BC1518 - Sistemas Operacionais

Threads

Aula 04

2° Quadrimestre de 2010

Prof. Marcelo Z. do Nascimento

Email: marcelo.nascimento@ufabc.edu.br

Universidade Federal do ABC

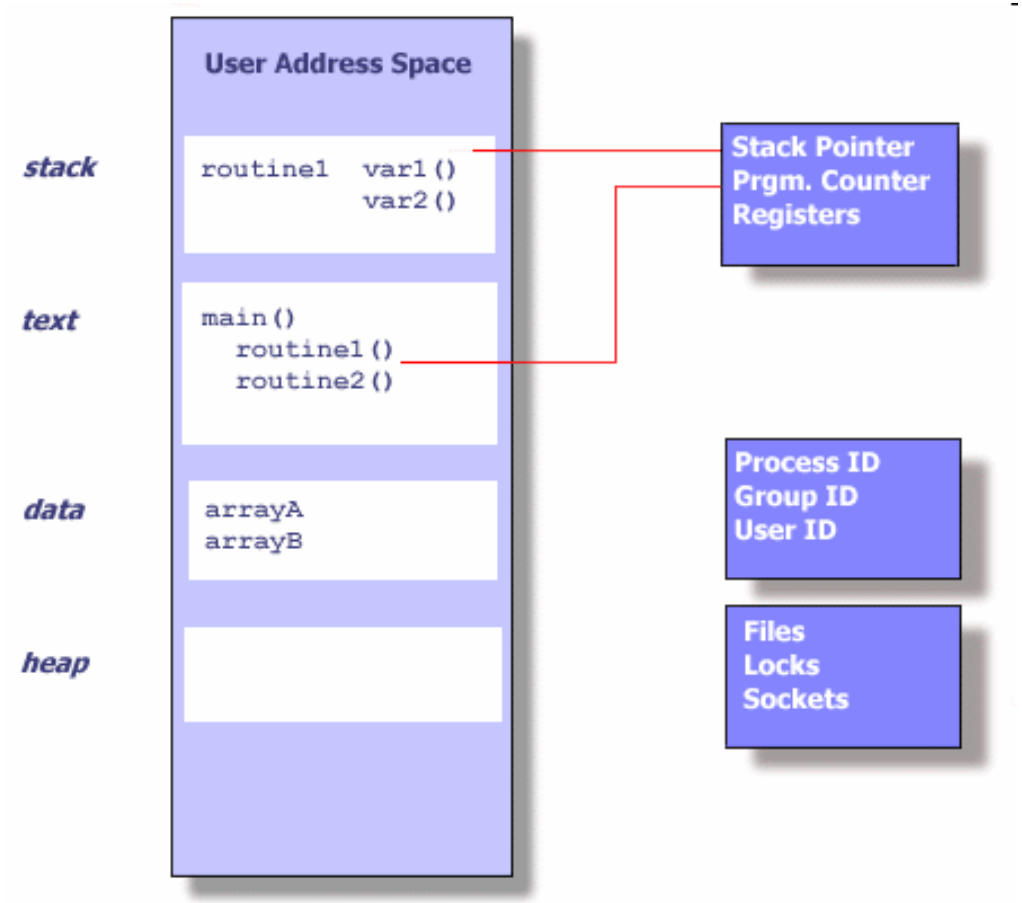
Roteiro

- Threads: Visão Geral
- Benefícios
- Tipos
- Modelos de multithread
- POSIX Thread
- Threads em Java
- Exemplo em SO
- Leituras Sugeridas

Universidade Federal do ABC

Processo

- Processo requer uma quantidade de sobrecarga para sua execução:



Universidade Federal do ABC

Threads

Processo leve (Lightweight Process)

- Ano de 79 => introduziram o conceito (*thread*), onde o espaço de endereçamento era compartilhado;

- Definição: é uma unidade básica de utilização da CPU; compreende um ID, um contador de programa, um conjunto de registradores e uma pilha;

- Um processo pode consistir de várias *threads*, cada uma delas executadas separadamente.

Universidade Federal do ABC

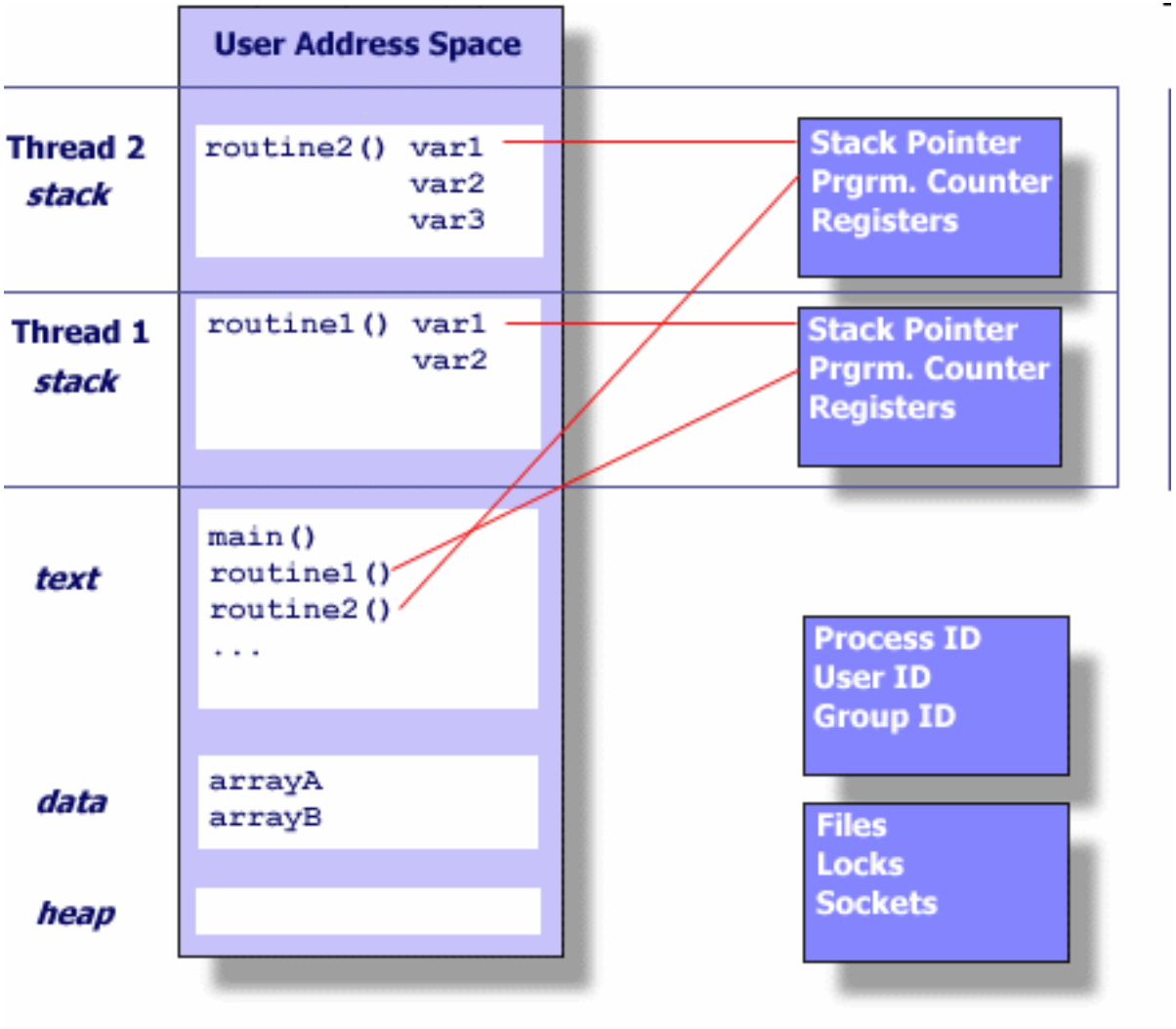
Threads

- A **thread** mantém suas proprias:
  - Pilha;
  - Registradores;
  - Propriedades de escalonamento (politica ou prioridade);
  - Conjunto de sinais de bloqueio;
  - Dados específicos da thread.

- Threads pertencentes ao mesmo processo compartilham:
  - Seção de Código;
  - Seção de dados;
  - Outros recursos do SO, como arquivos abertos.

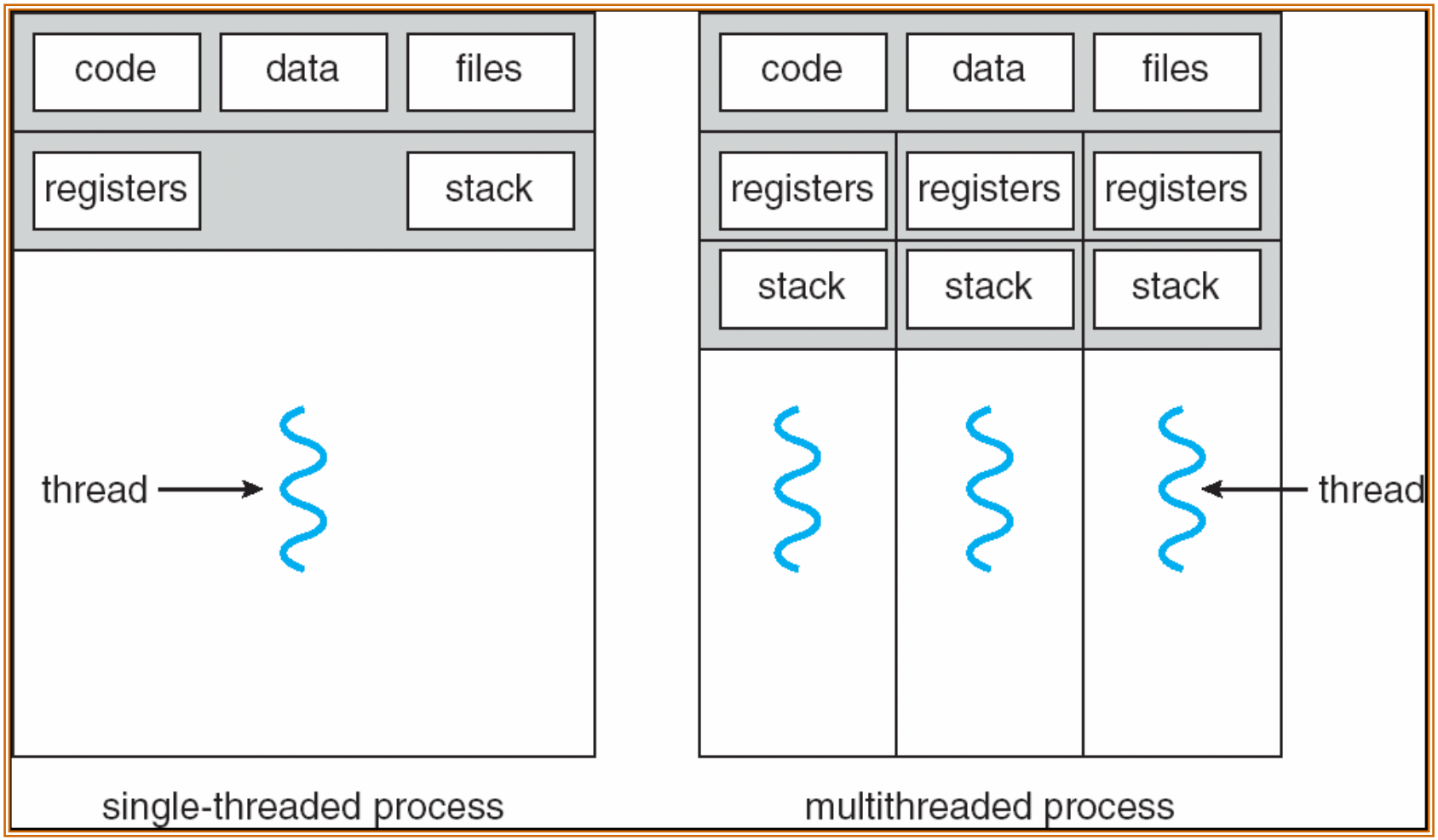
Universidade Federal do ABC

Threads



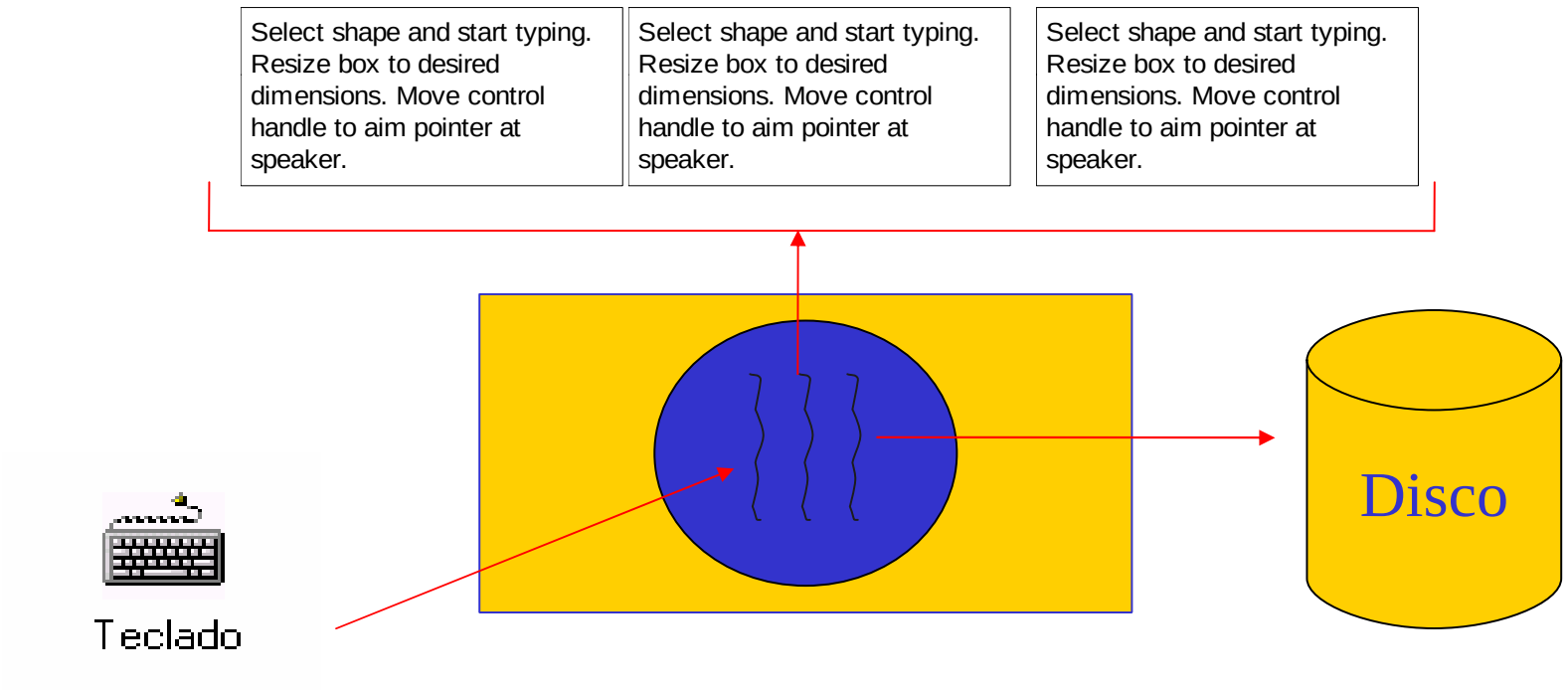
Universidade Federal do ABC

Processos simples ou múltiplos threads



Exemplo: Uso de Threads

- *Threads* para diferentes tarefas;



Universidade Federal do ABC

Exemplo: Uso de Threads

Processador de texto:

- thread para exibir gráfico;
- thread para ler seqüências de teclas do usuário;
- thread para efetuar a verificação ortográfica e gramatical em segundo plano.

- Sistemas computacionais => 3 processos separados não funcionariam, pois todos os três precisam operar sobre o mesmo documento;
- Então, 3 threads são utilizadas para compartilhar uma memória comum e, desse modo, permitir acesso ao documento que está sendo editado pelo usuário.

Universidade Federal do ABC

Exemplo: Uso de Threads

Servidor de arquivos:

- Recebe diversas requisições de leitura e escrita em arquivos e enviar respostas a essas requisições;
  - O servidor mantém um *cache* dos arquivos mais recentes;
  - Lê a *cache* e escreve na *cache* quando possível;

- Quando uma requisição é feita, uma *thread* é alocada para seu processamento.
  - Se essa *thread* for bloqueada devido a E/S - dos arquivos;
  - Outras *threads* podem continuar atendendo as demais solicitações.

Universidade Federal do ABC

Benefícios

Capacidade de resposta:

- Aplicações interativas com *multithreads* permite que um programa continue executando mesmo se parte dele estiver bloqueada, aumentando a capacidade de resposta ao usuário;

  - **Exemplo:**
    - Servidor WEB;
    - Interação com usuário através de uma *thread* enquanto uma imagem é armazenada em outra.

Universidade Federal do ABC

Benefícios

Compartilhamento de recursos:

- Mesmo endereçamento; memória e recursos do processo aos quais pertencem;

- A vantagem de compartilhamento de recurso é permitir que uma aplicação tenha diversos *threads* de atividades diferentes dentro do mesmo espaço de endereço.

Universidade Federal do ABC

Benefícios

Economia:

- Alocação de memória e de recursos para criação de processos é custoso (sobrecarga);
- Threads compartilham recursos do processo ao qual pertence;
- É mais econômico criar e realizar chaveamento de *threads*.
- Exemplo: Diferença de *overhead:*
  - Solaris2
    - Criação de um processo é cerca de 30 vezes mais lento do que a criação de um *thread*;
    - Troca de contexto é cinco vezes mais lenta.

Universidade Federal do ABC

Benefícios

Utilização de arquiteturas multiprocessador:

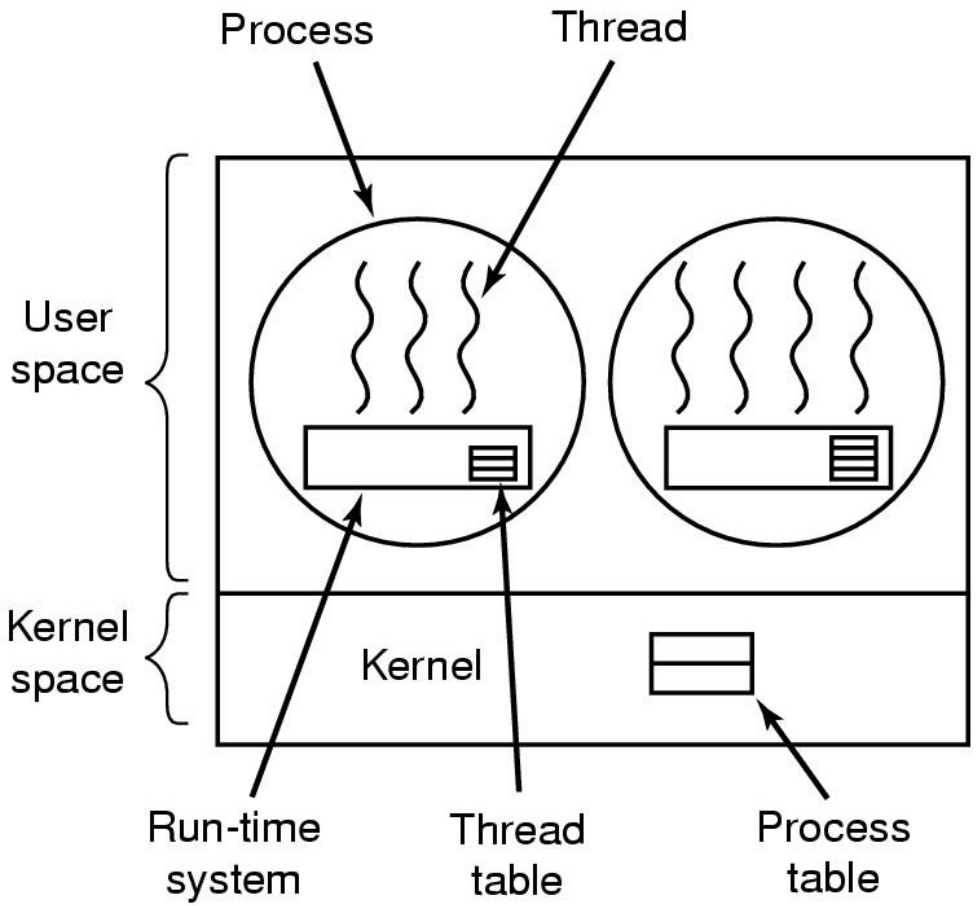
- Vantagem: Cada thread pode ser executado em paralelo em um processador diferente;
- Um processo com um único thread pode executar somente em uma CPU, não importando quanto estejam disponível;
- Multithreads em múltiplas CPUs aumenta a concorrência;
- As Linguagens:
  - **Java, C#, Visual C++ .NET, Visual Basic.NET e Python.**

Modelos de múltiplas threads

Usuário:

- Suportadas acima do kernel e implementadas por bibliotecas no **nível do usuário**;
- Criação e escalonamento são realizados sem o **conhecimento do *kernel***;
  - Tabela de threads para cada processo;
- Processo inteiro é bloqueado se uma *thread* realizar uma chamada bloqueante ao sistema;
  - Exemplo: *Pthreads do POSIX (IEEE 1003.1c).*

Modelos de múltiplas threads



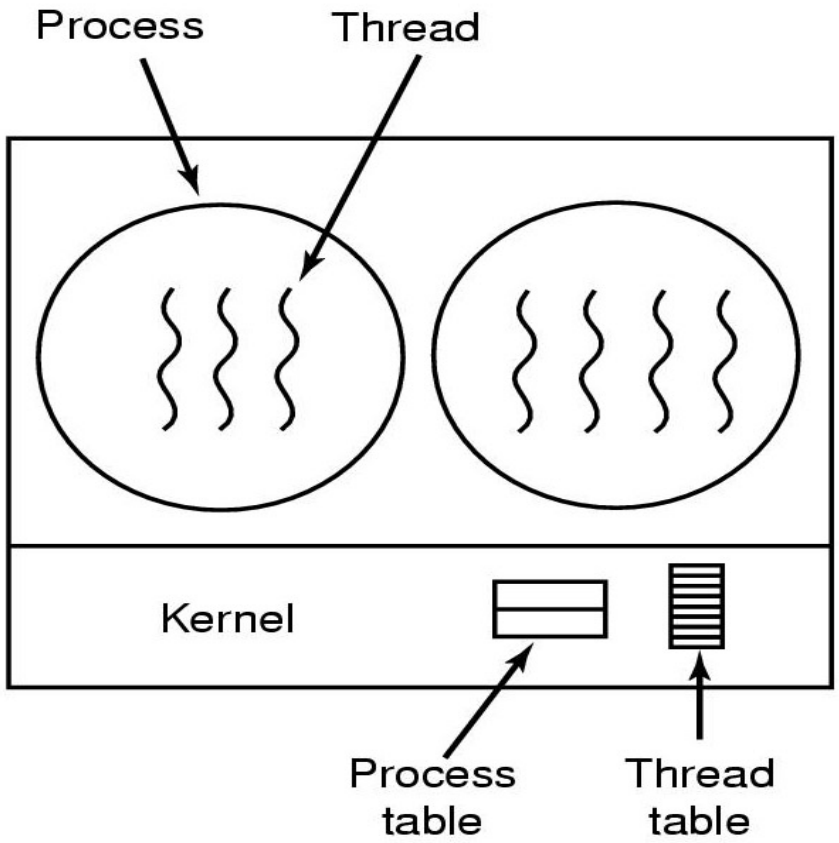
Mapeia todas a threads de um processo multithread para um único contexto de execução

Modelos de múltiplas threads

Kernel:

- Suportadas diretamente pelo SO;
- Criação, escalonamento e gerenciamento feitos pelo *kernel*;
- Trata de forma separada:
  - Tabela de threads – informações das threads usuário
  - Tabela de processos – informações de processos monothreads;
  - Processo não é bloqueado se uma *thread* realizar uma chamada bloqueante ao sistema;
- Gerenciamento de threads de nível kernel é mais lento que de nível usuário:
  - sinais enviados para os processos;

Modelos de múltiplas threads



Tenta resolver as limitações dos threads de usuário mapeando cada thread para seu próprio contexto de execução

Universidade Federal do ABC

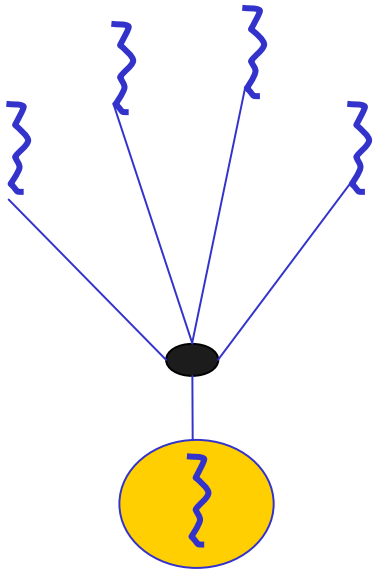
Threads - Kernel

- Quase todos os SOs comtemporâneo implementam esse modelo:
  - Windows XP;
  - Solaris
  - Linux;
  - Tru64 UNIX;
  - Mac OS X.

Modelo de multithreading

Muitos-para-um:

- Mapeia muitas threads de usuário em apenas uma thread de kernel - gerência de threads e feita em nível usuário;
- Não permite múltiplas threads em paralelo;



Threads de usuário

Thread de *kernel*

Exemplo:

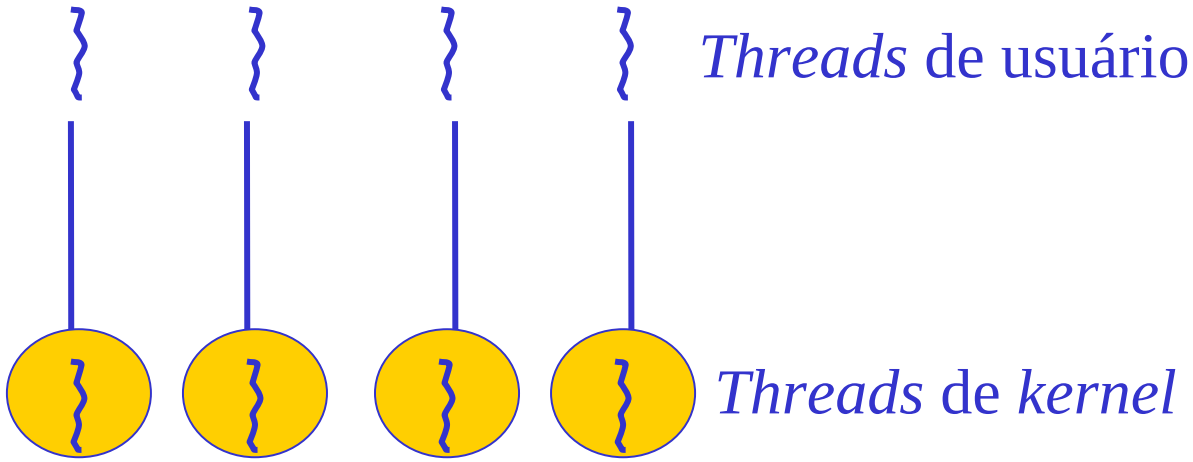
- Green threads:
- threads escalonadas em ambiente virtual em SO nativo.

Universidade Federal do ABC

Modelo de multithreading

Um-para-um:

- Mapeia para cada *thread* de usuário uma *thread* de *kernel*;
- Permite múltiplas *threads* em paralelo;
- Desvantagem: exige de uma thread de usuário crie uma thread de kernel.

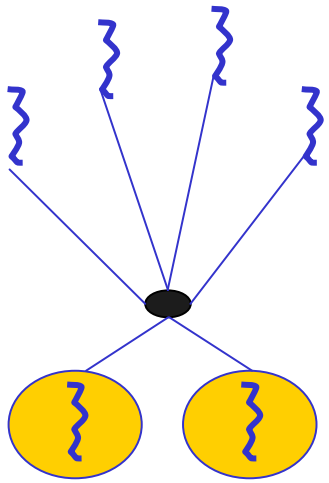


Modelo de multithreading

Universidade Federal do ABC

Muitos-para-muitos:

- Mapeia para múltiplos *threads* de usuário um número menor ou igual de *threads* de *kernel*;
- Permite múltiplas *threads* em paralelo;
- Cuidado para não ter muitas threads dentro de uma aplicação;
- Modelo Hibrido: Muitos para muitos e thread usuário ligada diretamente a nível kernel.



Threads de usuário

Thread de *kernel*

Exemplos :

Solaris 2,

True64 UNIX:

Bibliotecas Threads

- **POSIX Threads (Biblioteca Pthreads)**
  - Define uma API, implementada sobre o SO;
  - Utilizada em sistemas UNIX: Linux, Mac OS X;
  - Padrão IEEE POSIX 1003.1c ;
- **Win 32 Threads**
  - Implementação do modelo um para um no kernel;
- **Java**
  - threads são gerenciadas pela JVM, a qual é executada sobre um SO;
  - JVM especifica a interface com SO;
  - Utiliza uma biblioteca de thread do SO hospedeiro.

Pthreads

- POSIX Threads – Exemplo: Compilação: gcc thrd.c -lpthread

```
#include <pthread.h>
#include <stdio.h>
int sum; /* compartilhado entre as threads*/
void *runner(void *param); /* a thread */

int main(int argc, char *argv[]){
  pthread_t tid; /* identificador da thread*/
  pthread_attr_t attr; /* atributos para a thread */

  if (argc != 2) {
    fprintf(stderr,"usage: a.out <integer value>\n");
    return -1;
  }
```

Biblioteca para trabalhar com thread

Pthreads

- POSIX Threads – Exemplo (1):

```
 if (atoi(argv[1]) < 0) {
  fprintf(stderr,"Argumento %d deve ser não
       negativo\n",atoi(argv[1]));
  return -1;
  }
 /* recebe os atributos */
 pthread_attr_init(&attr);
 /* cria a thread */
 pthread_create(&tid,&attr,runner,argv[1]);
 /* espera a thread finalizar */
 pthread_join(tid,NULL);

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

Universidade Federal do ABC

Pthreads

- POSIX Threads – Exemplo (1):

```
/** A thread começa controlar essa função */
void *runner(void *param) {
 int i, upper = atoi(param);
 sum = 0;

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

  pthread_exit(0);
}
```

Universidade Federal do ABC

Threads em Java

- Fornece um ambiente abstrato, que permite aos programas Java executar em qualquer plataforma com o JVM;
  - Threads são gerenciadas pelo JVM;
  - Pacote java.lang.Thread;
- Duas formas de manipulação:
  - Criar uma nova classe derivada da **classe Thread**;
  - Classe que implemente a **interface Runnable** (mais utilizada);
  - Nos dois casos o programador deve apresentar uma implementação para o método run( ), que é o método principal da thread.

Universidade Federal do ABC

Threads em Java

- Exemplo (1):

```
public class Trabalhador extends Thread {
     String nome, produto;
     int tempo;

  public Trabalhador(String nome, String produto, int
   tempo) {
     this.nome = nome;
     this.produto = produto;
     this.tempo = tempo;
  }
```

Universidade Federal do ABC

Threads em Java

- Exemplo (1):

```
public void run() {
  for (int i=0; i<50; i++) {
   try {
     Thread.sleep(tempo);
   } catch (InterruptedException ex) {
       ex.printStackTrace();
   }
  System.out.println(nome+"produziu o(a)"+i+ "°"+ produto);
}
```

Universidade Federal do ABC

Threads em Java

- Exemplo (1):

```
public static void main(String[] args) {
   Trabalhador t1 = new Trabalhador("Mario", "bota", 1000);
   Trabalhador t2 = new Trabalhador("Sergio", "camisa",
   2000);

   t1.start();
   t2.start();
}
```

Universidade Federal do ABC

Threads em Java

- Exemplo (2):

```
class MutableInteger{
 private int value;
 public int get() {return value; }
 public void set(int sum) {
   this.value = sum;
 }
}
```

```
class Summation implements Runnable{
  private int upper;
  private MutableInteger sumValue;
```

Universidade Federal do ABC

Threads em Java

- Exemplo (2):

```
public Summation(int upper, MutableInteger sumValue) {
    if (upper < 0)
           throw new IllegalArgumentException();
    this.upper = upper;
    this.sumValue = sumValue;
   }
```

```
public void run() {
   int sum = 0;
   for (int i = 0; i <= upper; i++)
     sum += i;
   sumValue.set(sum);
   }
}
```

Universidade Federal do ABC

Threads em Java

- Exemplo (2):

```
public class Driver{
   public static void main(String[] args){
     if (args.length != 1){
      System.err.println("Usage Driver <integer>");
      System.exit(0);
     }
    MutableInteger sumObject = new MutableInteger();
   int upper = Integer.parseInt(args[0]);
    Thread worker = new Thread(new Summation(upper,
       sumObject));
    worker.start();
```

Universidade Federal do ABC

Threads em Java

- Exemplo (2):

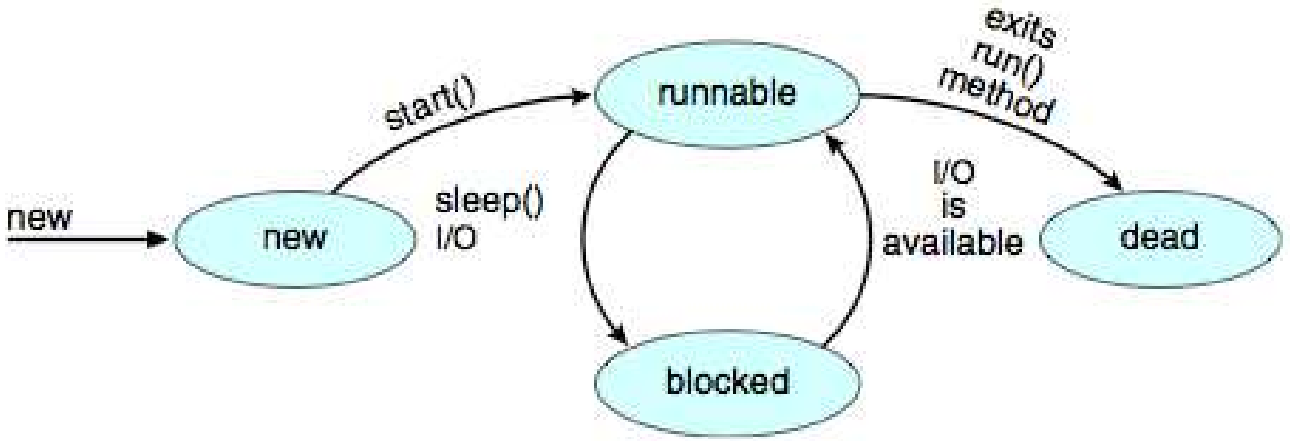
```
   try {
   worker.join();
   } catch (InterruptedException ie) {


    System.out.println("The value of " + upper + " is " +
   sumObject.get());
 }
}
```

Universidade Federal do ABC

Threads em Java

- As threads podem estar em um dos 4 estados:
  - **Novo:** quando é criada (new);
  - **Executável:** método start( ) para alocar memória e chama o método run ( );
  - **Bloqueado:** utilizado para realizar uma instrução de bloqueo ou para invocar certos métodos como sleep( );
  - **Morto:** passa para esse estado quando termina o método run();



Universidade Federal do ABC

Aspectos do uso de threads

Semântica do fork() e exec()

- fork( ) cria novo processo=>sistemas UNIX podem ter duas versões de fork( ):
  - Uma que duplica todas as threads
  - Outra que duplica apenas a thread que invocou a chamada de sistema fork()
- fork( ) seguida de exec( ):
  - é feita apenas a duplicação da thread de chamada;
- fork( ) não seguida de exec( ):
  - é feita a duplicação de todas as threads;
- Chamada de sistema exec( ) isoladas:
  - em geral substituem o processo inteiro, incluindo todas as suas threads.

Cancelamento de Thread

- Corresponde à tarefa de terminar um thread antes que se complete.
- Exemplos:
  - multiplas threads pesquisando em banco de dados;
  - Navegador web acessando uma página.
- Denominada **thread alvo** e pode ocorrer em dois diferentes cenários:
  - **Cancelamento assíncrono**: um thread imediatamente termina o thread-alvo. Pode não liberar os recursos necessários a nível de sistema.
  - **Cancelamento adiado** permite o thread alvo ser periodicamente verificado se deve ser cancelada;

Cancelamento de Thread

Cancelamento adiado em Java - Interromper a thread

```
Thread thrd = new Thread(new InterruptibleThread());
thrd.start();
. . .
thrd.interrupt();
```

Cancelamento de Thread

- Cancelamento adiado em Java - Verificando status de interrupção

```
class InterruptibleThread implements Runnable
{
    /**
     * This thread will continue to run as long
     * as it is not interrupted.
     */
    public void run() {
        while (true) {
            /**
             * do some work for awhile
             * . . .
             */

            if (Thread.currentThread().isInterrupted()) {
                System.out.println("I'm interrupted!");
                break;
            }
        }
        // clean up and terminate
    }
}
```

Universidade Federal do ABC

Tratamento de Sinais

- Sinais são usados nos sistemas UNIX para notificar um processo de que um evento específico ocorreu;
- Um sinal pode ser:
  - Síncrono: se forem liberados para o mesmo processo que provocou o sinal;
    - Exemplo: processo executa divisão por 0 e recebe sinal de notificação.
  - Assíncrono: se forem gerados por um evento externo (ou outro processo) e entregues a um processo;

Universidade Federal do ABC

Tratamento de Sinais

- Sinais para processos comuns
  - São liberados apenas para o processo específico (PID);
- Sinais para processos multithread: várias opções
  - Liberar o sinal para a thread conveniente (ex.: a que executou divisão por zero);
  - Liberar o sinal para todas as threads do processo (ex.: sinal para término do processo);
  - Liberar o sinal para determinadas threads;
  - Designar uma thread específica para receber todos os sinais.

Universidade Federal do ABC

Cadeias de Threads

- A criação/finalização de threads traz alguns problemas: overhead:
  - Caso do servidor web: recebe muitas conexões por segundo:
  - Criar e terminar inúmeras threads é um trabalho muito grande;
  - Trabalha-se com um número ilimitado de threads: término dos recursos

Cadeias de Threads

- Solução: utilizar cadeia de threads em espera
  - Na inicialização do processo, cria-se um número adequado de threads
  - As threads permanecem aguardando para entrar em funcionamento
  - Quando o servidor recebe uma solicitação, desperta uma thread da cadeia (se houver disponível, senão espera) e repassa trabalho
  - Quando thread completa serviço, volta à cadeia de espera
- Vantagens: mais rápido que criar thread, limita recursos

Universidade Federal do ABC

Threads em Linux

- O Linux refere-se a elas como tarefas, em vez de threads;
- A criação de thread é feita através da chamada de sistema clone( );

flag	meaning
CLONE_FS	File-system information is shared.
CLONE_VM	The same memory space is shared.
CLONE_SIGHAND	Signal handlers are shared.
CLONE_FILES	The set of open files is shared.

Universidade Federal do ABC

Aula 04 - Sumário

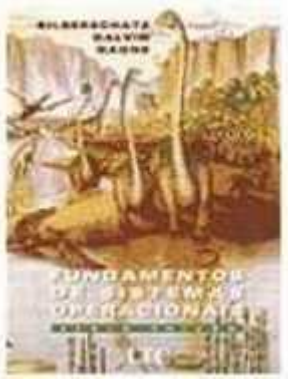
- Uma thread é a unidade básica utlizada pela CPU, onde um processo é composto de uma ou mais threads;
- Cada thread tem: registradores, pilha, contadores;
- As threads compartilham: código, dados, e recurso do SO, como arquiivo aberto;
- Threads de nível usuário e nível kernel;
- Modelos de mapeagemnto de threads usuário para kernel:
  - N-para -1, 1-para-1, N-para-N.
- Bibliotecas: Pthreads, Win32, Java

Universidade Federal do ABC

Leituras Sugeridas

- Silberschatz, A., Galvin, P. B. Gagne, G. Sistemas Operacionais com Java. 7º , edição. Editora, Campus, 2008 .

- Silberschatz, Abraham; Galvin, Peter Baer; Gagne, Greg. **Fundamentos de sistemas operacionais.** 6 ed. Rio de Janeiro: LTC, 2009.



- https://computing.llnl.gov/tutorials/pthreads/

Universidade Federal do ABC

Nota de Aula

- Acesse o link abaixo:

http://hostel.ufabc.edu.br/~marcelo.nascimento/

Obrigado!!!