Sistem Operasi 4

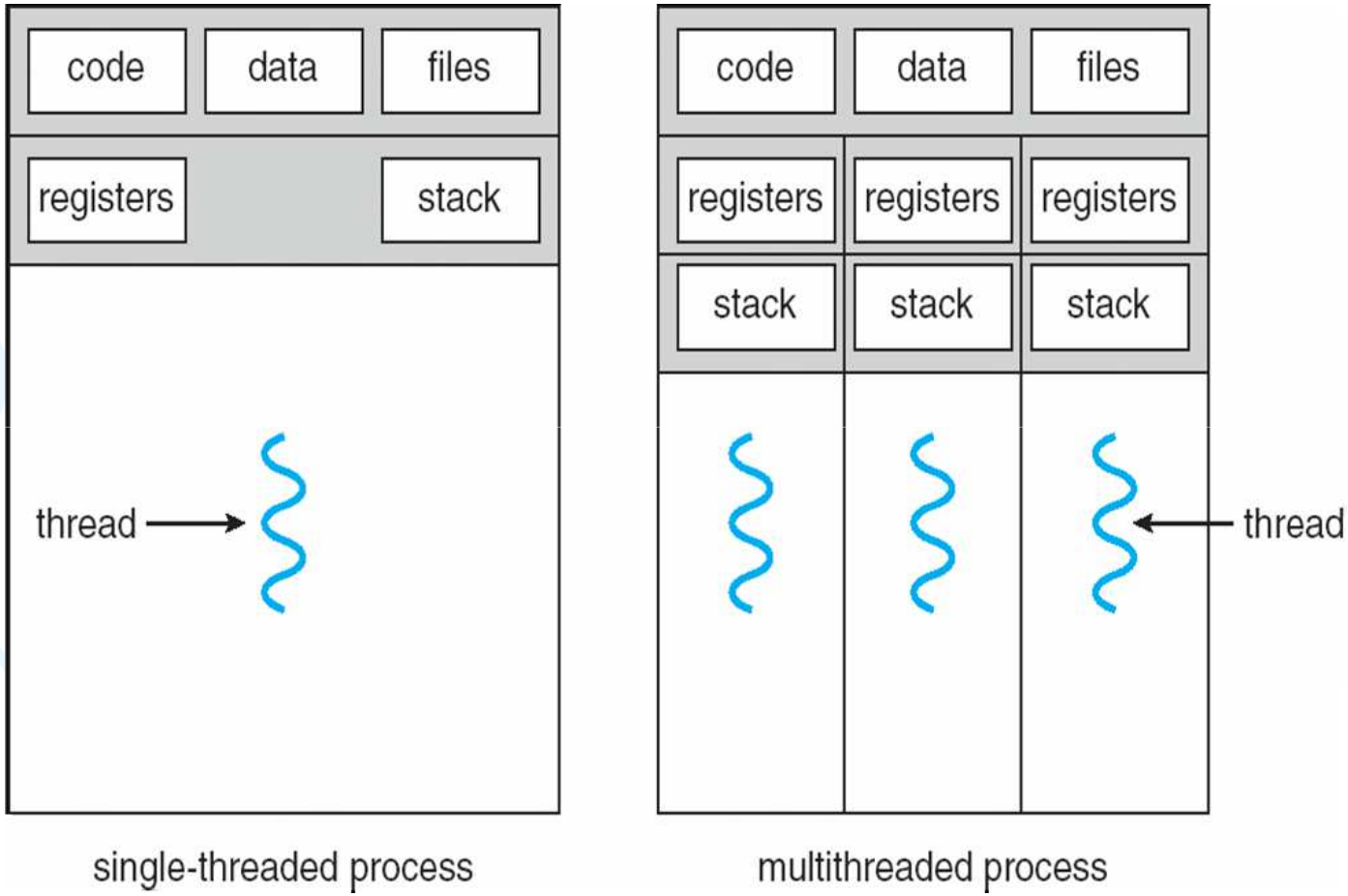
"Threads"

Antonius Rachmat C, S.Kom, M.Cs

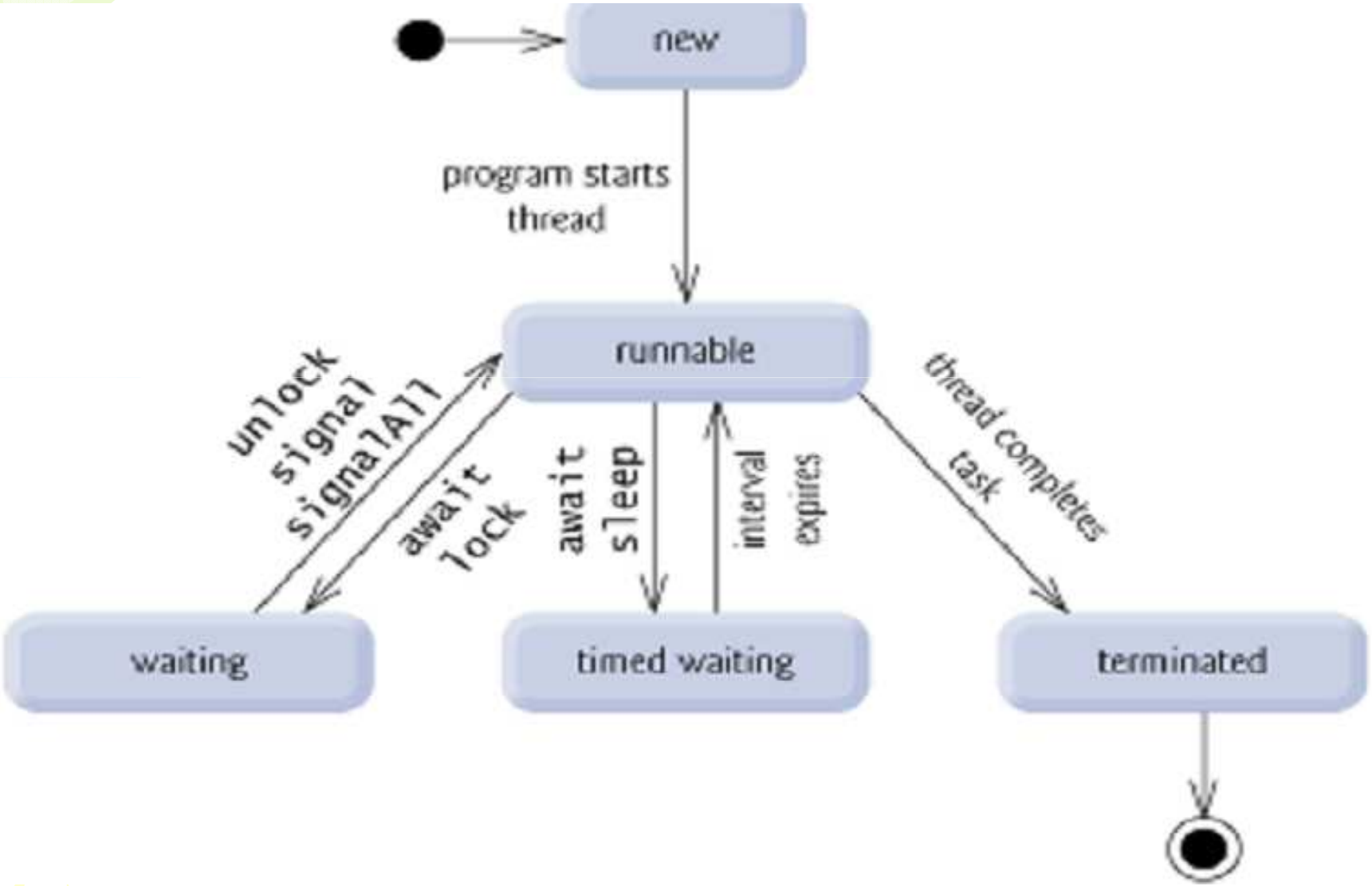
Thread

- Definisi: **unit dasar** dari penggunaan CPU.
- Thread terdiri dari:
  - thread ID, program counter, register, dan stack.
- Thread juga sering disebut **lightweight** process.
- Thread berbagi:
  - code section, data section, files dan sumber daya dengan thread lain yang termasuk dalam *proses yang sama.*
- Proses tradisional (**heavyweight** process) hanya mempunyai **thread tunggal**

Single and Multithreaded Processes



Thread Life Cycles



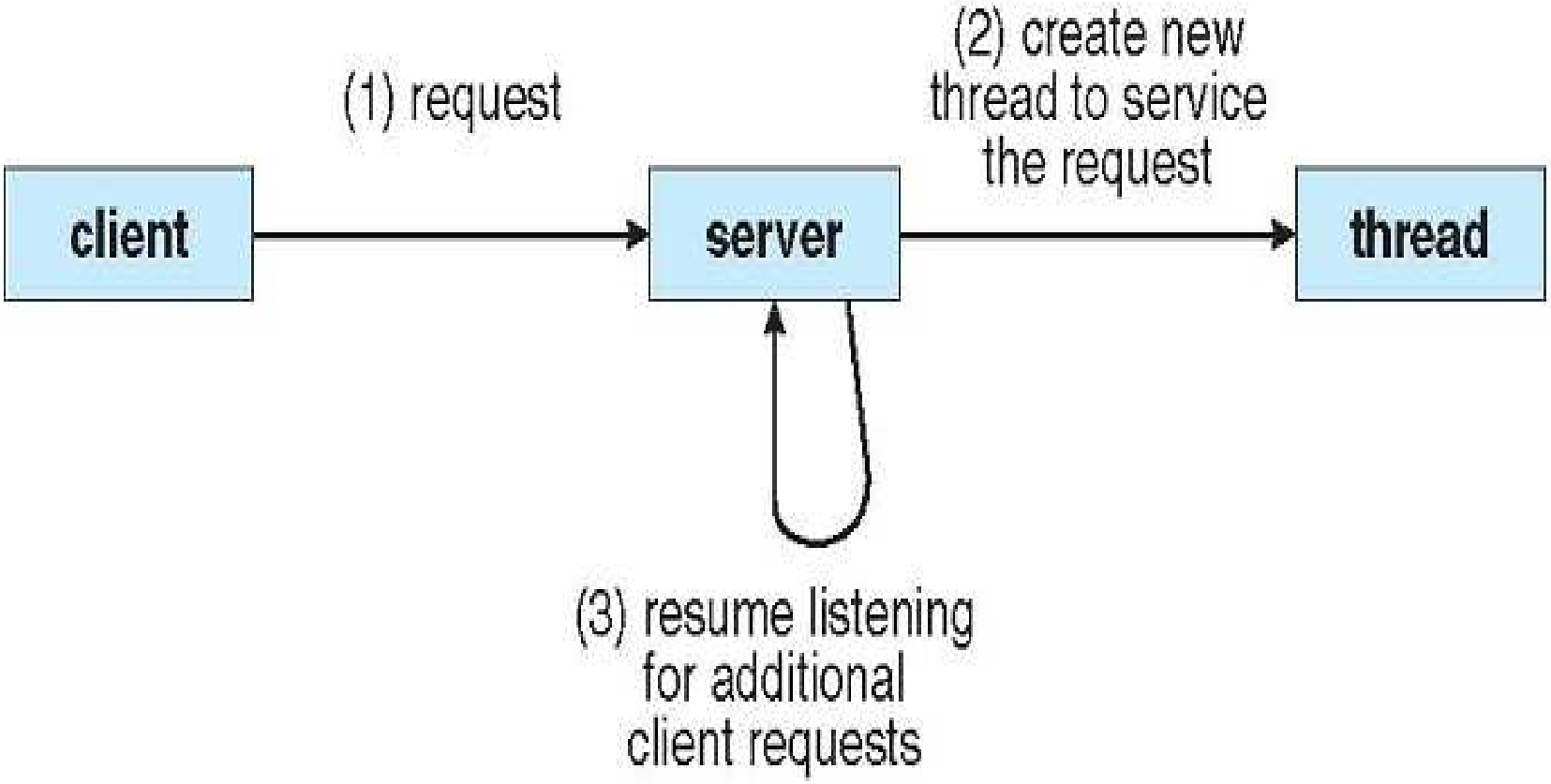
Keuntungan Thread

- **Responsiveness**
- **Resource sharing** : sebuah aplikasi dapat mempunyai beberapa **thread yang berbeda dalam sebuah alamat memori yang sama** karena thread berbagi memori dan sumber daya dari proses yang memilikinya.
- **Ekonomis**
- **Utilization of multiprocessor architectures**: setiap thread dapat berjalan secara paralel pada prosesor yang berbeda.

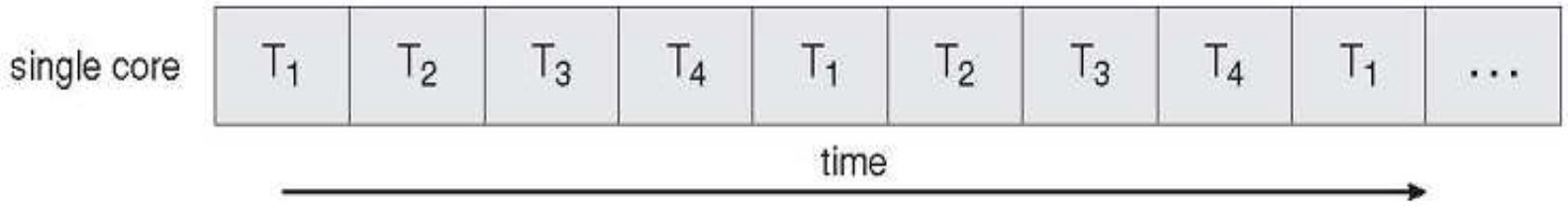
Multicore programming challenges

- Dividing activities
- Threads balance
- Data splitting
- Testing and debugging

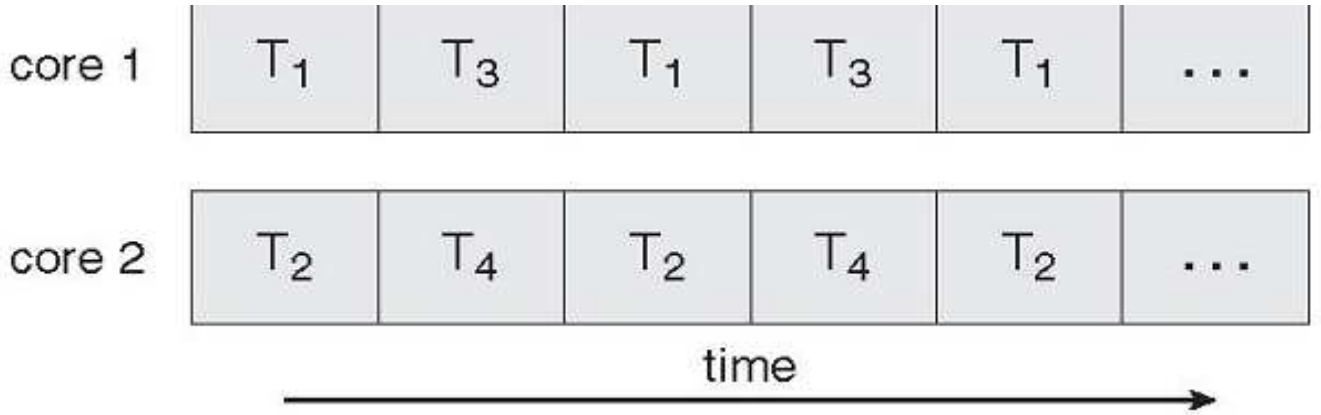
Multithreading Server



Single-core



Multi-core (paralel)



Fakta

- Pembuatan Proses **lebih berat** dan **lambat** daripada Thread
- Contoh pada Solaris:
  - Pembuatan proses **30x** lebih lama daripada Thread
  - Context Switch Proses **5x** lebih lama daripada Context Switch Thread

User Threads

- *User threads* didukung **diatas kernel** dan diimplementasikan oleh **thread library** pada level **user**.
- **User-Library** digunakan mendukung pembuatan thread, penjadwalan, dan manajemen tanpa bantuan langsung dari kernel.
- Karena kernel **tidak menyadari** adanya user-level threads maka pembuatan semua thread dan penjadwalan dilakukan di dalam **user space** tanpa intervensi dari kernel.
- Three primary **thread libraries**:
  - POSIX Pthreads
  - Win32 threads
  - Java threads
  - .NET threads

Thread Libraries

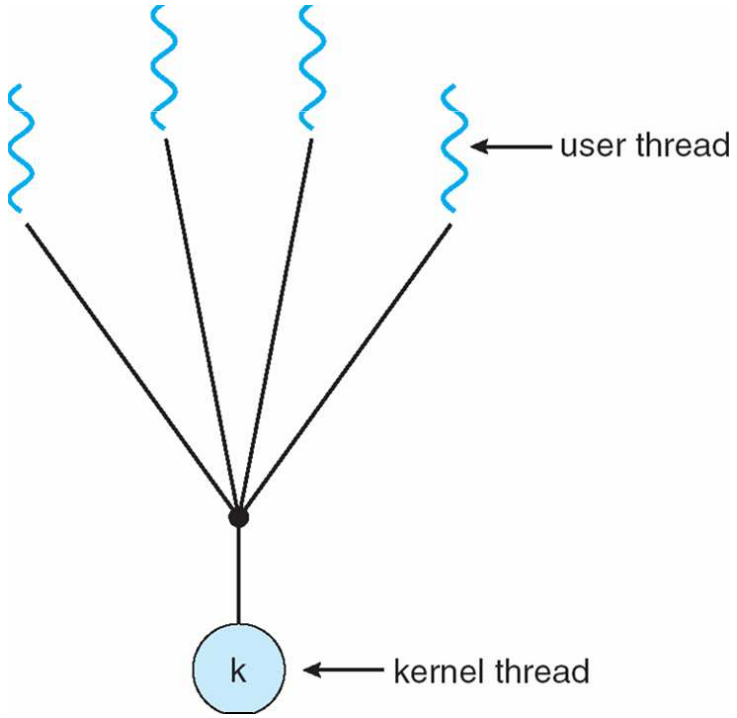
- Thread library provides programmers with API for creating and managing threads
- Two primary ways of implementing
  - Library entirely in user space
  - Kernel-level library supported by the OS

Kernel Threads

- *Kernel threads* didukung secara langsung dari sistem operasi.
  - Pembuatan thread, penjadwalan, dan manajemen dilakukan oleh **sistem operasi**,
  - Secara umum kernel threads lebih **lambat** untuk dibuat dibanding user thread.
- Keuntungan:
  - SO dapat menjadwalkan thread lain jika ada salah satu thread yang di block
  - SO dapat menjadwalkan thread-thread pada multi processor
- Examples
  - Windows XP/2000
  - Solaris
  - Linux
  - Mac OS X

Multithreading model

- **Many to One Model** memetakan beberapa user-level threads ke satu kernel threads.
- Examples:
  - Solaris Green Threads
  - GNU Portable Threads

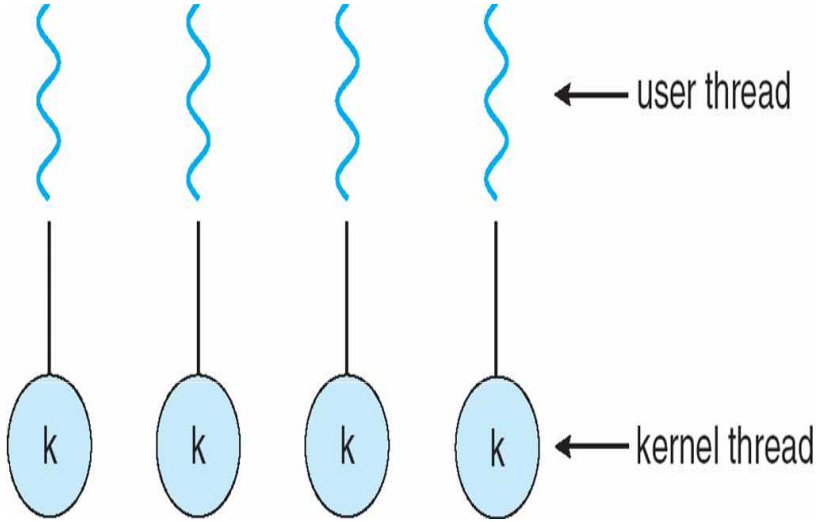


Many to one

- Efficient karena **irit** kernel thread
- All of user threads may **blocked** if a thread makes a blocking system call
- Multiple user thread **unable** to run in parallel on multiprocessors

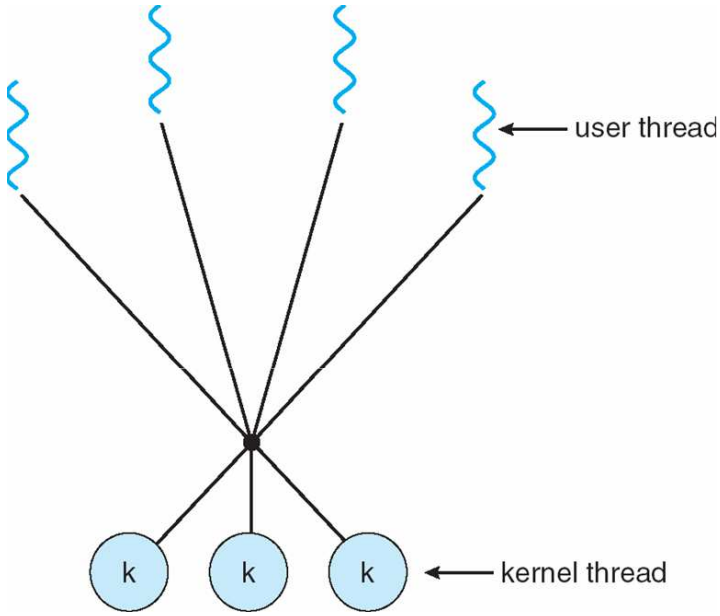
Multithreading model

- **One to One model**
  memetakan tiap user thread ke tiap kernel thread.
- Examples
  - Windows NT/XP/2000
  - Linux
  - Solaris 9 and later
- Keuntungan & Kerugian:
  - More **concurrency**
  - Multiple threads **can** run in parallel in multiprocessors
  - **Overhead** in creating kernel thread related to user thread



Multithreading model

- **Many to many** model memetakan banyak user-level thread ke <= kernel thread
- User dapat membuat banyak thread, masing2 kernel thread dpt jalan di lingkungan multiprocessor
- Allows the operating system to create a **sufficient** number of kernel threads
- Tidak terlalu overhead
- Examples
  - Solaris prior to version 9
  - Windows NT/2000 with the *ThreadFiber* package



Threading issue: cancellation

- **Thread cancellation** adalah sebuah tugas untuk memberhentikan thread sebelum thread tersebut selesai.
  - Contoh: JVM akan mematikan seluruh thread sblm end
- Thread yang akan diberhentikan disebut **target thread**.
- Dua skenario thread cancellation :
  - **Asynchronous cancellation** : satu thread memberhentikan target thread seketika itu juga.
  - **Deferred cancellation** : target thread secara periodik dapat mengecek apakah ia harus berhenti, skenario ini memberi kesempatan kepada target thread untuk memberhentikan dirinya sendiri.

Kesulitan Thread Cancellation

- **Asynchronous**: jika thread yg akan dimatikan, thread tersebut sedang mengupdate data.
- **Alternatifnya**: deferred cancellation

Signal handling

- Sebuah sinyal yang digunakan untuk memberitahu sebuah proses kalau peristiwa tertentu *sedang terjadi* (mis: perubahan state thread, error, interrupt).
- Pola sinyal bisa **asynchronous** / **synchronous**
  - Sebuah sinyal diaktifkan oleh munculnya suatu peristiwa.
  - Sinyal yang diaktifkan dikirim ke proses.
  - Sesudah dikirim, sinyal tersebut harus ditangani.

Signal handling

- **Sinyal Synchronous**:
  - Sinyal dimunculkan **oleh proses dan dikirim juga ke proses** yang melakukan operasi tersebut
  - Ketika ada pengaksesan *memory ilegal* atau *divison by zero*
- **Sinyal Asynchronous**:
  - Sinyal dimunculkan oleh peristiwa di **luar proses** tersebut
  - Ketika ada *proses menutup window* (ALT + F4)

Signal handling

- Penerima sinyal dapat:
  - Diatur oleh sistem operasi
  - Didefinisikan oleh user
- Pada single thread penanganan sinyal mudah, tapi multithreading sulit, krn 1 proses memiliki lbh dr 1 thread
- Kemungkinan sinyal dikirim ke:
  - Thread yg dituju
  - Setiap thread pd proses tersebut
  - Ke thread tertentu pd proses tersebut

Thread Pools

- Create a number of threads in a **pool** where they await work

- Advantages:
  - Usually slightly **faster** to service a request with an existing thread than create a new thread

Windows XP Threads

- Implements the **one-to-one mapping**, kernel-level
- Each thread contains
  - A thread ID
  - Registers set
  - Stack
  - Separate user and kernel stacks
  - Private data storage area
- The register set, stacks, and private storage area are known as the **context of the threads**
- The primary data structures of a thread include:
  - ETHREAD (executive thread block) – kernel space
  - KTHREAD (kernel thread block) – kernel space
  - TEB (thread environment block) – user space

Thread priorities

- THREAD_PRIORITY_IDLE
- THREAD_PRIORITY_LOWEST
- THREAD_PRIORITY_BELOW_NORMAL
- THREAD_PRIORITY_NORMAL
- THREAD_PRIORITY_ABOVE_NORMAL
- THREAD_PRIORITY_HIGHEST
- THREAD_PRIORITY_TIME_CRITICAL

Linux Thread

- Threads pada linux mulai digunakan di kernel versi **2.2**
- Threads lebih dianggap sebagai **tasks**
- Context switch Linux lebih cepat dari Windows

Java Threads

- Java threads are managed by the **JVM**
- Java threads may be created by:
  - **Extending** Thread class (subclassing)
  - **Implementing** the Runnable interface

Java Threads

- Keunggulan Java Threads adalah:
  - JVM menggunakan prioritas **pre-emptive** berdasarkan algoritma penjadwalan
  - Semua threads di Java mempunyai **prioritas** (1-10)
  - Jika ada 2 thread yang prioritasnya sama, digunakan algoritma First In First Out (**FIFO**)

Java vs Windows Thread

<table>
<tr><th>Java Priority</th><th>Windows Priority (Java 5)</th><th>Windows Priority (Java 6)</th></tr>
<tr><td>1</td><td rowspan="2">THREAD_PRIORITY_LOWEST</td><td rowspan="2"></td></tr>
<tr><td>2</td></tr>
<tr><td>3</td><td rowspan="2">THREAD_PRIORITY_BELOW_NORMAL</td><td rowspan="2"></td></tr>
<tr><td>4</td></tr>
<tr><td>5</td><td>THREAD_PRIORITY_NORMAL</td><td rowspan="2">THREAD_PRIORITY_NORMAL</td></tr>
<tr><td>6</td><td rowspan="2">THREAD_PRIORITY_ABOVE_NORMAL</td></tr>
<tr><td>7</td><td rowspan="2">THREAD_PRIORITY_ABOVE_NORMAL</td></tr>
<tr><td>8</td><td rowspan="2">THREAD_PRIORITY_HIGHEST</td></tr>
<tr><td>9</td><td rowspan="2">THREAD_PRIORITY_HIGHEST</td></tr>
<tr><td>10</td><td>THREAD_PRIORITY_TIME_CRITICAL</td></tr>
</table>

Contoh “extend Thread”

```
class SimpleThread extends Thread {
    public SimpleThread(String str) {
        super(str);
    }
    public void run() {
        for (int i = 0; i < 10; i++) {
            System.out.println(i + " " + getName());
            try {
                sleep((int)(Math.random() * 1000));
            } catch (InterruptedException e) {}
        }
        System.out.println("DONE! " + getName());
    }

    public static void main(String[] args){
        new SimpleThread("mythread").start();
    }
}
```

```
--------------------Configuration: <Default>----
0 mythread
1 mythread
2 mythread
3 mythread
4 mythread
5 mythread
6 mythread
7 mythread
8 mythread
9 mythread
DONE! mythread

Process completed.
```

Lainnya

```
public static void main(String[] args){
    new SimpleThread("thread 1").start();
    new SimpleThread("thread 2").start();
}
```

```
--------------------Configuration: <Default>--
0 thread 1
0 thread 2
1 thread 1
1 thread 2
2 thread 2
2 thread 1
3 thread 2
4 thread 2
5 thread 2
6 thread 2
3 thread 1
4 thread 1
7 thread 2
5 thread 1
8 thread 2
6 thread 1
7 thread 1
8 thread 1
9 thread 1
9 thread 2
DONE! thread 2
DONE! thread 1

Process completed.
```

Implements Runnable

```
class theThread implements Runnable{
   Thread t;

   public theThread(){
      t = new Thread(this,"Demo Thread");
      System.out.println("Thread anak mulai = "+t);
      t.start();
   }

   public void run(){
      try{
         for(int i=5;i>=1;i--){
            System.out.println("Thread anak : "+i);
            Thread.sleep(500);
         }
      } catch (InterruptedException ie){
         System.out.println("Thread anak diinterupsi");
      }
      System.out.println("Thread anak selesai.");
   }
}

public class DemoThread{
   public static void main(String[] args){
      new theThread();
      System.out.println("Thread utama mulai");
      try{
         for(int i=0;i<10;i++){
            System.out.println("Thread utama : "+(i+1));
            Thread.sleep(1000);
         }
      }catch (InterruptedException ie){
         System.out.println("Thread utama diinterupsi");
      }
      System.out.println("Thread utama selesai.");
   }
}
```

Hasil

```
E:\Documents\Dosen\progjar\thread>java DemoThread
Thread anak mulai = Thread[Demo Thread,5,main]
Thread utama mulai
Thread anak : 5
Thread utama : 1
Thread anak : 4
Thread anak : 3
Thread utama : 2
Thread anak : 2
Thread utama : 3
Thread anak : 1
Thread anak selesai.
Thread utama : 4
Thread utama : 5
Thread utama : 6
Thread utama : 7
Thread utama : 8
Thread utama : 9
Thread utama : 10
Thread utama selesai.
```

NEXT

- Process Schedulling / Penjadwalan