM EXECUÇÃO.

Como o Sistema Operacional lida com o paralelismo???

DEFINIDO O QUE É PROCESSO, então por que não permitir a criação de vários processos?? É isto que nossos SOs fazem também e muito bem!!!

A chamada de sistema para criar novos processos é Fundamental

Múltiplos processos => SOLUÇÕES TRAZEM PROBLEMAS.....

CONDIÇÕES DE DISPUTA

Condições de disputa

Em muitos casos processos paralelos precisam sincronizar e se comunicar

Exemplos: normalmente, recurso compartilhado precisa haver S e C

THREADS: recurso disponível para o programador para criação de processos,
Chamados **light process**

Imagine que você deve somar duas matrizes. Como fazer um programa paralelo

Outros exemplos de
dependabilidade....

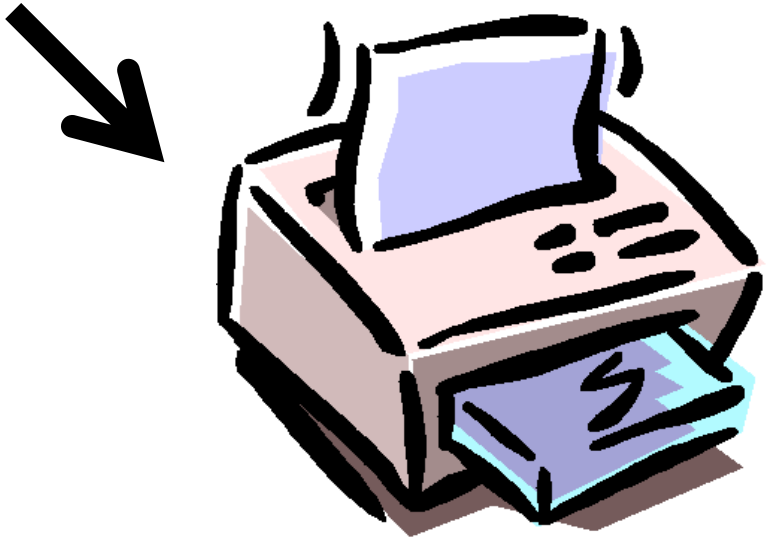
PRODUTOR CONSUMIDOR

Um estilo de solução para problemas recorrentes !!

NA COMPUTAÇÃO

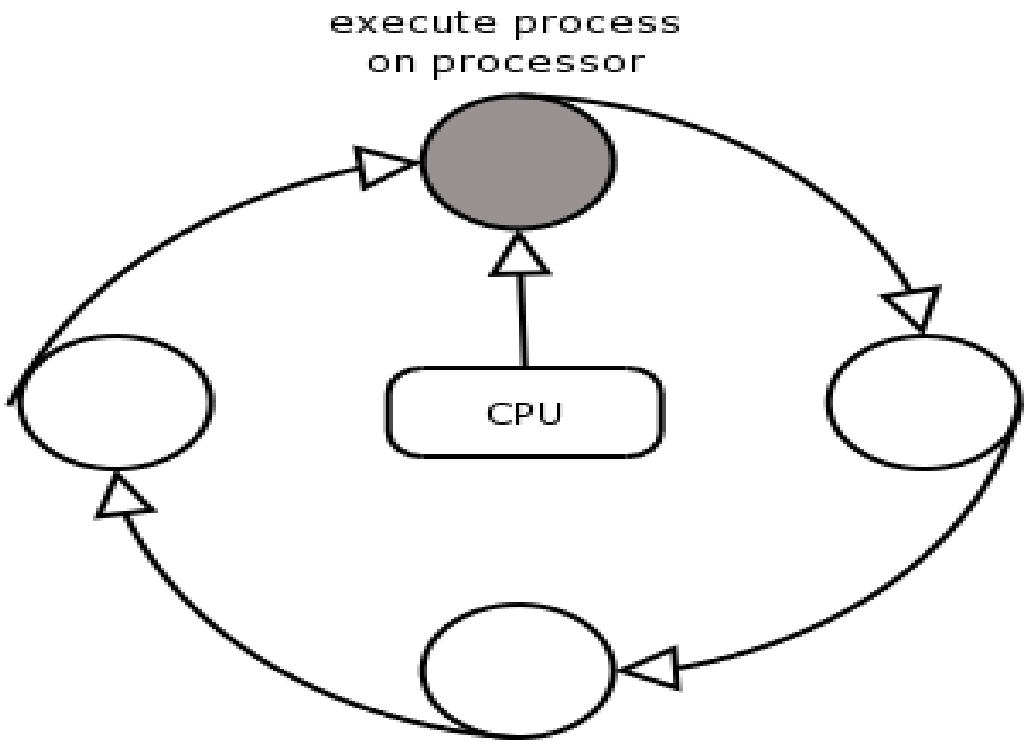
PRODUTOR : software responsável por ler um arquivo do disco (HD) e colocar num recurso (vetor, por exemplo).

CONSUMIDOR : software responsável por retirar do recurso (vetor, por Exemplo) e fazer algo (imprimir, por exemplo).



Princípio básico: Sincronização entre processos

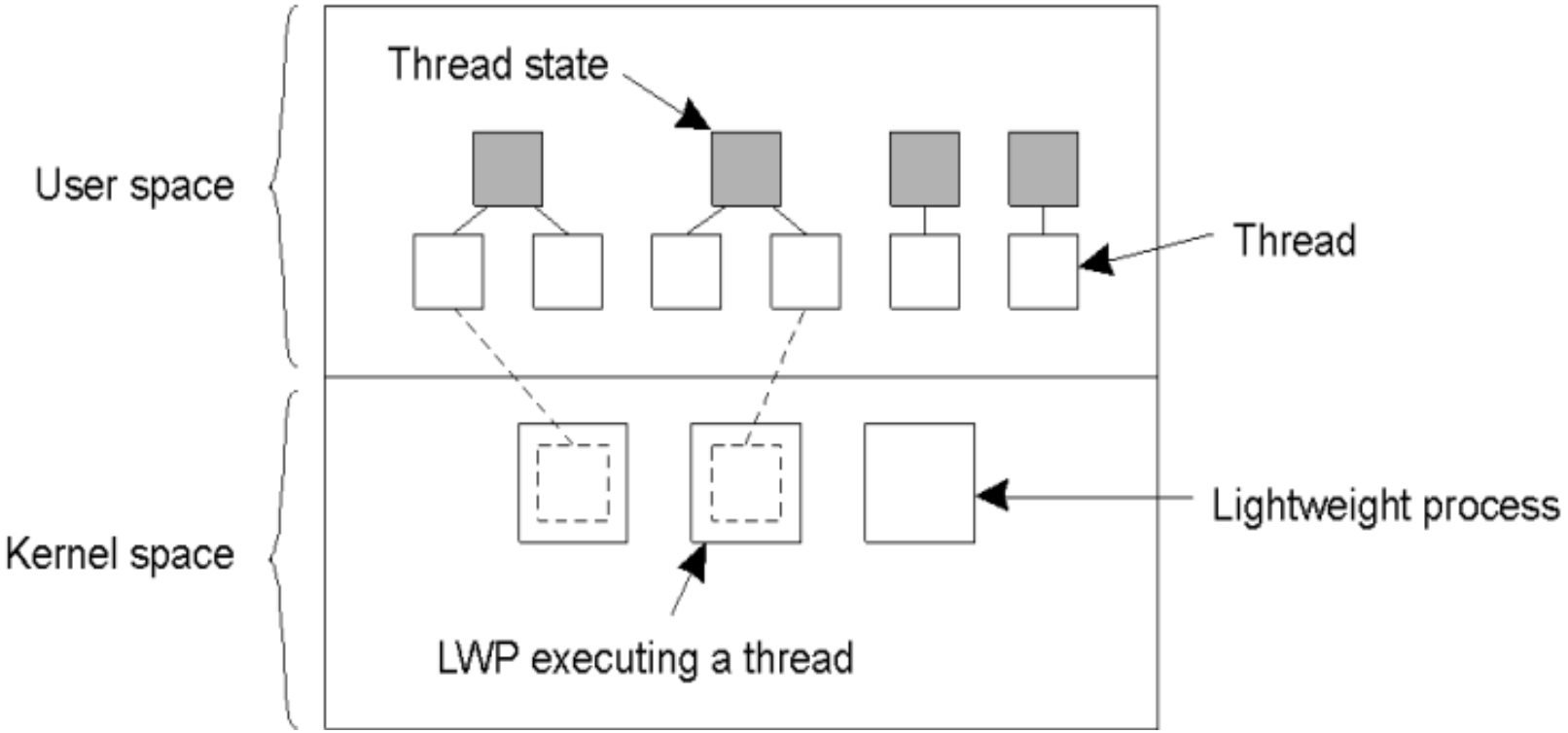
Escalonamento



Por que é interessante criar múltiplos processos, mesmo com um único processador?

CONCEITO ELEGANTE DA MULTIPROGRAMAÇÃO.

Thread Implementation



Threads

- Uma porção de código (uma task, por exemplo!!!) que pode ser executada independentemente de outras Threads

- Um processo possui pelo menos uma Thread

Threads

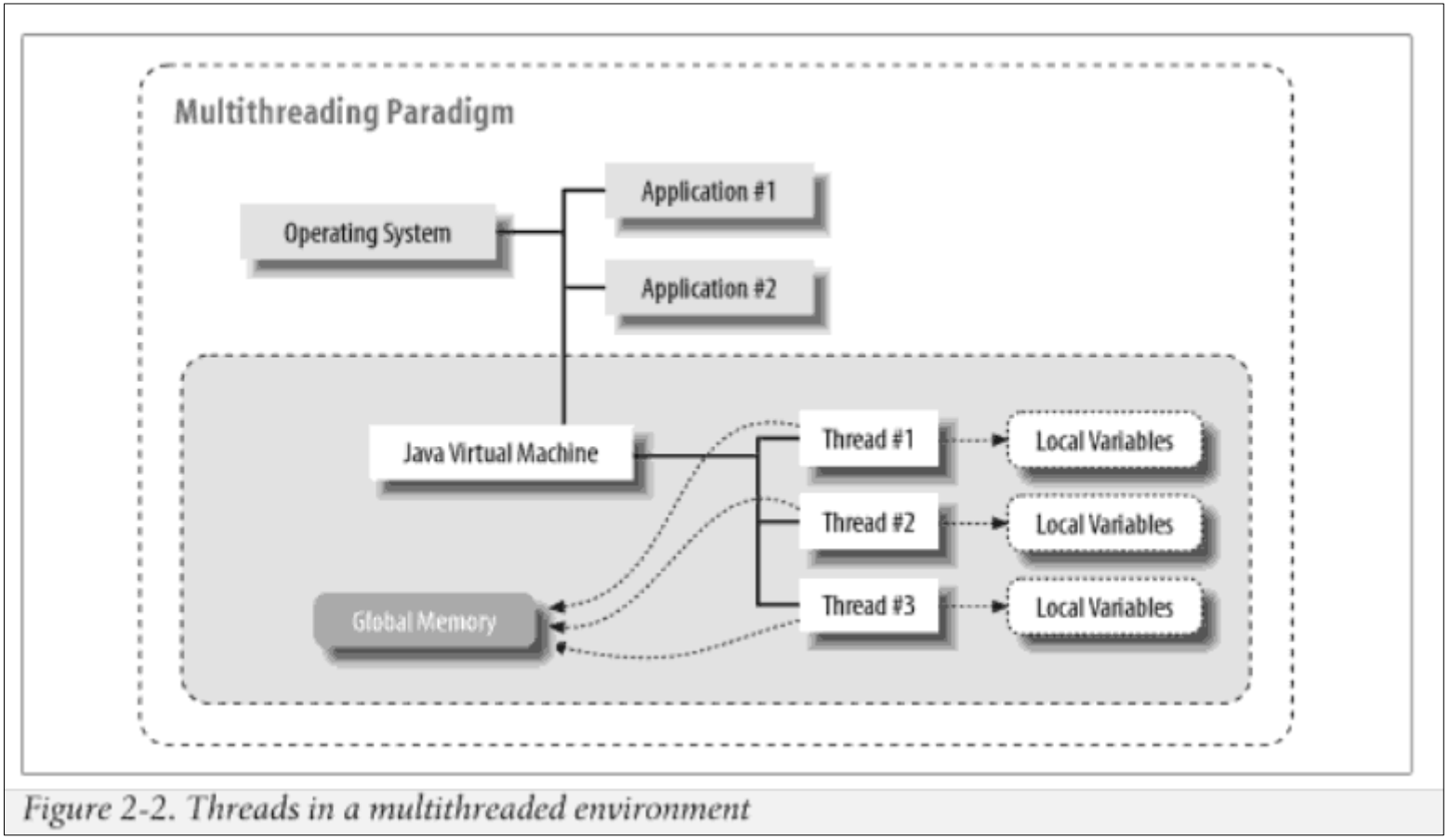


Figure 2-2. Threads in a multithreaded environment

Retirada do livro Java Threads
Scott Oaks

Iniciando Threads em Java

- Usando a interface Runnable

```
public class HelloRunnable implements Runnable {

    public void run() {
        System.out.println("Hello from a thread!");
    }

    public static void main(String args[]) {
        (new Thread(new HelloRunnable())).start();
    }

}
```

Iniciando Threads em Java

- Usando a classe Thread

```
public class HelloThread extends Thread {

    public void run() {
        System.out.println("Hello from a thread!");
    }

    public static void main(String args[]) {
        (new HelloThread()).start();
    }

}
```

Thread ou Runnable

- A classe Thread é uma implementação de Runnable, portanto Thread é uma sub-classe

- Em Java não temos herança múltipla, portanto a classe Runnable é necessária

Class minhaClasse extends minhaSuperClasse implements Runnable

minhaClasse é uma thread
minhaClasse é uma minhaSuperClasse

Parando a execução

Thread.sleep causes the current thread to suspend execution for a specified period.

This is an efficient means of making processor time available to the other threads of an application or other applications that might be running on a computer system.

```
public class SleepMessages {
    public static void main(String args[]) throws InterruptedException {
        String importantInfo[] = {
            "Mares eat oats",
            "Does eat oats",
            "Little lambs eat ivy",
            "A kid will eat ivy too"
        };

        for (int i = 0; i < importantInfo.length; i++) {
            //Pause for 4 seconds
            Thread.sleep(4000);
            //Print a message
            System.out.println(importantInfo[i]);
        }
    }
}
```

Se uma outra thread interromper esta thread a exceção é lançada

Se SleepMessages extends Thread, basta colocar sleep(4000);

Joins

The join method allows one thread to wait for the completion of another.

If t is a Thread object whose thread is currently executing, t.join(); causes the

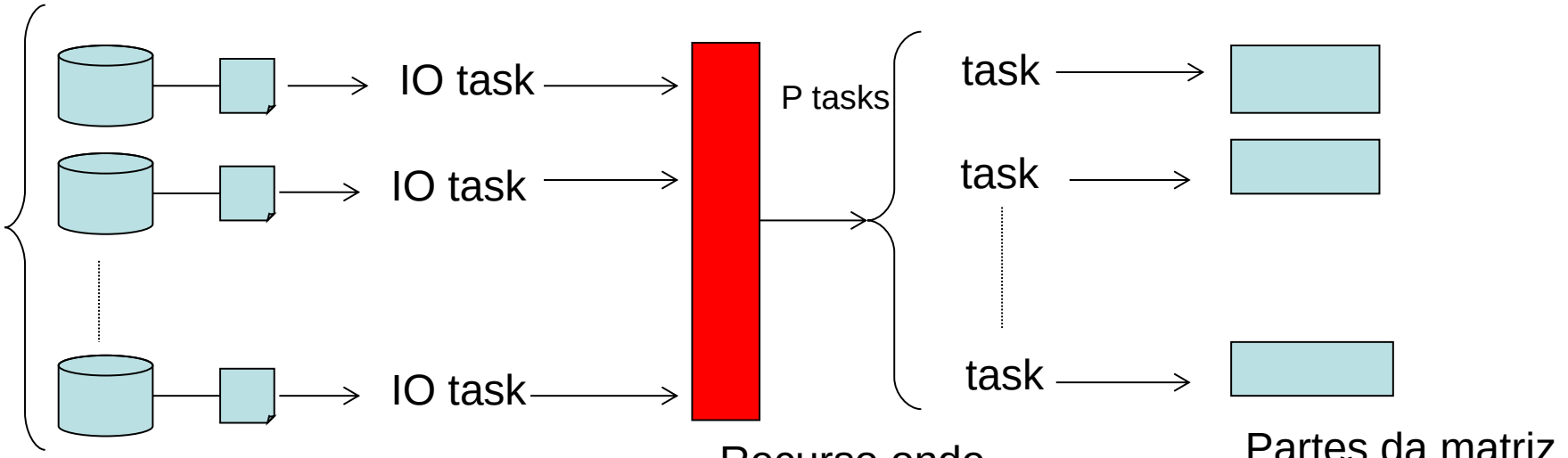
current thread to pause execution until t's thread terminates.

```
public class FiboII extends Thread{
    private int x;
    public int answer;

    public FiboII(int x) {
        this.x = x;
    }

    public void run() {
        if( x < 2 )
            answer = 1;
        else {
            try {
                FiboII f1 = new FiboII(x-1);
                FiboII f2 = new FiboII(x-2);
                f1.start();
                f2.start();
                f1.join();
                f2.join();
                answer = f1.answer + f2.answer;
            }
            catch(InterruptedException ex) { }
        }
    }

}
```

Sincronização, Wait e notify

EXEMPLO

PRODUTOR - CONSUMIDOR



D discos
D arquivos

Cada arquivo com n/D linhas

Recurso onde as linhas são armazenadas

Partes da matriz transposta

USAREMOS CÓDIGO para explicar estes conceitos!!!