Threads

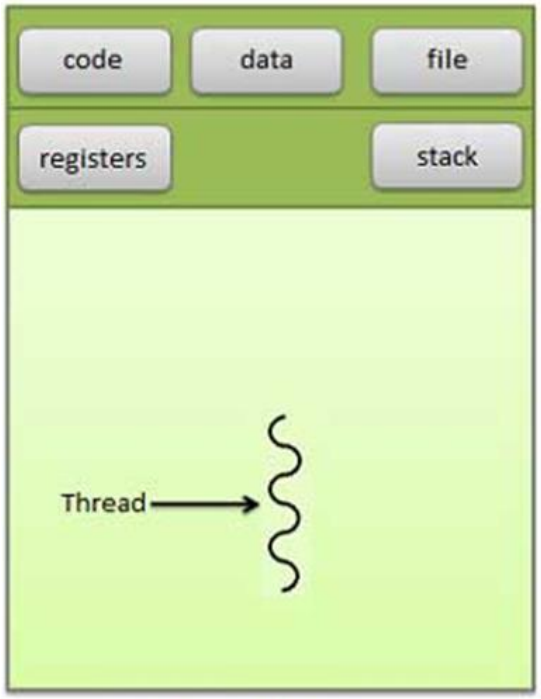
CS3600 Spring2022

Threads

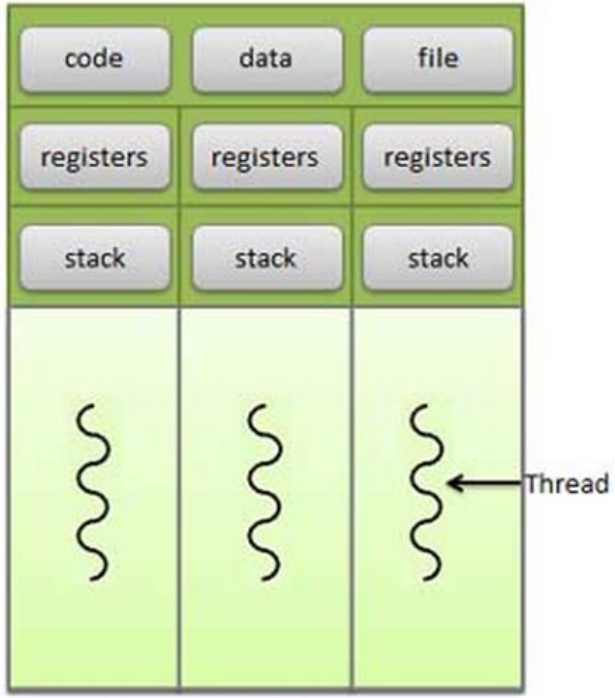
- Threads –Subsets of Processes

- Some Properties of threads
  - An active entity.
  - Works simultaneously .
  - A basic unit of CPU utilization
  - Requires Coordination.

Single and Multithread

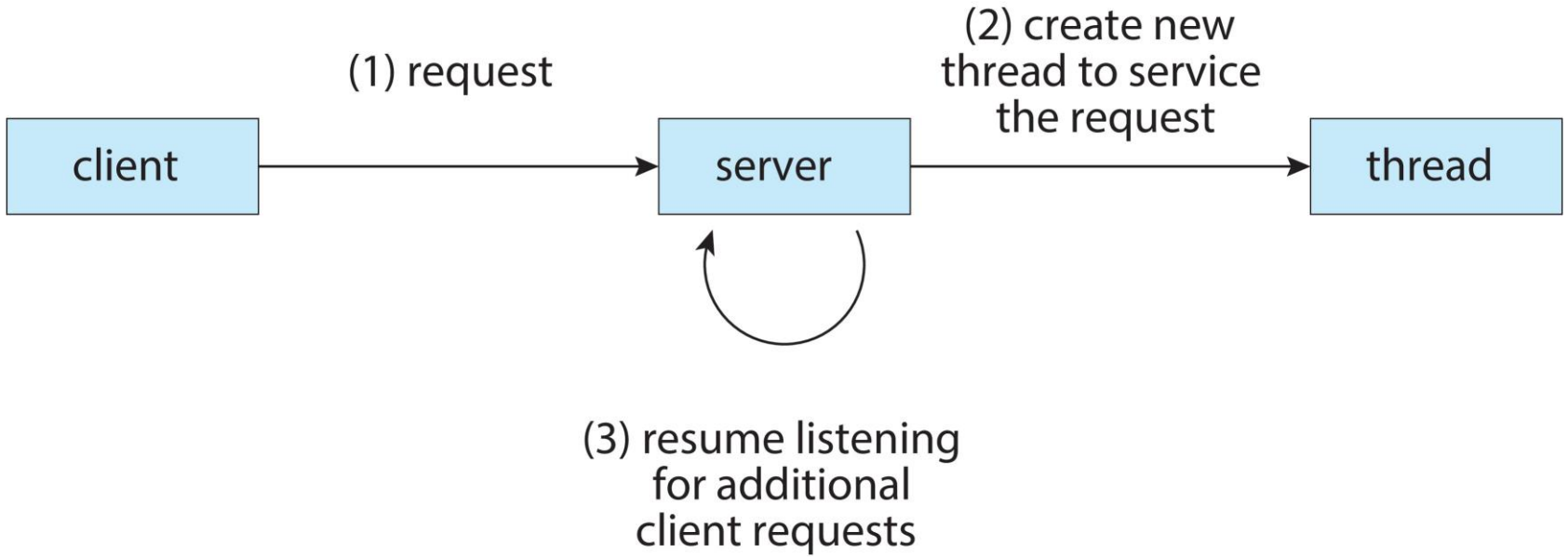


Single threaded Process



Multi-threaded Process

Multithreaded Server Architecture

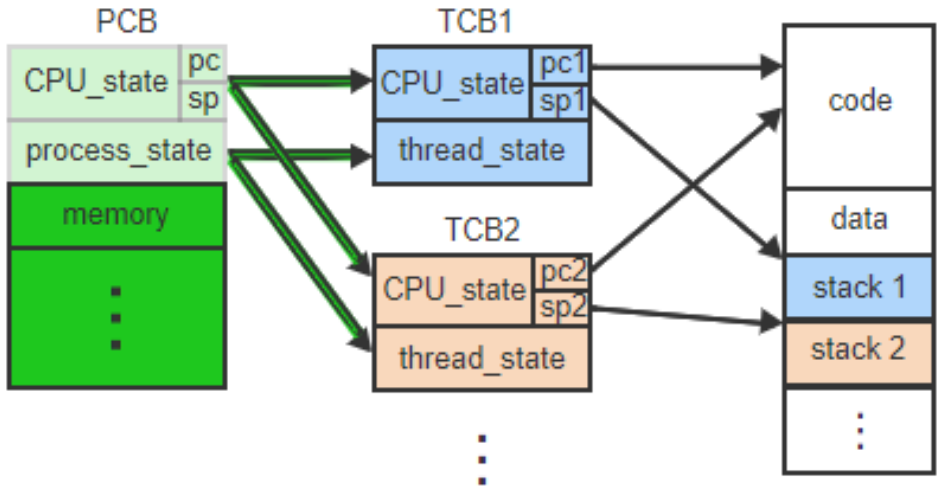


Benefits

- **Responsiveness** – may allow continued execution if part of process is blocked, especially important for user interfaces.
- **Resource Sharing** – threads share resources of process, easier than shared memory or message passing.
- **Economy** – cheaper than process creation, thread switching lower overhead than context switching.
- **Scalability** – process can take advantage of multicore architectures

TCB

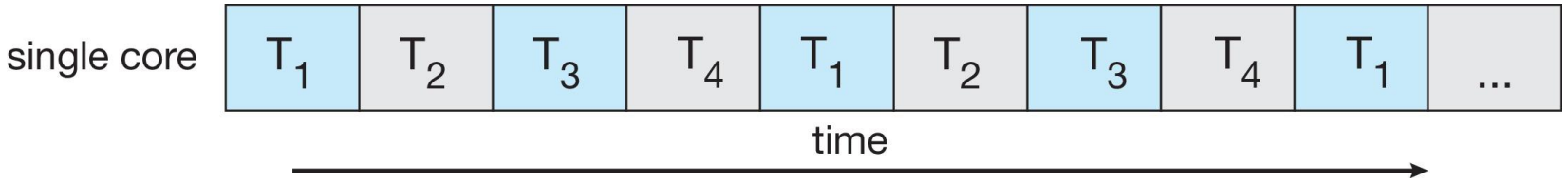
- A ***Thread control block (TCB)*** is a data structure that holds a separate copy of the dynamically changing information necessary for a thread to execute independently.



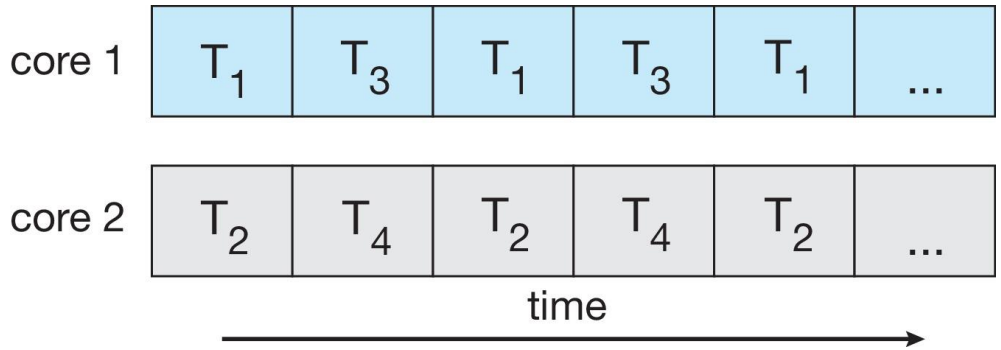
- The replication of only the bare minimum of information in each TCB, while sharing the same code, global data, resources, and open files, is what makes threads much more efficient to manage than processes.

Concurrency vs. Parallelism

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

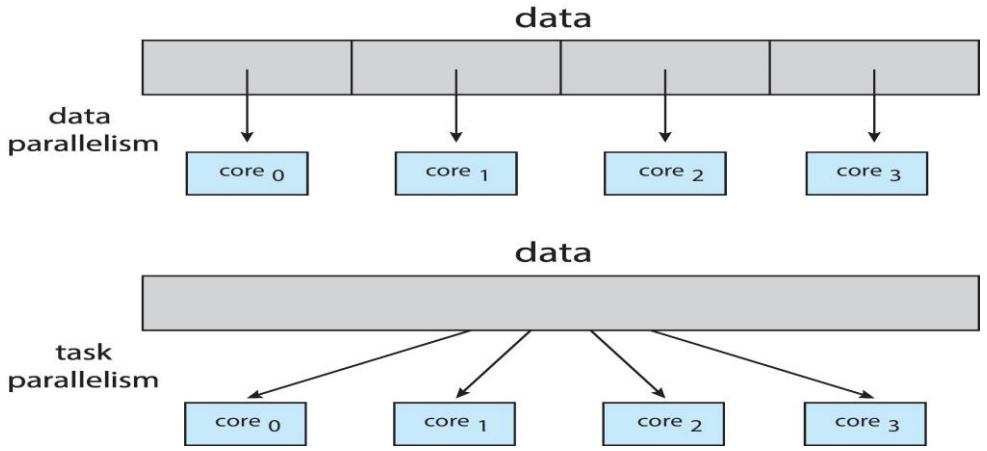


- **Parallelism on a multi-core system:**



Multicore Programming

- Types of parallelism
  - **Data parallelism** – distributes subsets of the same data across multiple cores, same operation on each
  - **Task parallelism** – distributing threads across cores, each thread performing unique operation

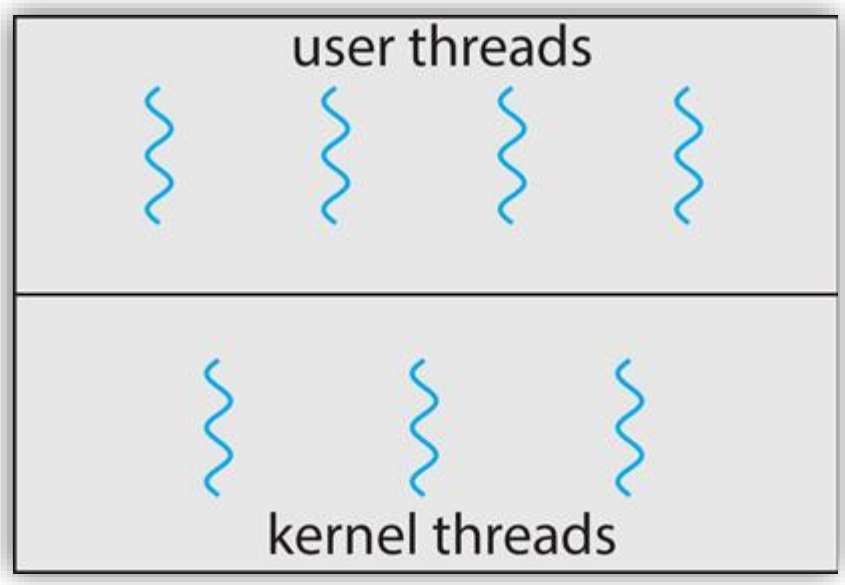


User Threads and Kernel Threads

- **User threads** - management done by user-level threads library
  - Application level
- Three primary thread libraries:
  - POSIX **Pthreads**
  - Windows threads
  - Java threads

- **Kernel threads** - Supported by the Kernel
- Examples – virtually all general-purpose operating systems, including:
  - Windows
  - Linux
  - Mac OS X
  - iOS
  - Android

User Threads and Kernel Threads

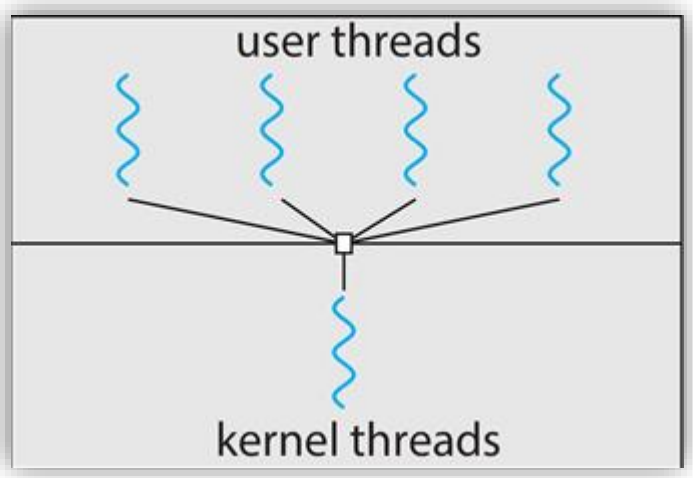


User Space

Kernel Space

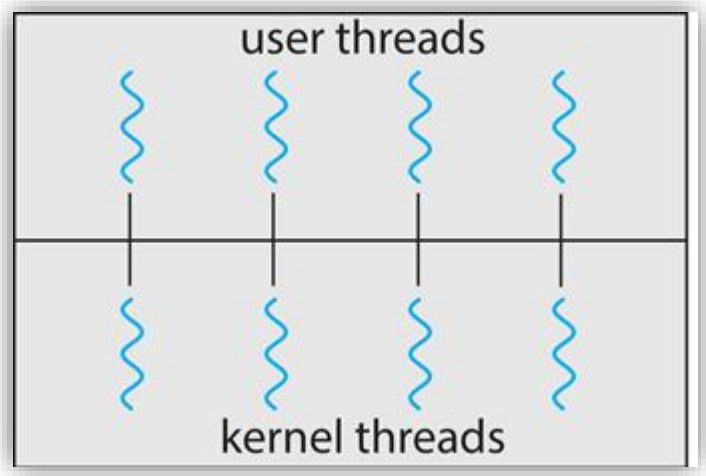
Many-to-One

- Many user-level threads mapped to single kernel thread
- One thread blocking causes all to block
- Multiple threads may not run in parallel on multicore system because only one may be in kernel at a time
- Few systems currently use this model
- Examples:
  - **Solaris Green Threads**
  - **GNU Portable Threads**



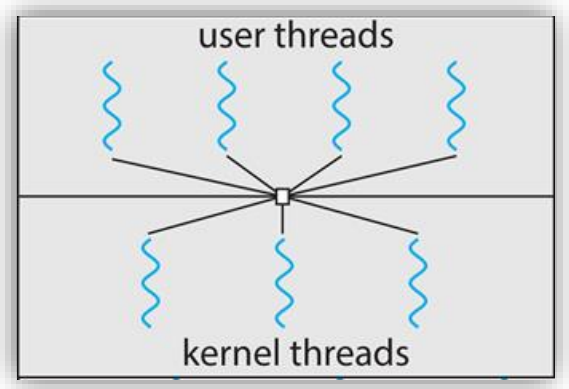
One-to-One

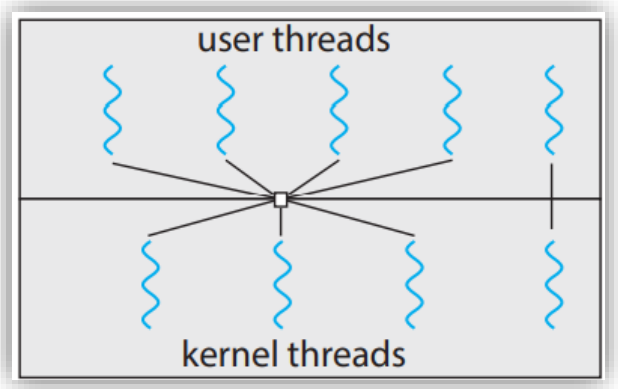
- Each user-level thread maps to kernel thread
- Creating a user-level thread creates a kernel thread
- More concurrency than many-to-one
- Number of threads per process sometimes restricted due to overhead
- Examples
  - Windows
  - Linux



Many-to-Many Model

- Allows many user level threads to be mapped to many kernel threads
- Allows the operating system to create appropriate number of kernel threads
- Windows with the *ThreadFiber* package
- Otherwise not very common





Q

	PCB	TCB1	TCB2
1	blocked	ready	ready
2	running	running	ready
3	ready	ready	blocked

In a multi threaded process, which of the following combinations of process state and thread states could occur?

Thread Libraries

- Provide a library entirely in user space.

- Implement a kernel-level library supported directly by the operating system.

Pthreads Library

- A POSIX standard (IEEE 1003.1c) API for thread creation and synchronization

```
#include<pthread.h>
//Thread variables declared
pthread_t   p1, p2;
//Creating Thread
pthread_create(&p1, NULL,mythread,"A");
//Join a thread
pthread_join(p1,NULL);
```

Thread Executing function

```
void *mythread(void *arg){
        printf("%s \n",(char *)arg);
        return NULL;
 }
```

Compile using : gcc –o objname filename.c -pthread

Lab 3 –Team of 2

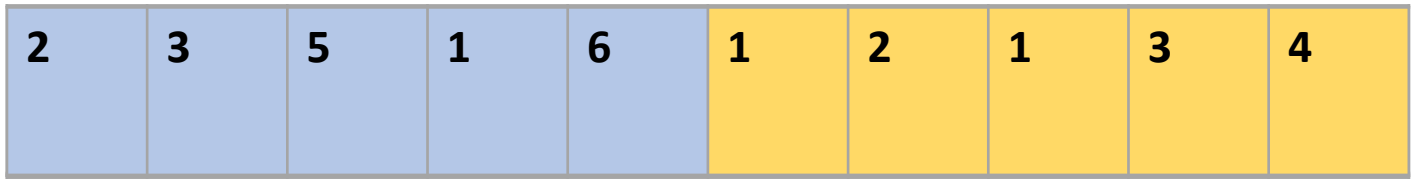
Q1) Declare a number (n) as a global variable in main function.

- Create 2 threads in the program.
  - Create the first thread to generate and print a list of numbers from n to 1 in descending order.
  - Another thread to find the factorial of the number n.
  - Add sleep(1) inside the loop to simulate thread switching when many threads works at the same time
  - What did you observe while running the program?

- Teamwork , submit as a team with names of all team members commented in the c file submitted for Lab3.

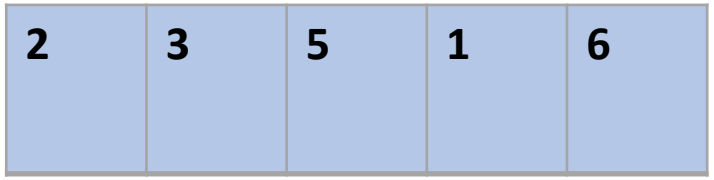
HW2

- Array Sum with four array partitions, here we see example with 2 partitions.

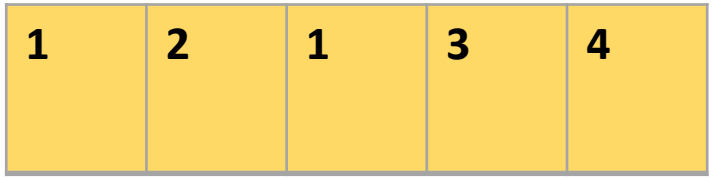


One Thread fun to find sum of 5 elements and accumulate the sum in a global variable sum

Thread1



Thread2



Do not look for correct answers you may not get the actual sum as we are not synchronizing the thread for the same data.

Announcements (02/10/22)

- Lab 3 Due Friday 02/11/22
- Weekly Quiz will be open by Friday 02/11/22
- Homework 02 available from Friday (02/11/22) 4pm