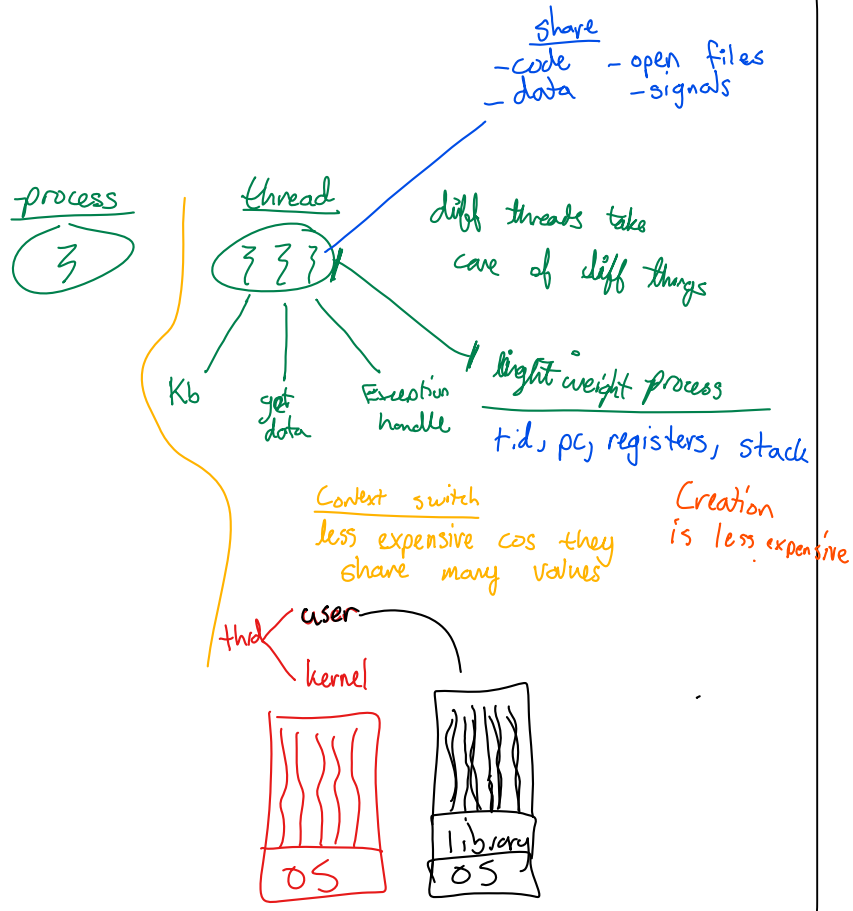
Module 5: Threads

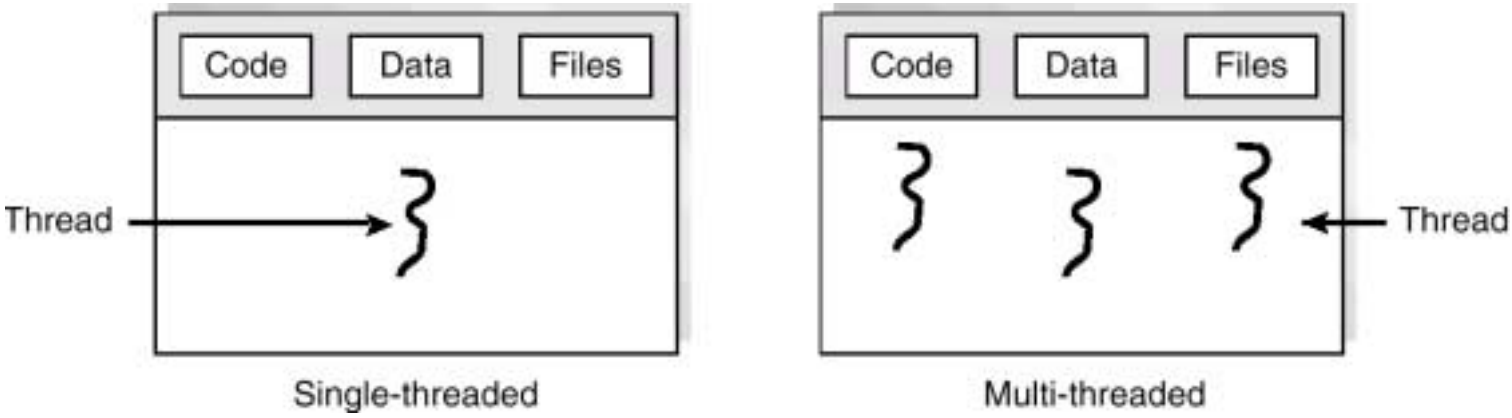
- Benefits
- User and Kernel Threads
- Multithreading Models
- Solaris 2 Threads
- Java Threads



Benefits

- Responsiveness
- Resource Sharing
- Economy
- Utilization of MP Architectures

Single and Multithreaded Processes



User Threads

- Thread Management Done by User-Level Threads Library
- Examples
  - POSIX *Pthreads*
  - Mach *C-threads*
  - Solaris *threads*

Kernel Threads

- Supported by the Kernel
- Examples
  - Windows 95/98/NT
  - Solaris
  - Digital UNIX

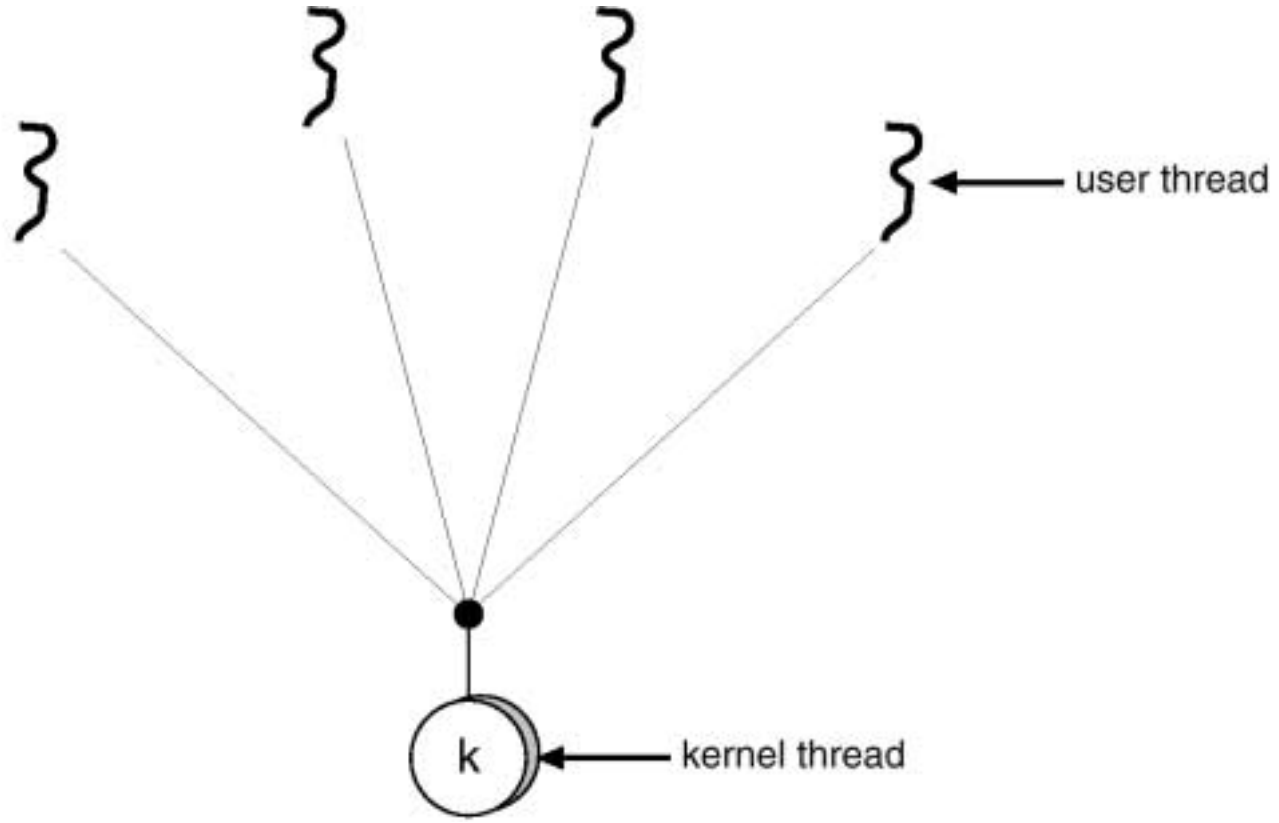
Multithreading Models

- Many-to-One
- One-to-One
- Many-to-Many

Many-to-One

- Many User-Level Threads Mapped to Single Kernel Thread.
- Used on Systems That Do Not Support Kernel Threads.

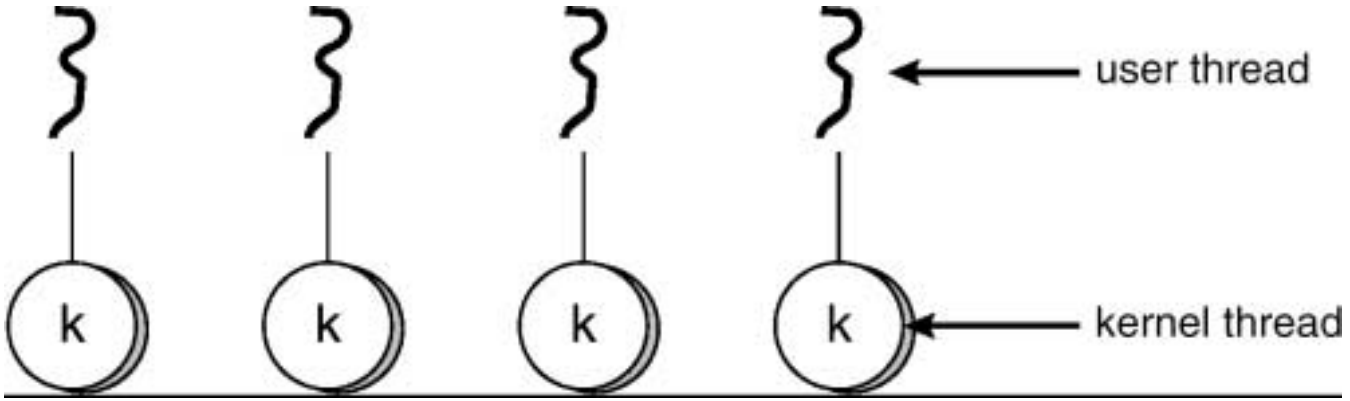
Many-to-one Model



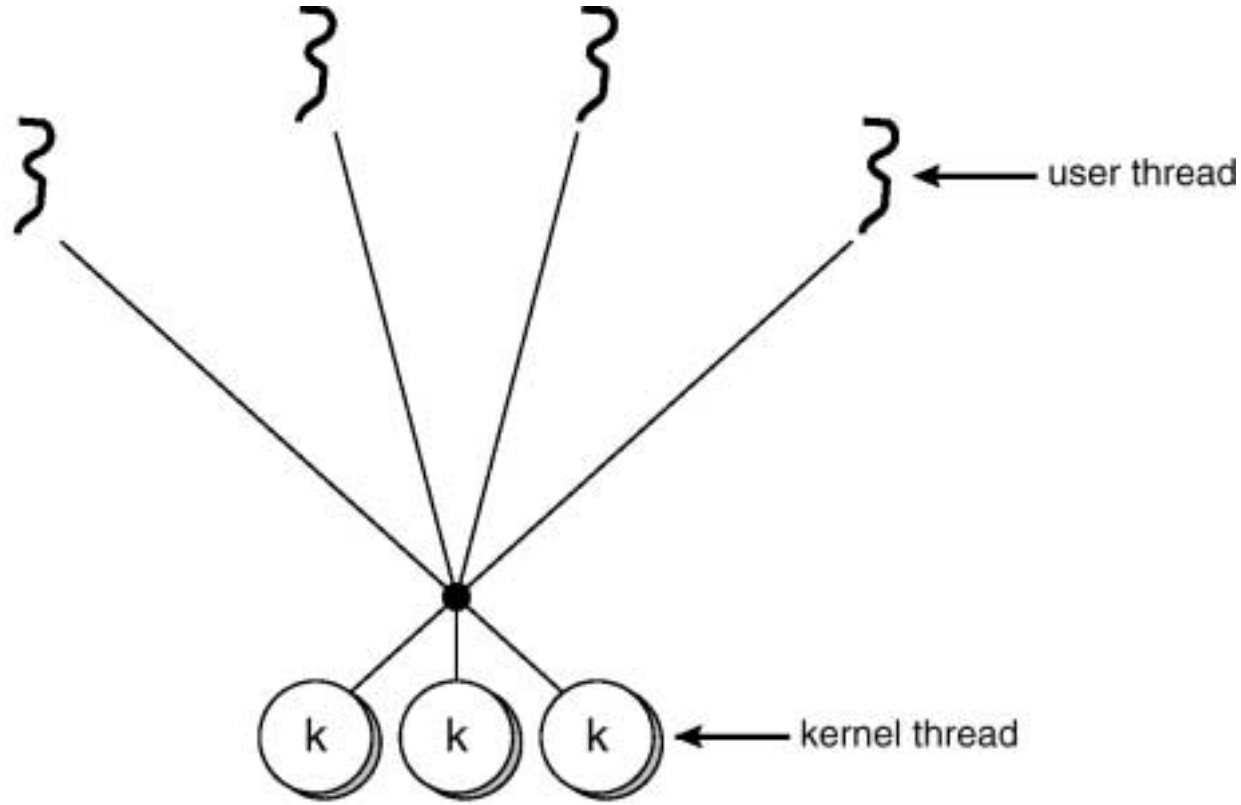
One-to-One

- Each User-Level Thread Maps to Kernel Thread.
- Examples
  - Windows 95/98/NT
  - OS/2

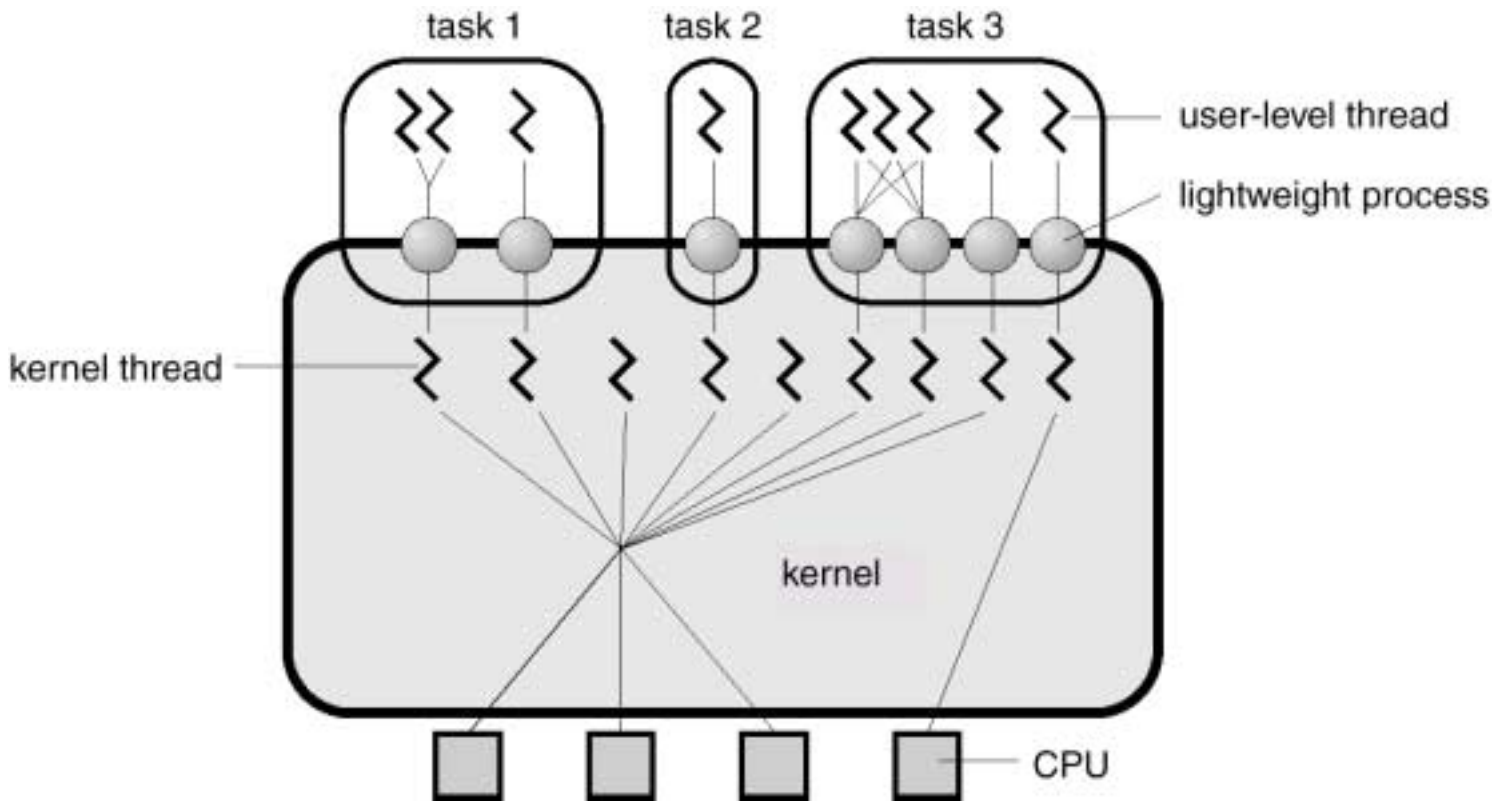
One-to-one Model



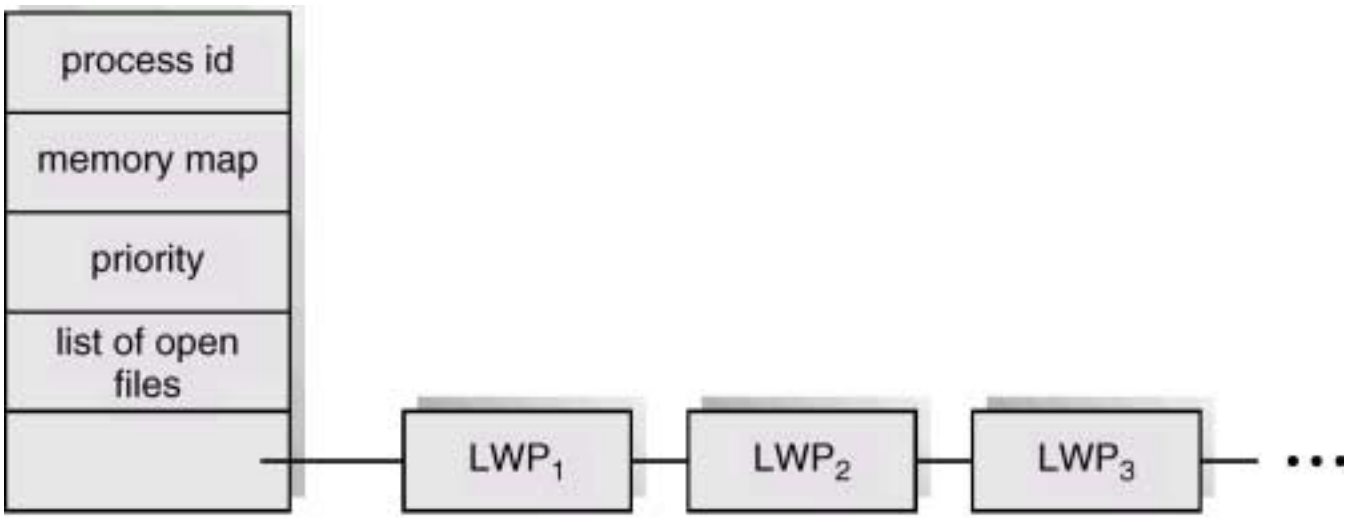
Many-to-many Model



Solaris 2 Threads



Solaris Process



Solaris process

Java Threads

- **Java Threads May be Created by:**
  - **Extending Thread class**
  - **Implementing the Runnable interface**

Extending the Thread Class

```
class  Worker1 extends Thread
{
   public void run() {
            System.out.println("I am a Worker Thread");
   }
}
```

Creating the Thread

```
public class First
{
   public static void main(String args[]) {
         Worker runner = new Worker1();

         runner.start();

         System.out.println("I am the main thread");
   }
}
```

The Runnable Interface

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

Implementing the Runnable Interface

```
class  Worker2 implements Runnable
{
   public void run() {
            System.out.println("I am a Worker Thread");
   }
}
```

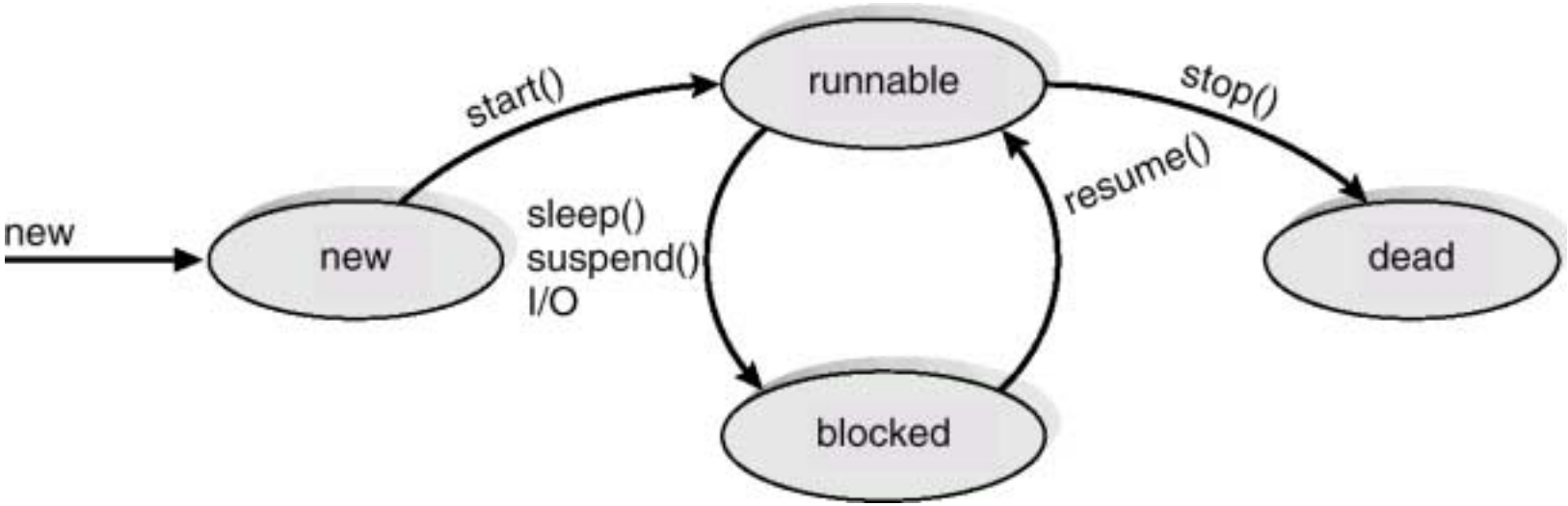
Creating the Thread

```
public class Second
{
   public static void main(String args[]) {
            Runnable runner = new Worker2();
            Thread thrd = new Thread(runner);
            thrd.start();

            System.out.println("I am the main thread");
   }
}
```

Java Thread Management

- **suspend()** – suspends execution of the currently running thread.
- **sleep()** – puts the currently running thread to sleep for a specified amount of time.
- **resume()** – resumes execution of a suspended thread.
- **stop()** – stops execution of a thread.

Java Thread States



Producer Consumer Problem

```
public class Server {
    public Server() {
            MessageQueue mailBox = new MessageQueue();

            Producer producerThread = new Producer(mailBox);
            Consumer consumerThread = new Consumer(mailBox);


            producerThread.start();
            consumerThread.start();
    }
    public static void main(String args[])   {
            Server server = new Server();
    }
}
```

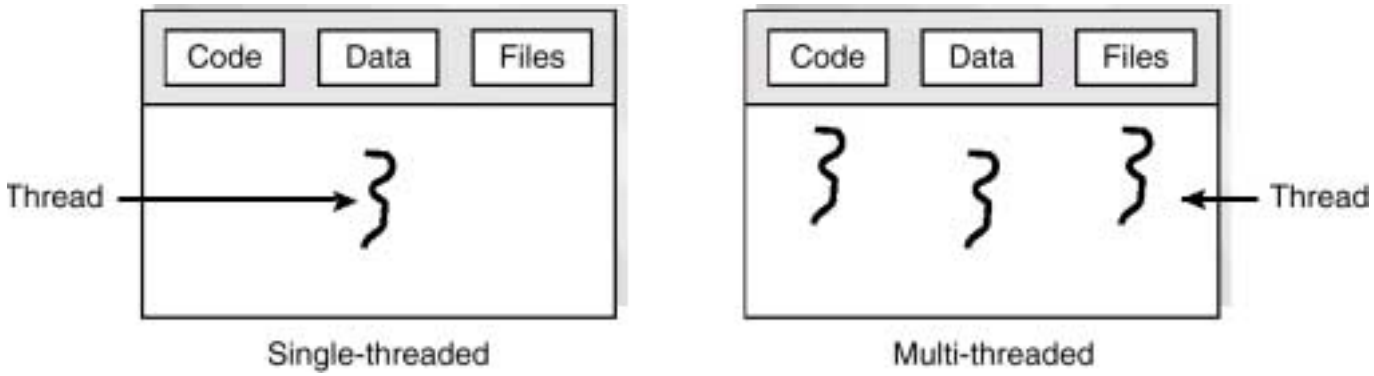
Producer Thread

```
class Producer extends Thread {
    public Producer(MessageQueue m) {
            mbox = m;
    }

   public void run() {
     while (true) {
             // produce an item & enter it into the buffer
            Date message = new Date();
            mbox.send(message);
      }
   }
   private  MessageQueue mbox;
}
```

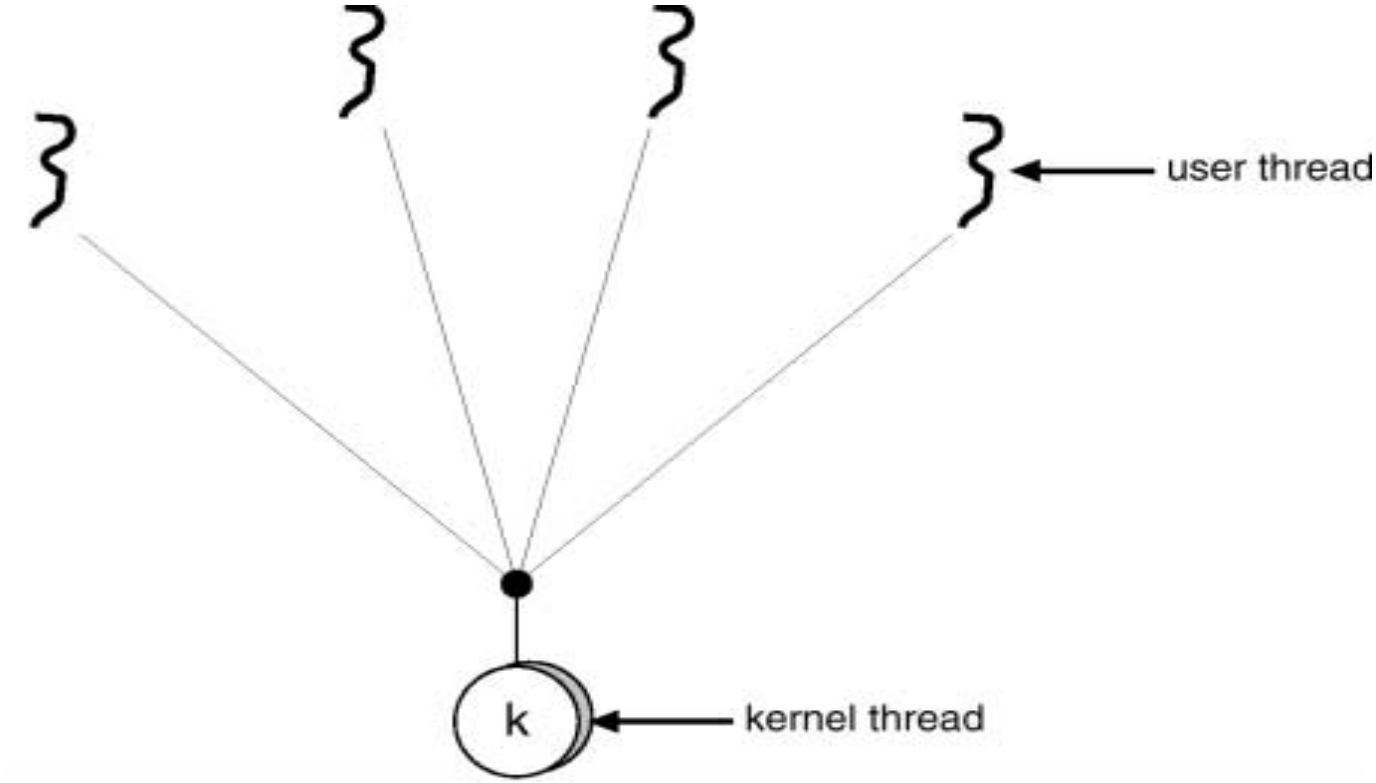
Consumer Thread

```
class Consumer extends Thread {
   public Consumer(MessageQueue m) {
            mbox = m;
   }

  public void run() {
    while (true) {
            Date message = (Date)mbox.receive();
            if (message != null)
                    // consume the message
   }
  }
  private  MessageQueue mbox;
}
```

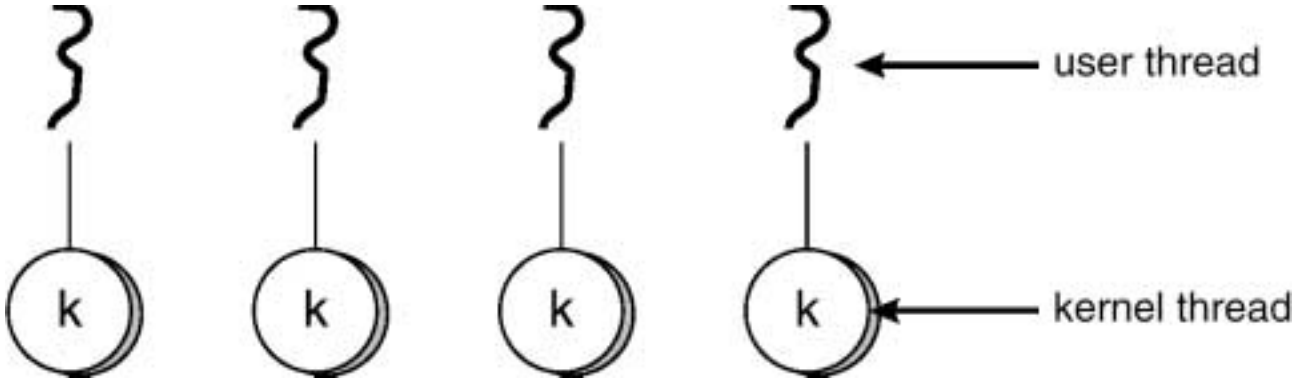
5.01


Code
Data
Files
Thread
Single-threaded
Code
Data
Files
Thread
Multi-threaded

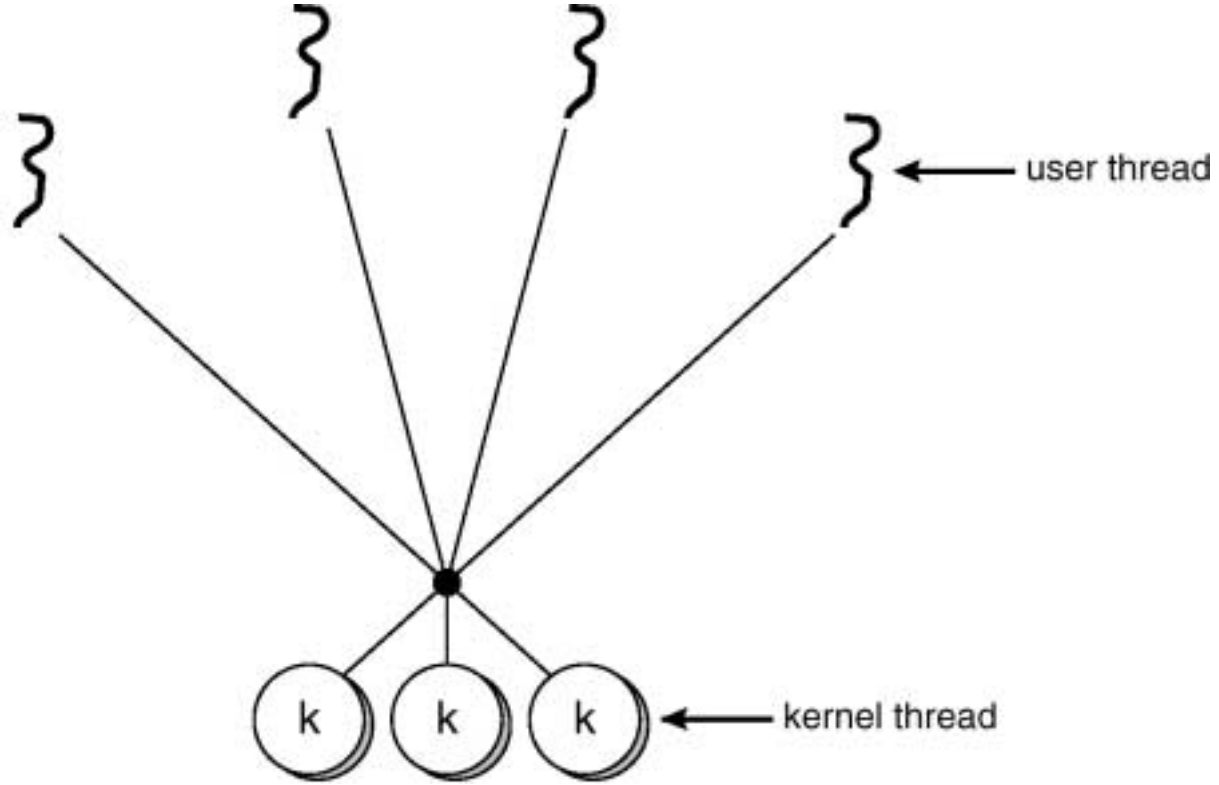
5.02



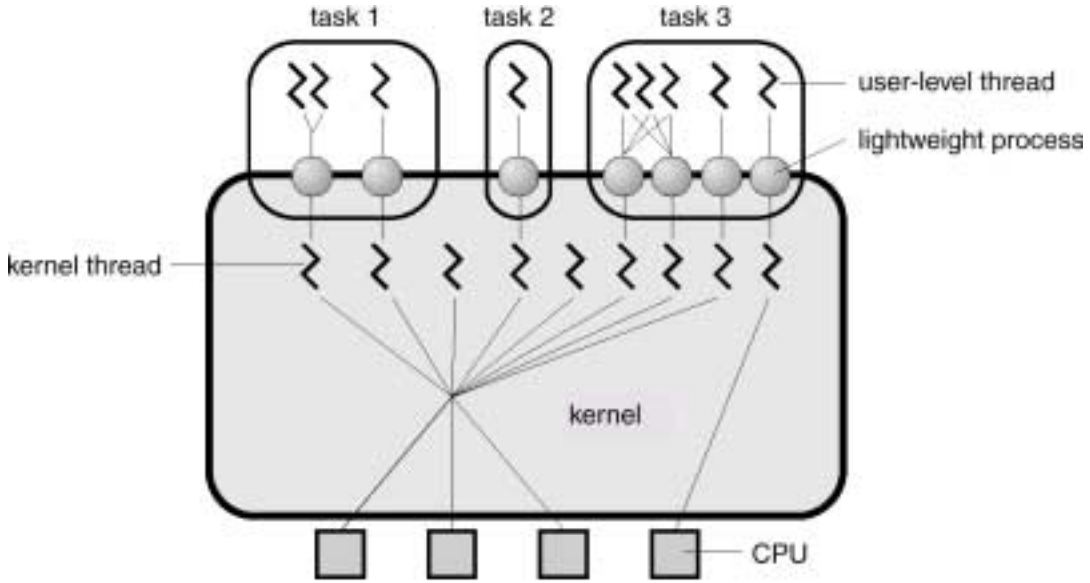
5.03



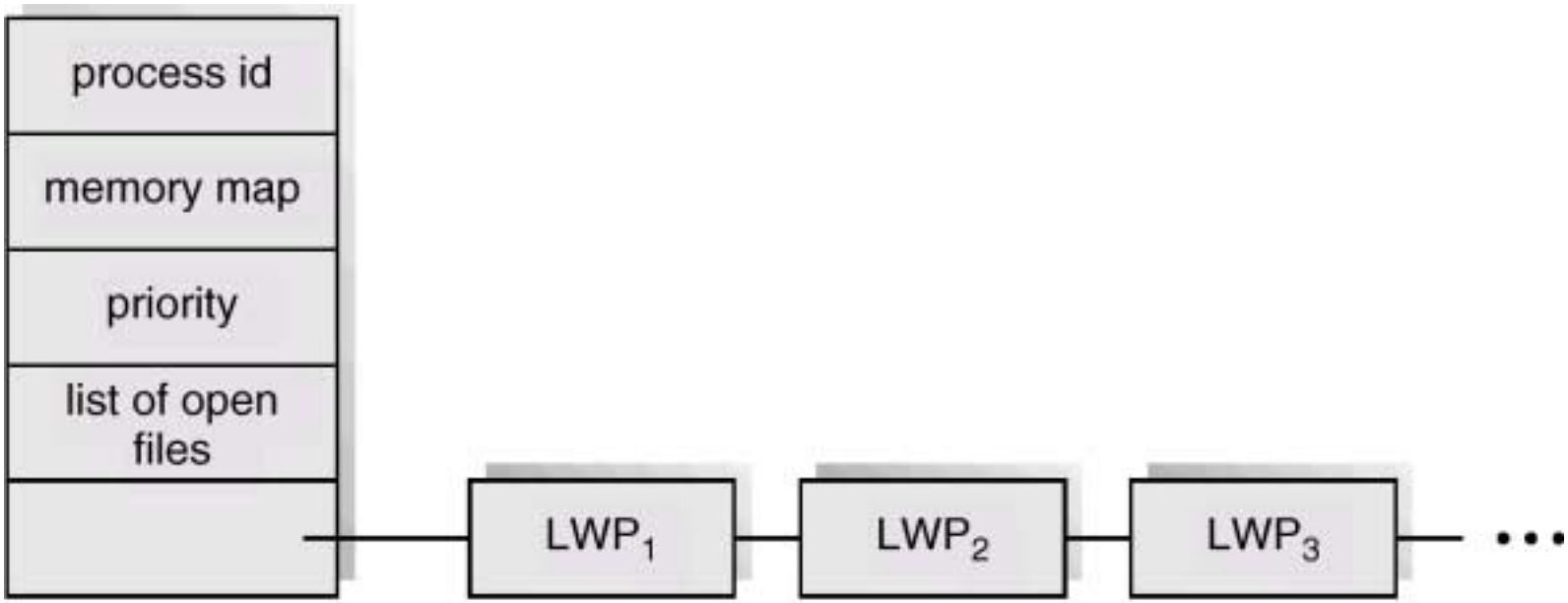
5.04



5.05

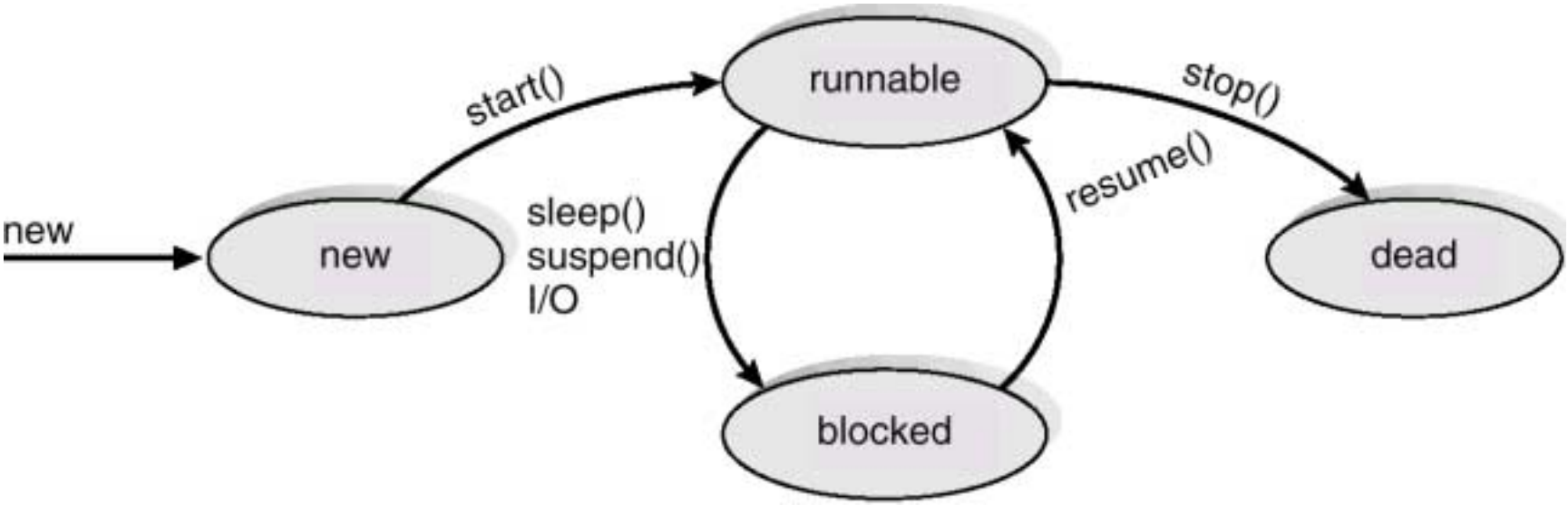

task 1
task 2
task 3
user-level thread
lightweight process
kernel thread
kernel
CPU

5.06



Solaris process

5.10


new
new
start()
runnable
stop()
sleep()
suspend()
I/O
resume()
dead
blocked