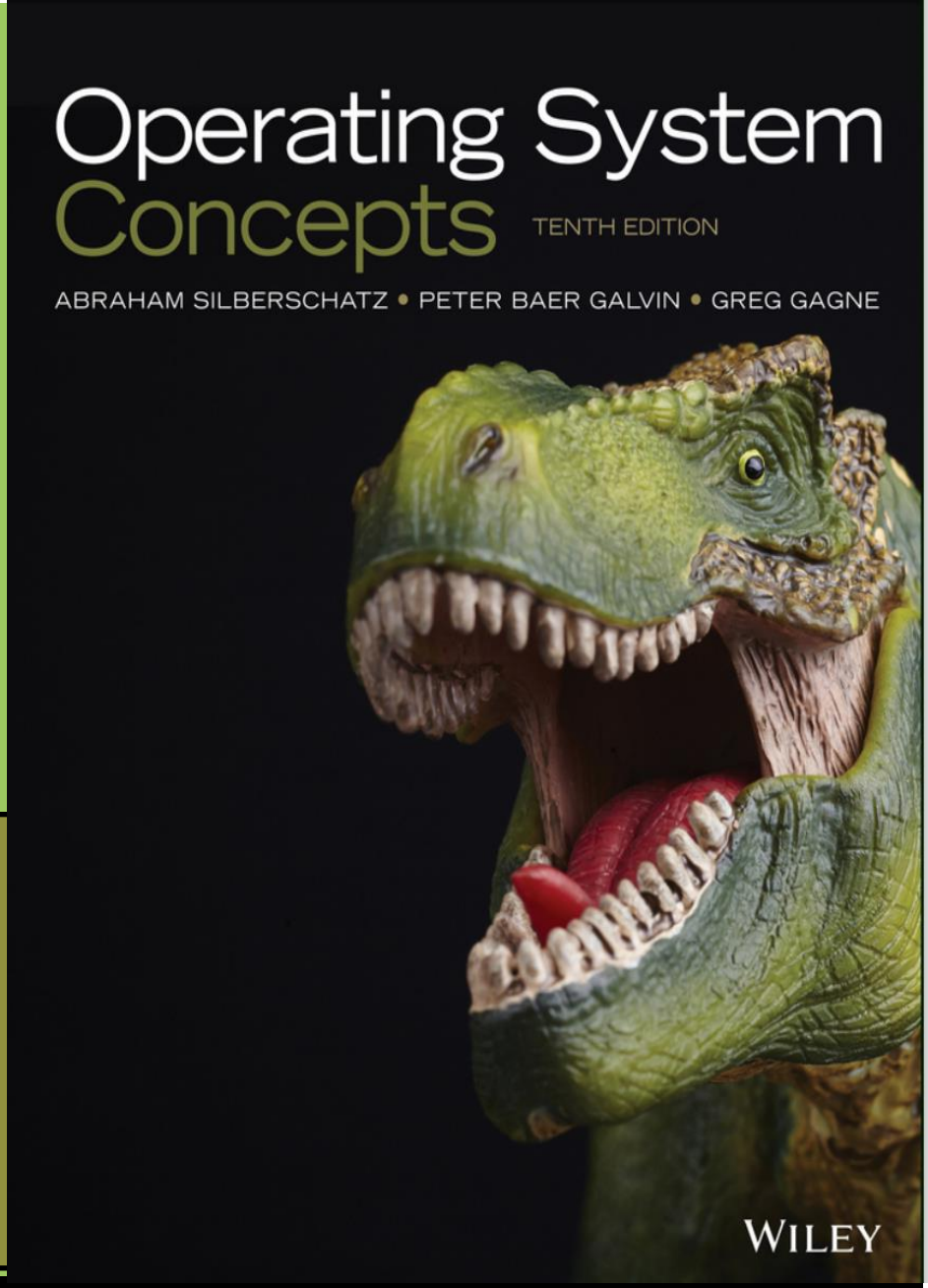
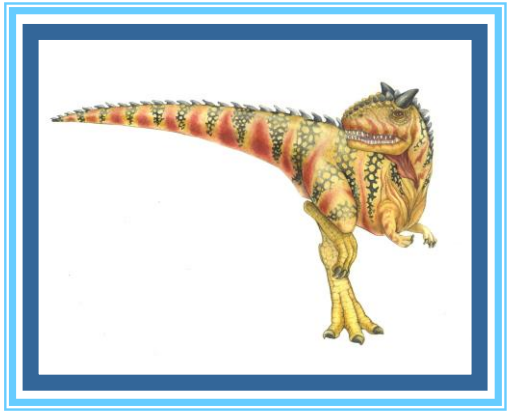
Operation Systems

Dr. A. Taghinezhad



Website: ataghinezhad.github.io, Email:a0taghinezhad@gmail.com

فصل ۴: رشته‌ها و همزمانی



Outline

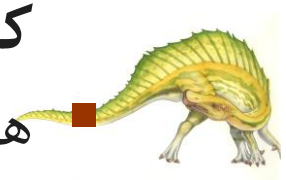
- برنامه نویسی چند هسته ای (Multicore Programming)
- مدل های چند رشته ای (Multithreading Models)
- کتابخانه های ترد (Thread Libraries)
- رشته بندی ضمنی (Implicit Threading)
- مسائل مربوط به چند رشته ای(Threading Issues)
- نمونه های سیستم عامل(Operating System Examples)

Objectives

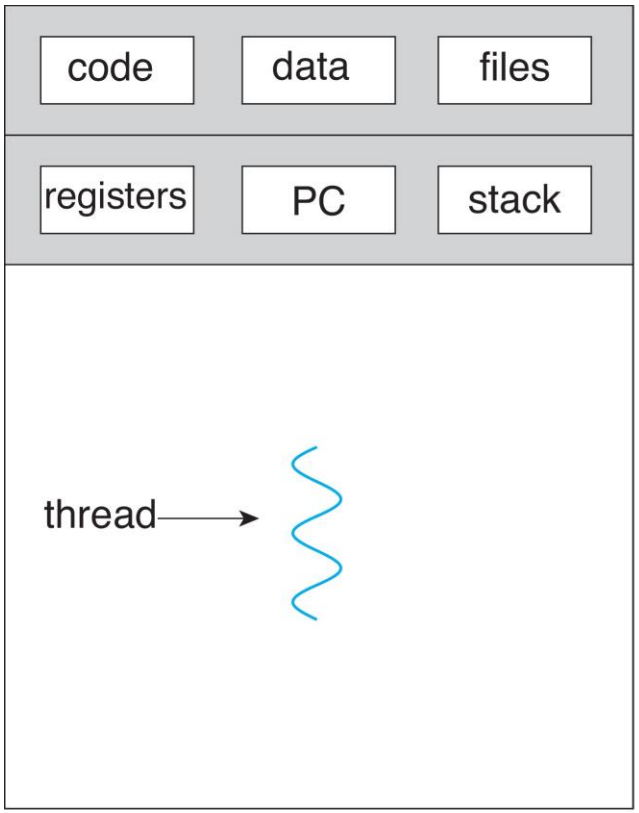
- اجزای اصلی یک ترد را شناسایی کنید و تردها را با فرایندها مقایسه نمایید.
- توضیح دهید که طراحی برنامه های چند رشته ای چه مزایا و چالش هایی به همراه دارد.
- رویکردهای مختلف برای رشته بندی ضمنی از جمله thread pool ها، fork-joinو Grand Central Dispatch را شرح دهید.
- نحوه نمایش تردها در سیستم عامل های ویندوز و لینوکس را توضیح دهید.
- با استفاده از API های رشته بندیPthreads ، جاوا و ویندوز، نحوه طراحی برنامه های چند رشته ای را شرح دهید.

Motivation

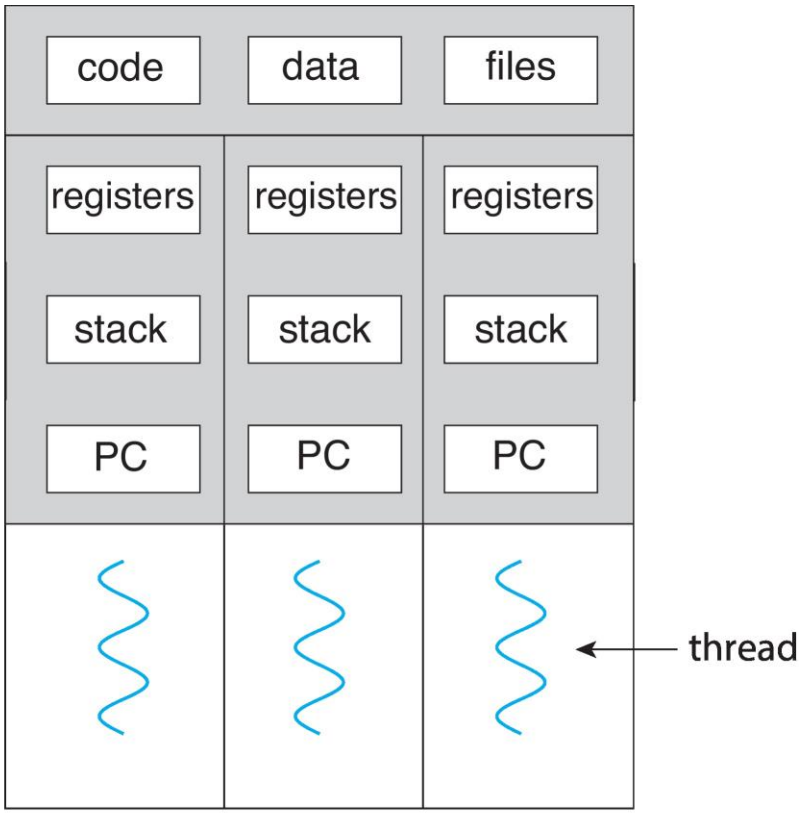
- اکثر برنامه های کاربردی مدرن چند رشته ای هستند.
- تردها درون یک برنامه اجرا می شوند.
- وظایف متعدد یک برنامه را می توان با تردهای جداگانه اجرا کرد.
  - به روز رسانی رابط کاربری
  - دریافت داده ها
  - بررسی املای متن
  - پاسخ به درخواست شبکه
- ایجاد فرایند یک کار سنگین است، در حالی که ایجاد ترد سبک وزن است.
- استفاده از تردها می تواند باعث **ساده سازی کد** و افزایش **کارایی** برنامه شود.
- هسته های سیستم عامل به طور کلی چند رشته ای هستند.



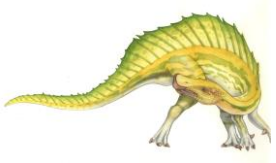
Single and Multithreaded Processes



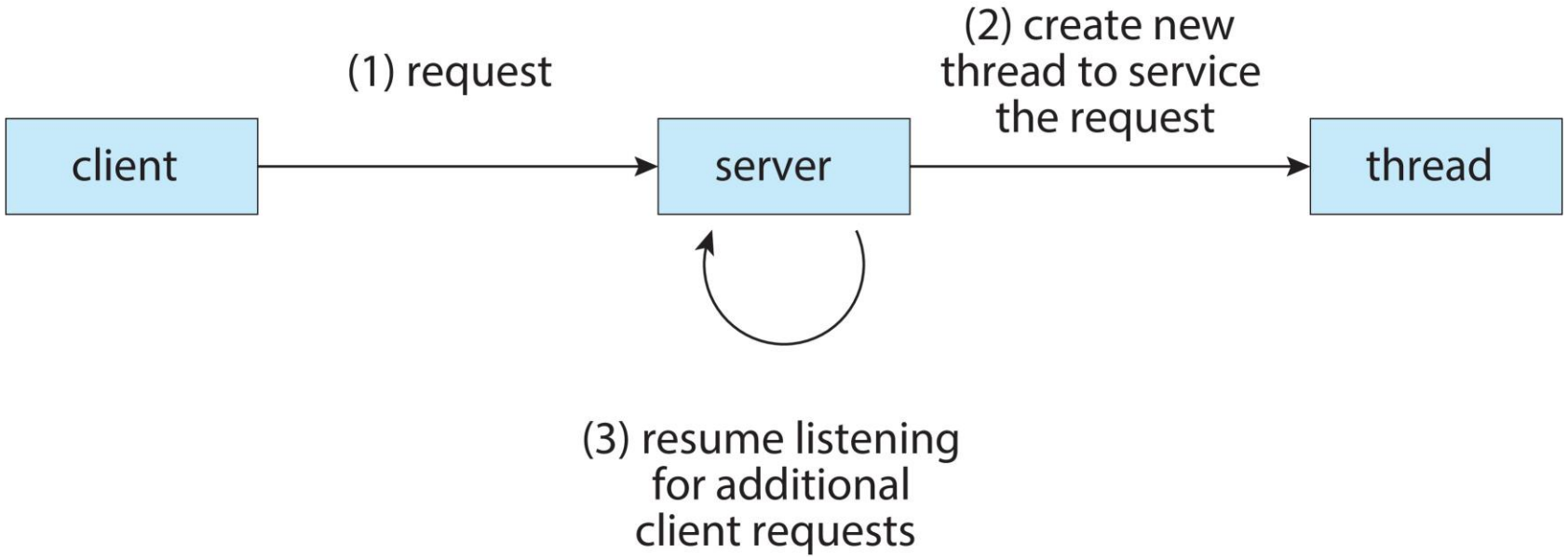
single-threaded process



multithreaded process



Multithreaded Server Architecture



مزایای چند رشته ای بودن

- اکثر برنامه های کاربردی مدرن چند رشته ای هستند.
- تردها درون یک برنامه اجرا می شوند.
- وظایف متعدد یک برنامه را می توان با تردهای جداگانه اجرا کرد.
  - به روز رسانی رابط کاربری
  - دریافت داده ها
  - بررسی املای متن
  - پاسخ به درخواست شبکه
- ایجاد فرایند یک کار سنگین است، در حالی که ایجاد ترد سبک وزن است.
- استفاده از تردها می تواند باعث ساده سازی کد و افزایش کارایی برنامه شود.
- هسته های سیستم عامل به طور کلی چند رشته ای هستند.

Multicore Programming

- سیستم های چند هسته ای یا چند پردازنده برنامه نویسان را با چالش هایی مواجه می کنند که برخی از آنها عبارتند از:
  - تقسیم فعالیت‌ها
  - توازن
  - تقسیم‌بندی داده ها
  - وابستگی داده‌ها
  - تست و اشکال‌زدایی

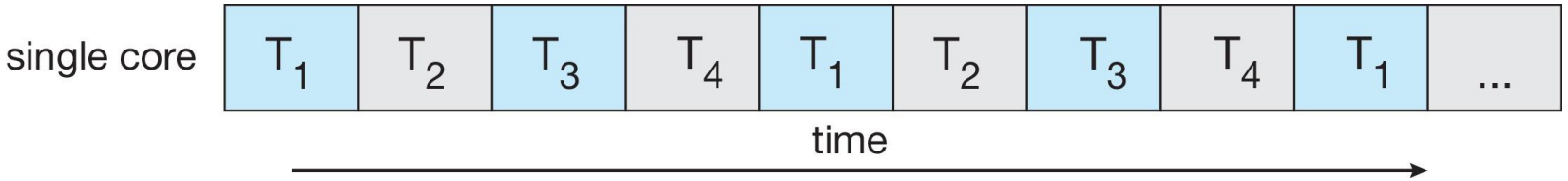
Multicore Programming

- **موازی کاری در مقابل همزمانی (Parallelism vs Concurrency):**

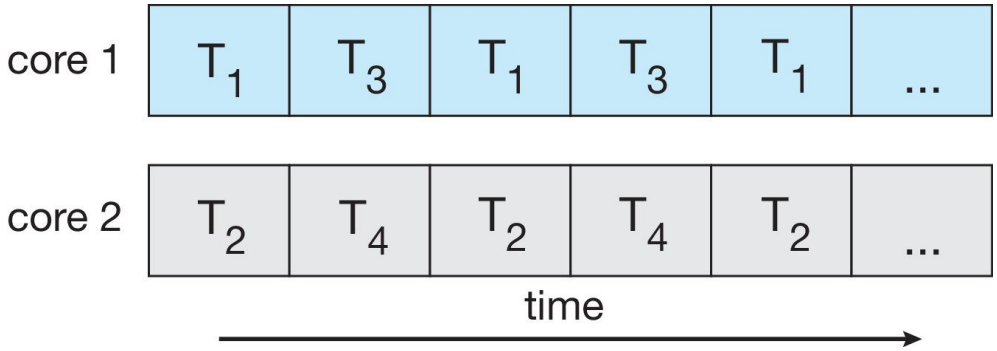
- **موازی کاری :(Parallelism)** این مفهوم به توانایی یک سیستم برای انجام بیش از یک کار به طور همزمان اشاره دارد.
- **همزمانی :(Concurrency)** این مفهوم به توانایی یک سیستم برای پیشبرد بیش از یک کار اشاره دارد. حتی در یک پردازنده یا هسته واحد، زمانبندی کننده سیستم می تواند همزمانی را فراهم کند.

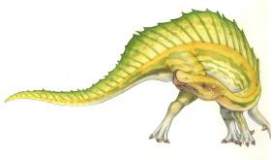
Concurrency vs. Parallelism

- **Concurrent execution on single-core system:**



- **Parallelism on a multi-core system:**

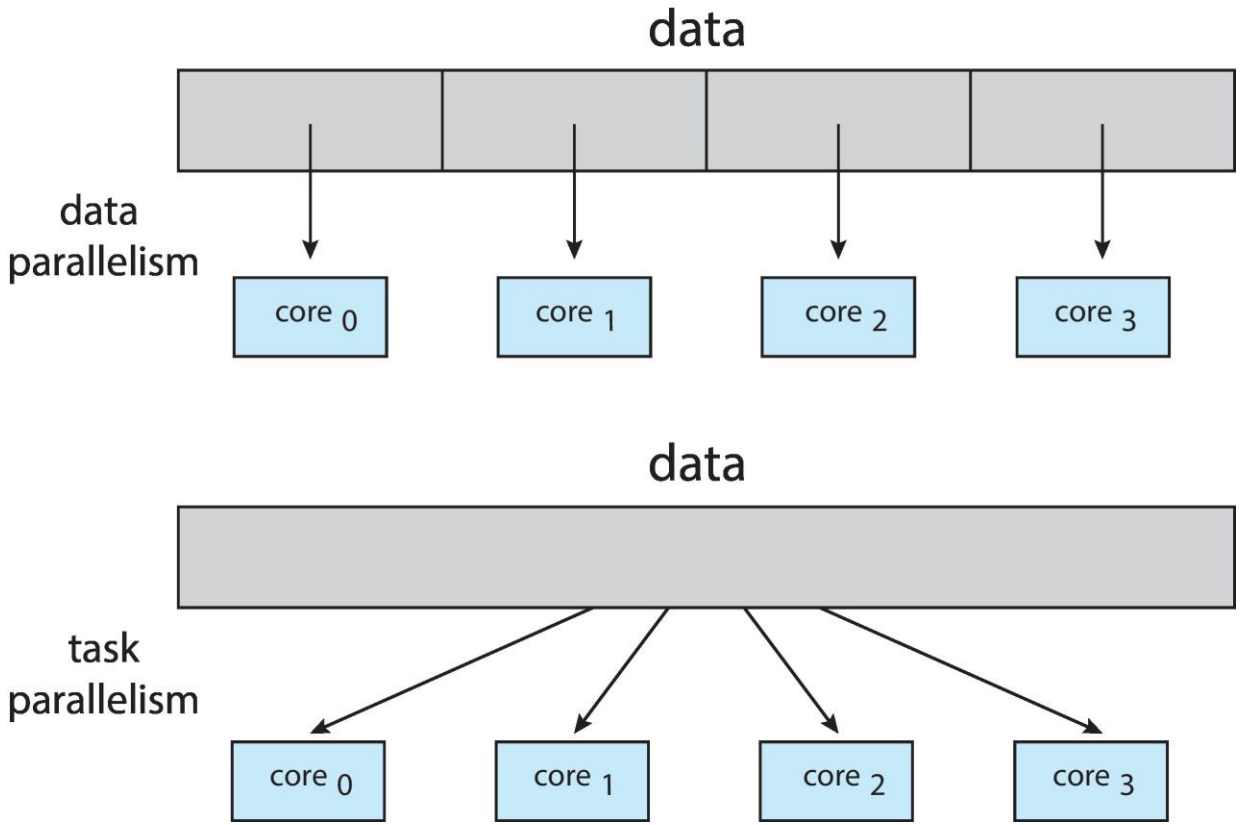




Multicore Programming

- **موازی کاری داده (Data parallelism):** توزیع زیرمجموعه هایی از داده های مشابه در هسته های مختلف، به طوری که عملیات یکسانی روی هر مجموعه داده انجام شود.
- **موازی کاری وظیفه (Task parallelism):** توزیع تردها در هسته های مختلف، به طوری که هر ترد یک عملیات منحصر به فرد را انجام دهد.

Data and Task Parallelism



قانون امدل

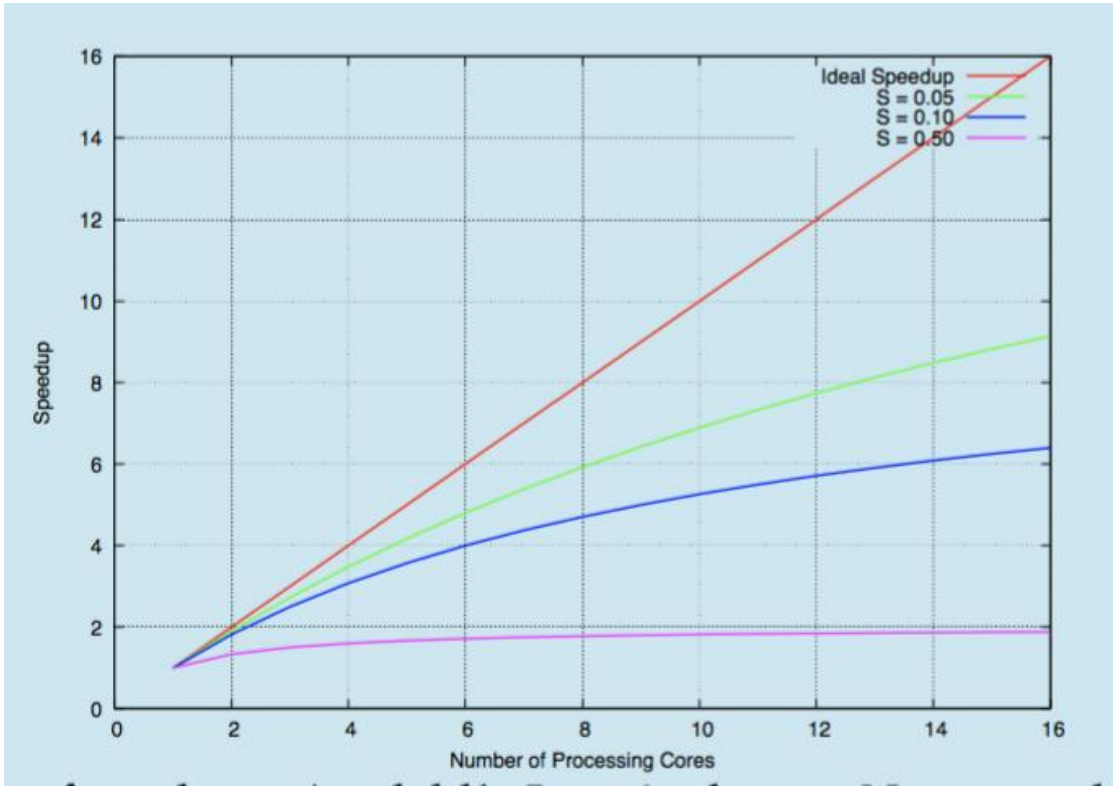
- این قانون، سود بالقوه ناشی از اضافه کردن هسته های اضافی به یک برنامه کاربردی را که دارای اجزای سریال و موازی است، مشخص می کند.

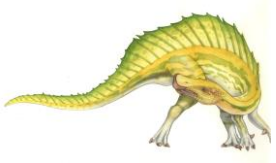
$$speedup \leq \frac{1}{S + \frac{(1-S)}{N}}$$

- Sبخش سریال برنامه است.
- Nتعداد هسته های پردازنده است.

- به عبارت دیگر، اگر یک برنامه کاربردی ۷۵ درصد موازی و ۲۵ درصد سریال باشد، انتقال از ۱ هسته به ۲ هسته منجر به افزایش سرعت ۱.۶ برابری می شود.
- با نزدیک شدن N به بی نهایت، افزایش سرعت به ۱ / S نزدیک می شود.
- بخش سریال یک برنامه تاثیر نامتناسبی بر روی بهبود عملکرد ناشی از اضافه کردن هسته‌های اضافی دارد.

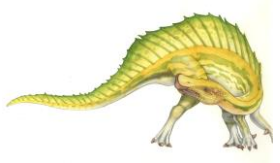
Amdahl's Law


Ideal Speedup
S = 0.05
S = 0.10
S = 0.50
Speedup
Number of Processing Cores
0
2
4
6
8
10
12
14
16

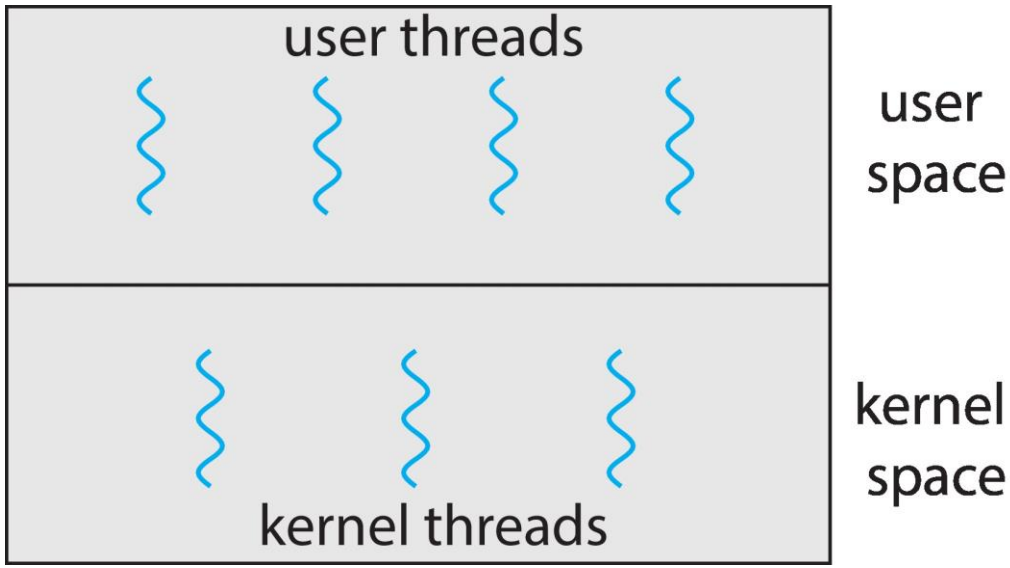


User Threads and Kernel Threads

- **تردهای کاربر (User Threads):** مدیریت این نوع ترد توسط کتابخانه ترد در سطح کاربر انجام می شود.
  - نمونه ها:
    - تردهای POSIX (Pthreads)
    - تردهای ویندوز
    - تردهای جاوا
- **تردهای هسته (Kernel Thread):** این نوع ترد توسط هسته سیستم عامل پشتیبانی می شود.
  - نمونه ها:
    - تقریباً تمام سیستم عامل های عمومی، از جمله:
      - ویندوز – لینوکس-مکینتاش-اندروید



User and Kernel Threads



مدل‌های چندرشته‌ای

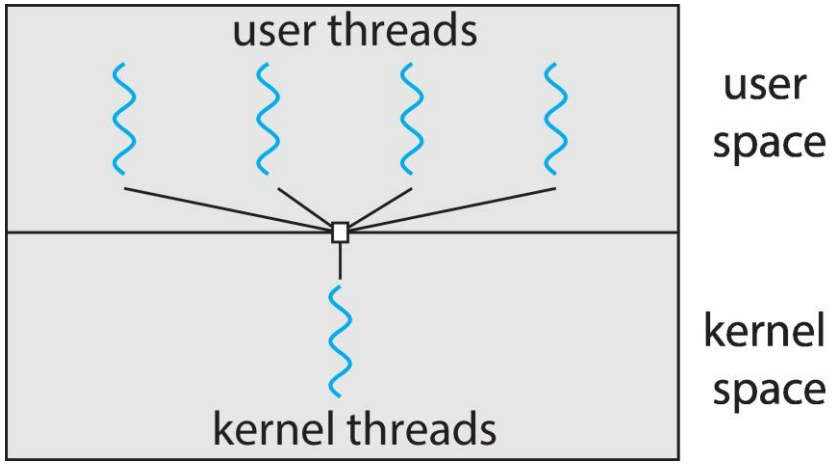
- Many-to-Oneچند به یک
- One-to-One یکی به یک
- Many-to-Many چند به چند

Many-to-One

- **تردهای کاربر :(User Threads)** مدیریت این نوع ترد توسط کتابخانه ترد در سطح کاربر انجام می شود.
  - نمونه ها:
    - تردهای POSIX (Pthreads)
    - تردهای ویندوز
    - تردهای جاوا
- **تردهای هسته :(Kernel Thread)** این نوع ترد توسط هسته سیستم عامل پشتیبانی می شود.
  - نمونه ها:
    - تقریباً تمام سیستم عامل های عمومی، از جمله:
      - ویندوز
      - لینوکس
      - مکینتاش
      - iOS
      - اندروید

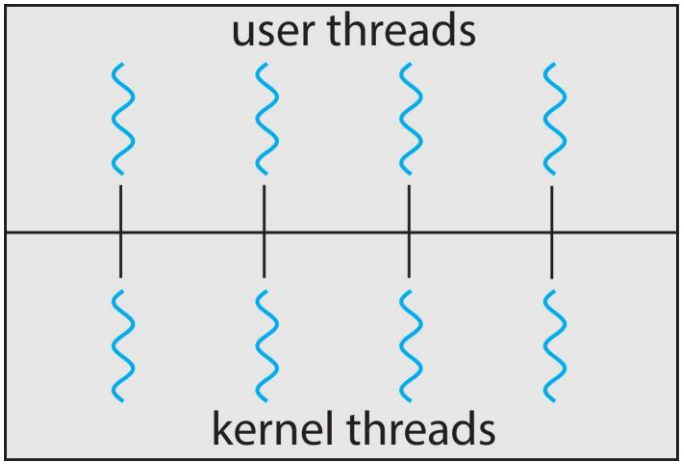
Many-to-One

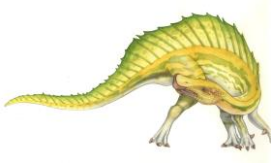
- مدل چند به یک (شکل ۴,۷) رشته‌های زیادی در سطح کاربر را به یک رشته هسته نگاشت می‌کند.
- بدلیل اینکه مدیریت رشته‌ها توسط کتابخانه رشته در فضای کاربر انجام می‌شود، این روش کارآمد است (کتابخانه‌های رشته در بخش ۴,۴ مورد بحث قرار خواهند گرفت).
- با این حال، کل فرایند در صورتی که یک رشته فراخوانی سیستم مسدود کننده انجام دهد، مسدود خواهد شد. همچنین، از آنجایی که فقط یک رشته می‌تواند به هسته در یک زمان دسترسی پیدا کند، چندین رشته قادر به اجرای موازی در سیستم‌های چند هسته‌ای نیستند.



One-to-One

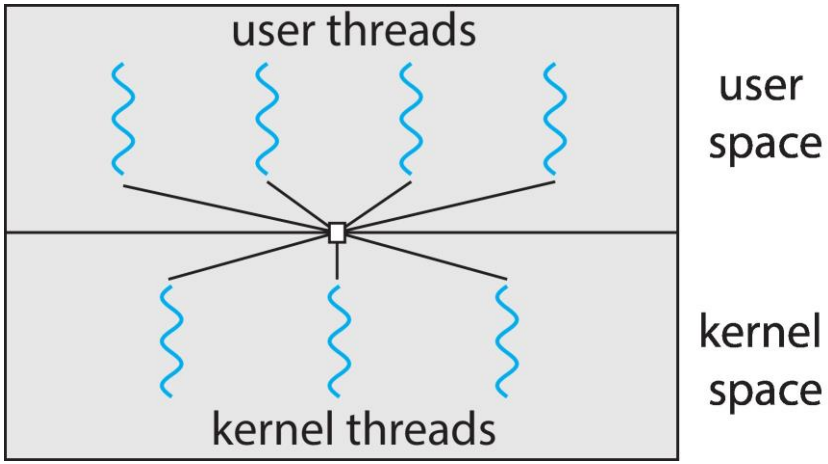
- هر ترد کاربر به یک ترد هسته نگاشت می شود.
- ایجاد یک ترد کاربر، یک ترد هسته ایجاد می کند.
- همزمانی (Concurrency) بیشتری نسبت به مدل چند به یک (Many-to-One)دارد.
- تعداد زیاد تردهای هسته به ازای هر فرایند گاهی اوقات به دلیل سربار (overhead)محدود می شود.
- نمونه ها:
  - ویندوز
  - لینوکس

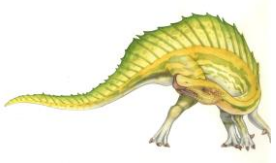




Many-to-Many Model

- اجازه می‌دهد چندین نخ در سطح کاربر به چندین نخ (کمتر یا برابر) در سطح هسته نگاشت شوند.
- این مدل به سیستم عامل اجازه می دهد تا تعداد کافی ترد کاربر ایجاد کند. اما این کار منجر به موازی‌سازی نمی‌شود. زیرا هسته فقط می‌تواند یک رشته هسته را در یک زمان زمان‌بندی کند.





Thread Libraries

- کتابخانه ترد، API برای ایجاد و مدیریت تردها را برای برنامه نویس فراهم می کند.
- دو روش اصلی برای پیاده سازی وجود دارد:
  - کتابخانه به طور کامل در فضای کاربر
  - کتابخانه سطح هسته که توسط سیستم عامل پشتیبانی می شود

Pthreads

- ممکن است به صورت سطح کاربر یا سطح هسته ارائه شود.
- یک API استاندارد POSIX (IEEE 1003.1c) است برای ایجاد و همگام سازی ترد
  - APIرفتار کتابخانه ترد را مشخص می کند، پیاده سازی به توسعه کتابخانه بستگی دارد.
  - رایج در سیستم عامل های یونیکس (لینوکس و مکینتاش)
  - Pthreads به‌صورت مستقیم در ویندوز پشتیبانی نمی‌شوند ولی یک برنامه شخص‌ثالث برای آن وجود دارد.

Pthreads Example

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

#include <stdlib.h>

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

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

   /* set the default attributes of the thread */
   pthread_attr_init(&attr);
   /* create the thread */
   pthread_create(&tid, &attr, runner, argv[1]);
   /* wait for the thread to exit */
   pthread_join(tid,NULL);

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

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

    for (i = 1; i <= upper; i++)
       sum += i;

    pthread_exit(0);
 }
```

Java Threads

- تردهای جاوا توسط JVM مدیریت می شوند.
  - به طور معمول با استفاده از مدل تردهایی که توسط سیستم عامل زیربنایی ارائه می شود، اجرا می شوند.
  - تردهای جاوا ممکن است با موارد زیر ایجاد شوند:
    - توسعه (Extend) کردن کلاس Thread
    - پیاده سازی رابط Runnable

```
public interface Runnable
{
    public abstract void run();
}
```

- رویه استاندارد، پیاده سازی رابط Runnable است.

Java Threads

Implementing Runnable interface:

```
class Task implements Runnable
{
   public void run() {
      System.out.println("I am a thread.");
   }
}
```

Creating a thread:

```
Thread worker = new Thread(new Task());
worker.start();
```

Waiting on a thread:

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