Threads

Tran, Van Hoai

Faculty of Computer Science & Engineering
HCMC University of Technology

E-mail: hoai@hcmut.edu.vn
(partly based on slides of Le Thanh Van)

Outline

Outline

Motivation

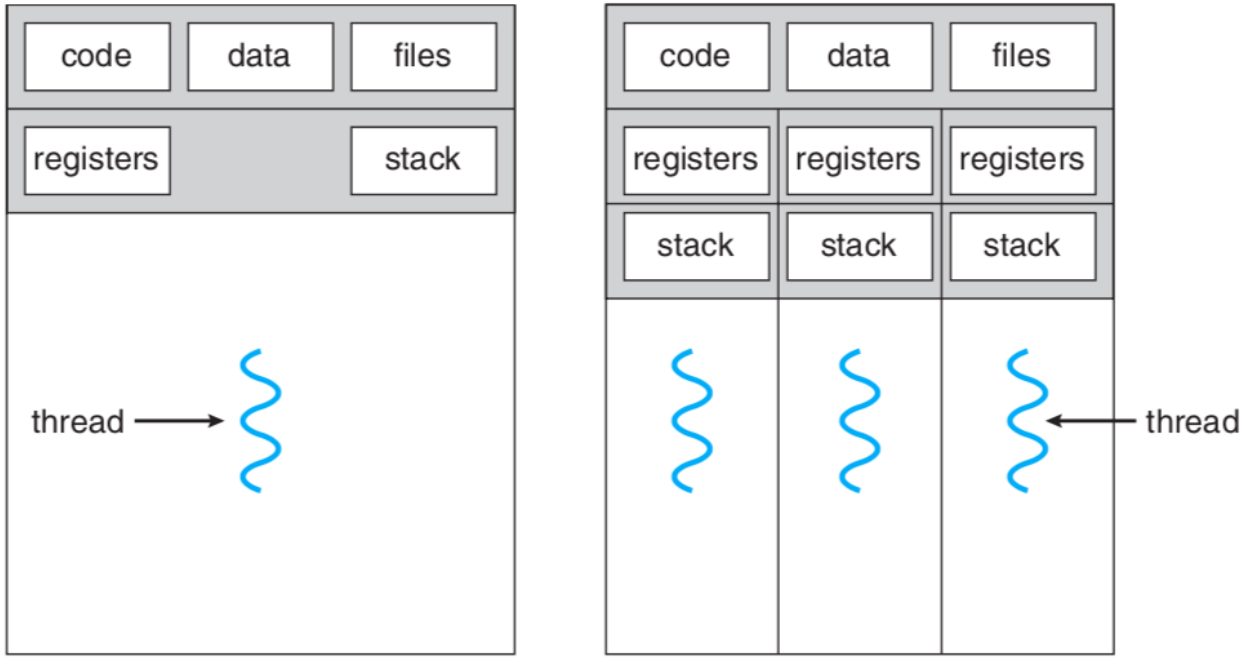
- An application normally has several controls

  A word-processor has a control on mouse input, a control for keyboard, a control for function completion, ...

- The model "an application = a process" does not catch up with multiprocessor environment

  A modern processor has multiple cores

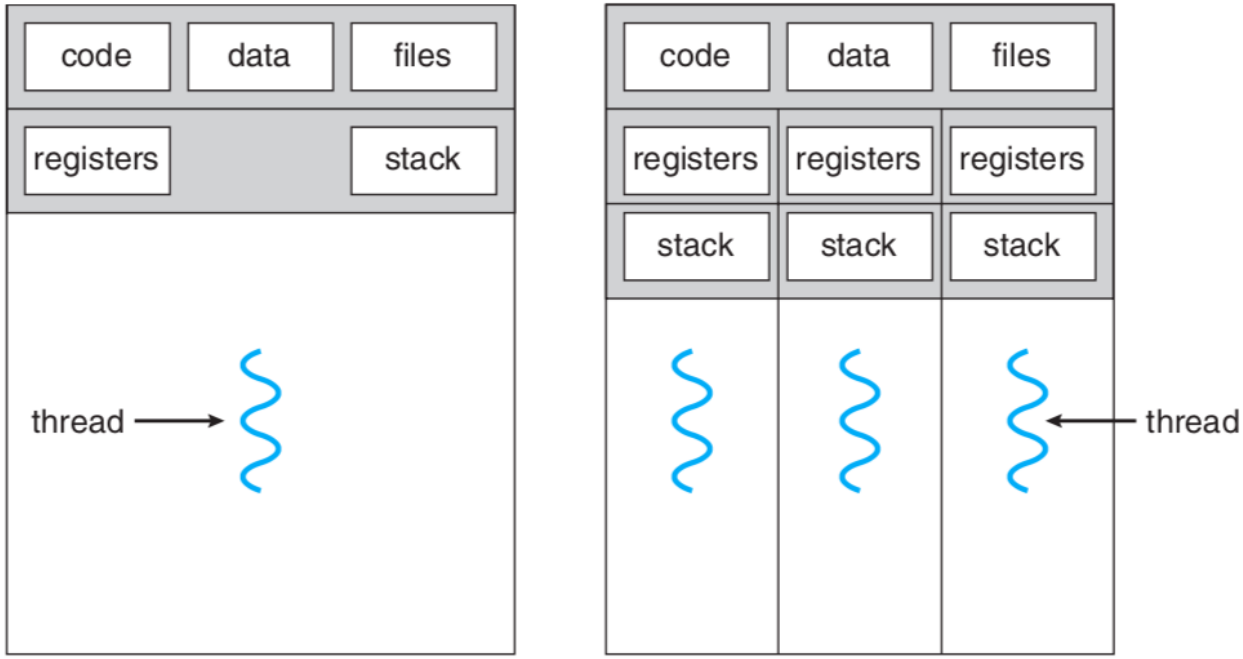
Single vs. Multithreaded processes



single-threaded process

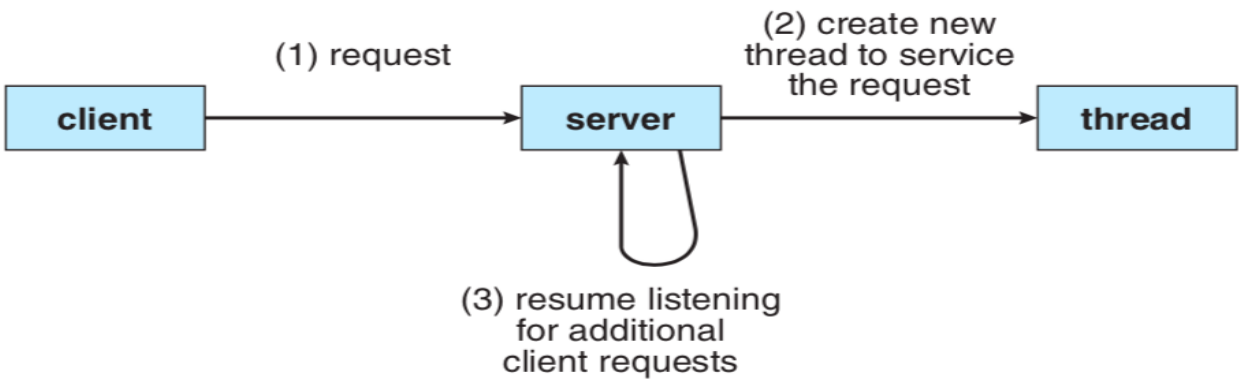
multithreaded process

Single vs. Multithreaded processes



single-threaded process multithreaded process

With the above model, one server (by one process) can service several concurrent requests.



Benefits of multithreading model

- **Responsiveness**: a program (process) continues running even if a part of it is blocked or is performing a lengthy operation.
- **Resource sharing**: By defaults, threads share memory and resources of its process $\Rightarrow$ same address-space.
- **Economy**: Resource allocation, context-switching are time-consuming. Threads do it more economically.
- **Scalability**: threads may be running in parallel on different processing cores.

Benefits of multithreading model

- **Responsiveness**: a program (process) continues running even if a part of it is blocked or is performing a lengthy operation.
- **Resource sharing**: By defaults, threads share memory and resources of its process $\Rightarrow$ same address-space.
- **Economy**: Resource allocation, context-switching are time-consuming. Threads do it more economically.
- **Scalability**: threads may be running in parallel on different processing cores.

concurrency vs. parallelism

- Concurrency = many tasks are allowed to make progress
- Parallelism = many tasks can be performed simultaneously

Programming challenges

- **Identifying tasks**: how to divide an application into tasks ?
- **Balance**: how tasks do the same amount of workload ?
- **Data splitting**: how data of tasks to be splitted ?
- **Data dependency**: data surely does not live alone, how they are synchronized ?
- **Testing and debugging**: how to follow many different execution paths ?

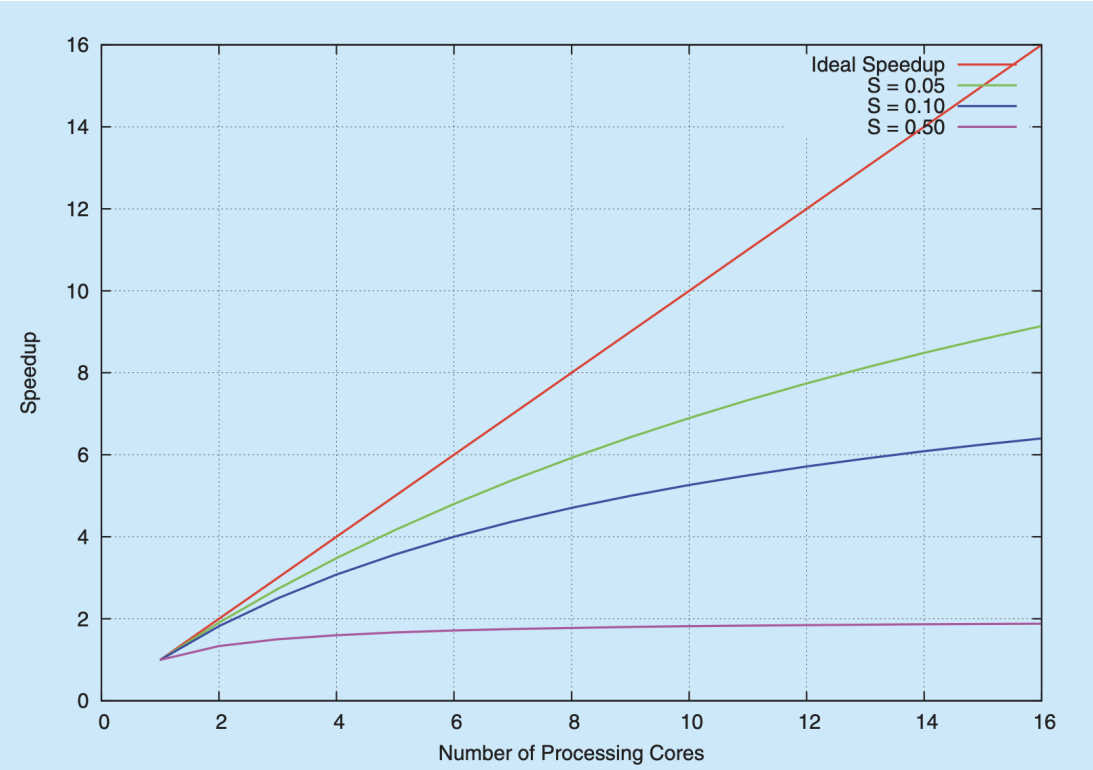
Programming challenges

- **Identifying tasks**: how to divide an application into tasks ?
- **Balance**: how tasks do the same amount of workload ?
- **Data splitting**: how data of tasks to be splitted ?
- **Data dependency**: data surely does not live alone, how they are synchronized ?
- **Testing and debugging**: how to follow many different execution paths ?

Textbook

"Many computer science educators believe that software development must be taught with increased emphasis on parallel programming."

Do not expect much on multiple threading



Amdahl's Law

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

in which, S: serial portion; N: number of cores (threads).

Outline

User vs. Kernel threads

User threads

- Thread management done by user-level thread library
- Examples: POSIX Pthreads, Mach C-threads, Solaris threads

Kernel threads

- Thread management done at kernel-level by OS
- Examples: Windows, Linux, Max OS X, Solaris

User vs. Kernel threads

User threads

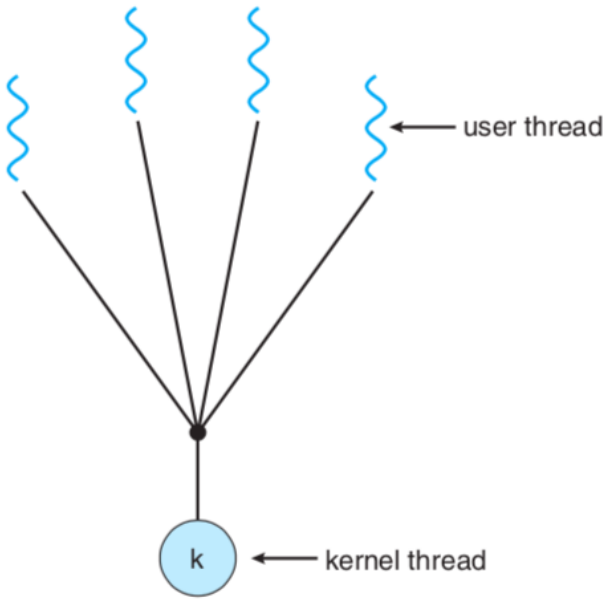
- Thread management done by user-level thread library
- Examples: POSIX Pthreads, Mach C-threads, Solaris threads

Kernel threads

- Thread management done at kernel-level by OS
- Examples: Windows, Linux, Max OS X, Solaris

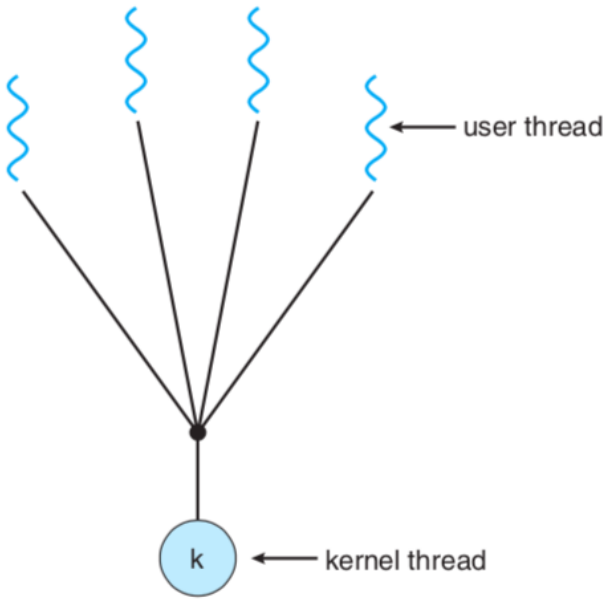
A relationship must exist between user threads and kernel threads

Many-to-one



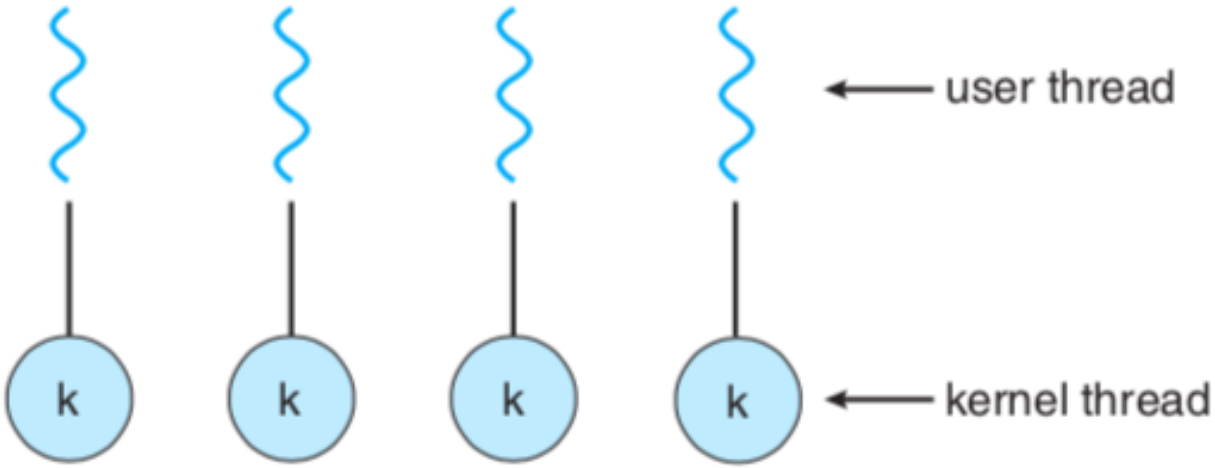
- Mapping many user-level threads to one kernel thread
- Issues:

Many-to-one



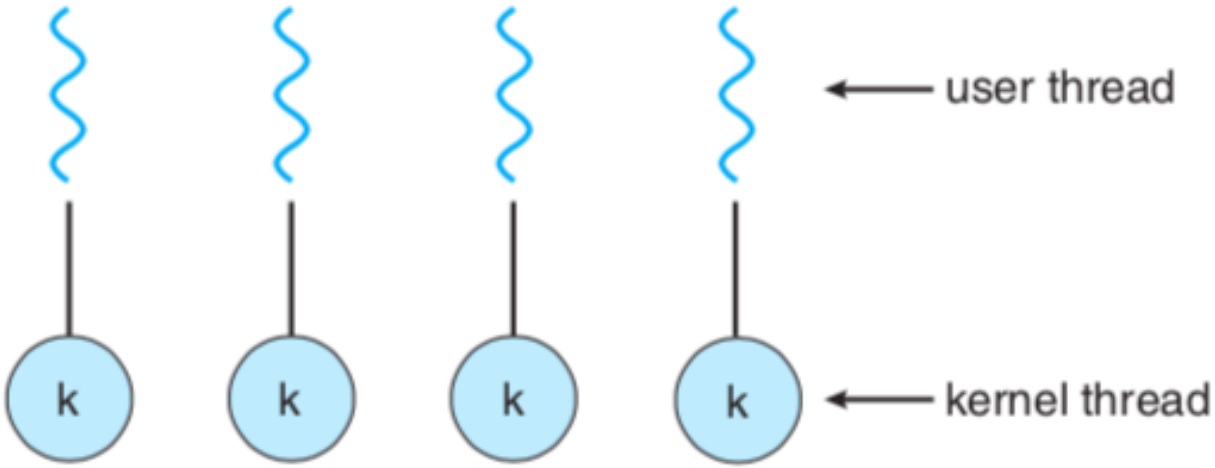
- Mapping many user-level threads to one kernel thread
- Issues:
  - if a thread is blocked, the entire process is blocked too.
  - Unable to run in parallel on multicore systems

One-to-one



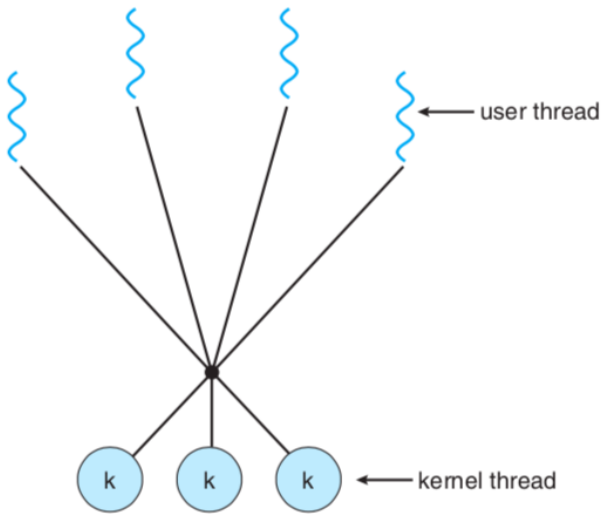
- Mapping each user thread to a kernel thread
- Issues:

One-to-one



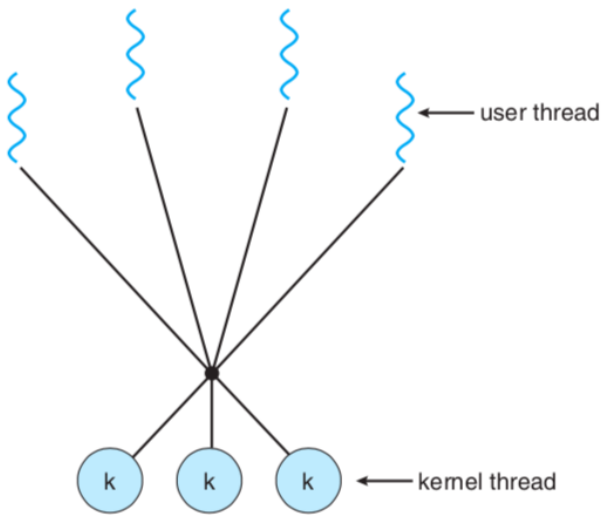
- Mapping each user thread to a kernel thread
- Issues:
  - Creating a user thread means creating a kernel thread ⇒ overhead.
  - Number of threads is restricted
  - Linux, Windows

Many-to-many



- Multiplexing many user-level threads to a smaller or equal number of kernel threads
- Issues:

Many-to-many



- Multiplexing many user-level threads to a smaller or equal number of kernel threads
- Issues:
  - Not so many OS implementations apply this model, (Solaris supports)