《 操 作 系 统 》

进程与线程

首都师范大学
信息工程学院
霍其润

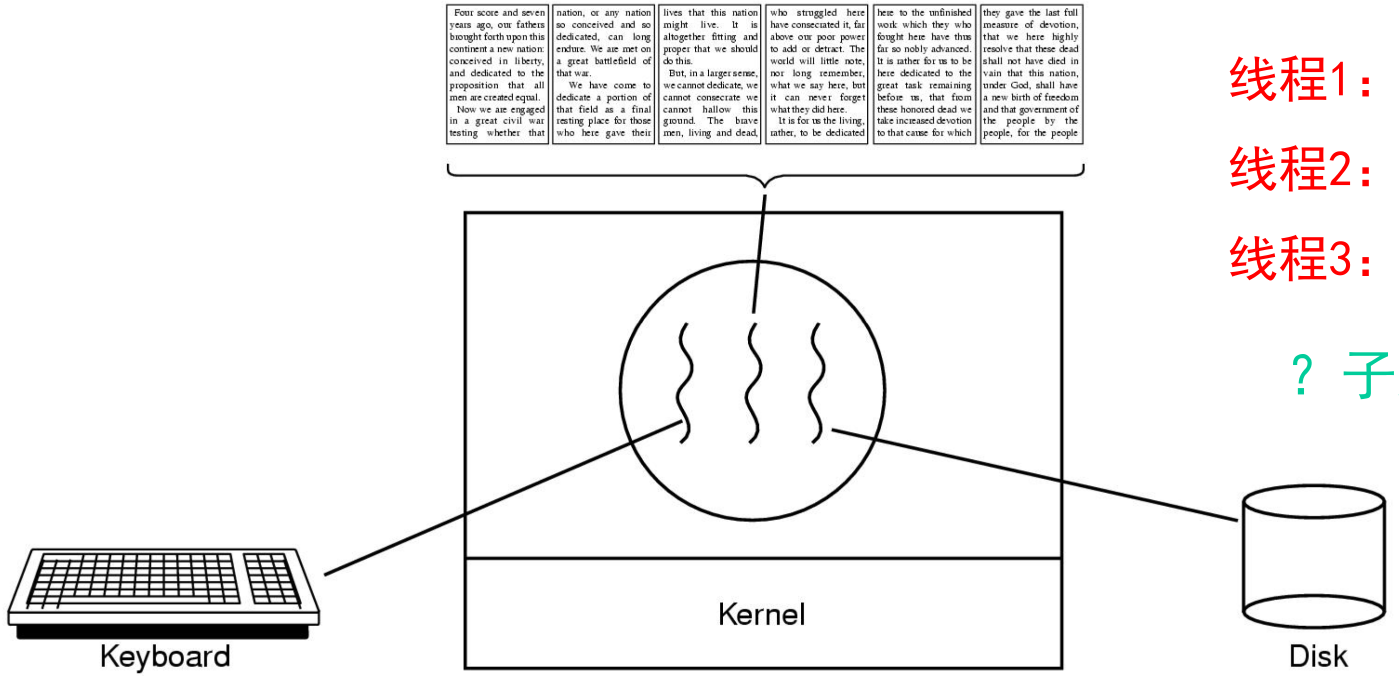
Processes and Threads

- Processes
- Threads
- Interprocess communication
- Classical IPC problems
- Scheduling

二 线程（Threads）

1、线程的引入

Example:



线程1：用户交互

线程2：后台排版

线程3：自动保存

？子进程

A word processor with three threads

- **线程：** “轻量级进程（Lightweight Process）”，进程的一个实体，是被独立调度和分派的基本单位，表示进程中的一个执行流，执行一系列指令。

现代os：一个进程可以创建多个线程

以比较小的系统开销提高进程内的并发程度。

进程作为其他资源分配单位

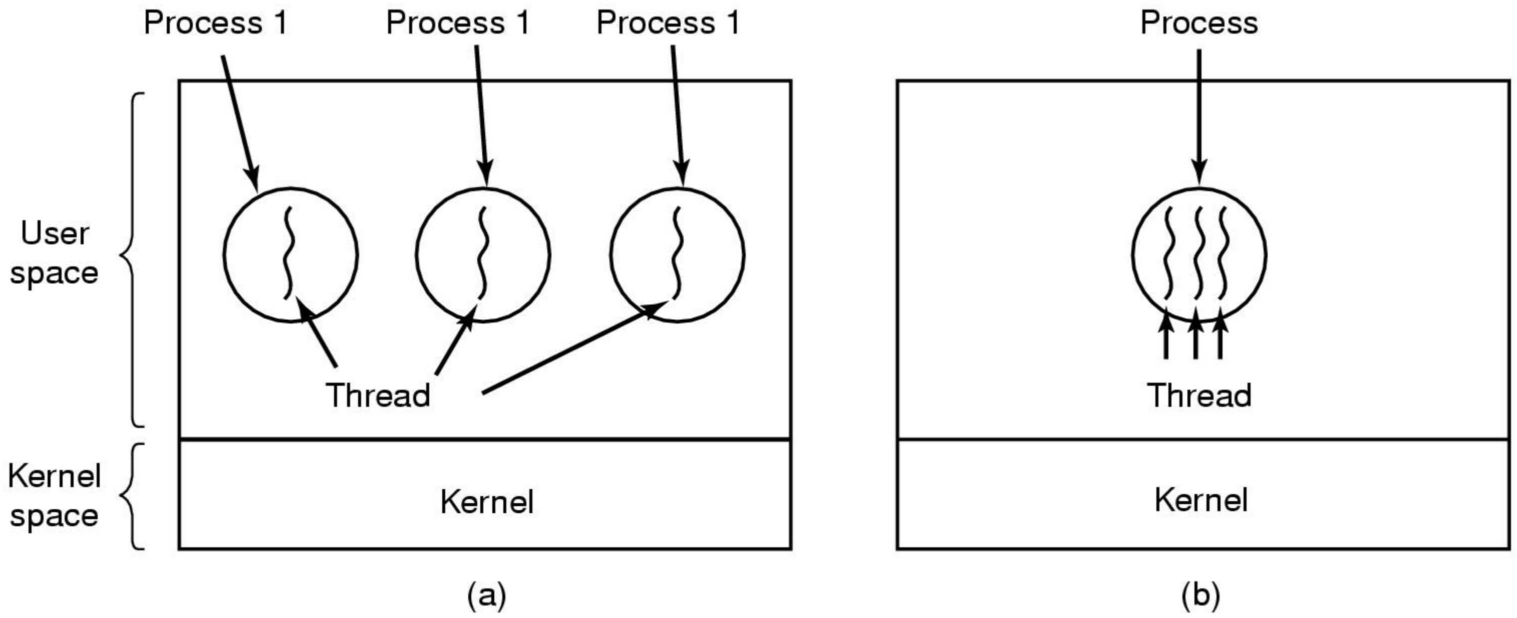
线程作为CPU调度单位

只拥有必不可少的资源，如：PC、寄存器上下文和栈

二 线程（Threads）

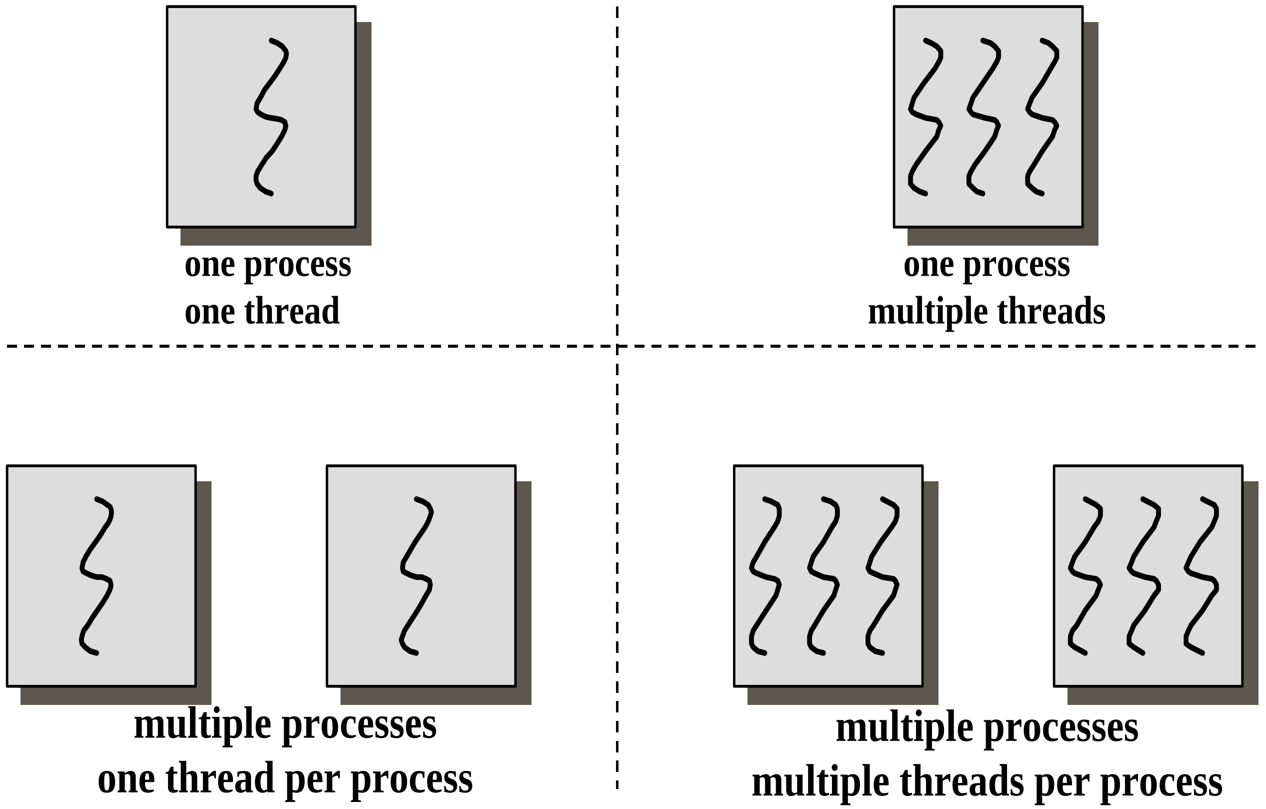
2、线程模型

The Thread Model (1)

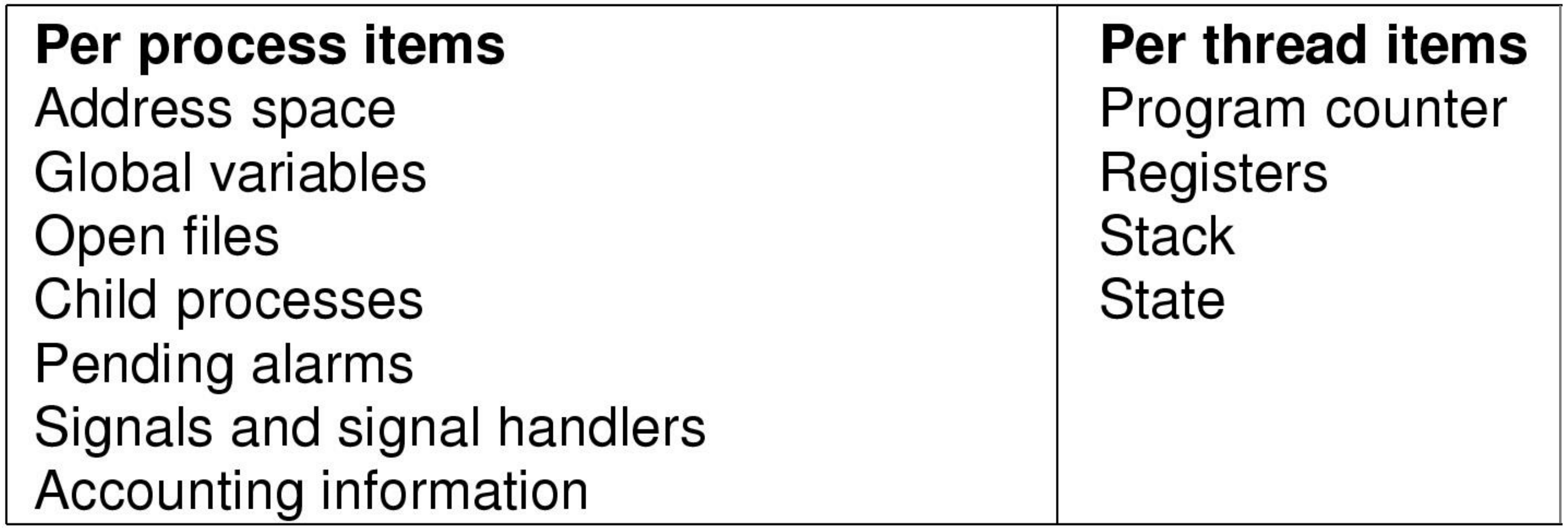


(a) Three processes each with one thread

(b) One process with three threads


one process
one thread
one process
multiple threads
multiple processes
one thread per process
multiple processes
multiple threads per process

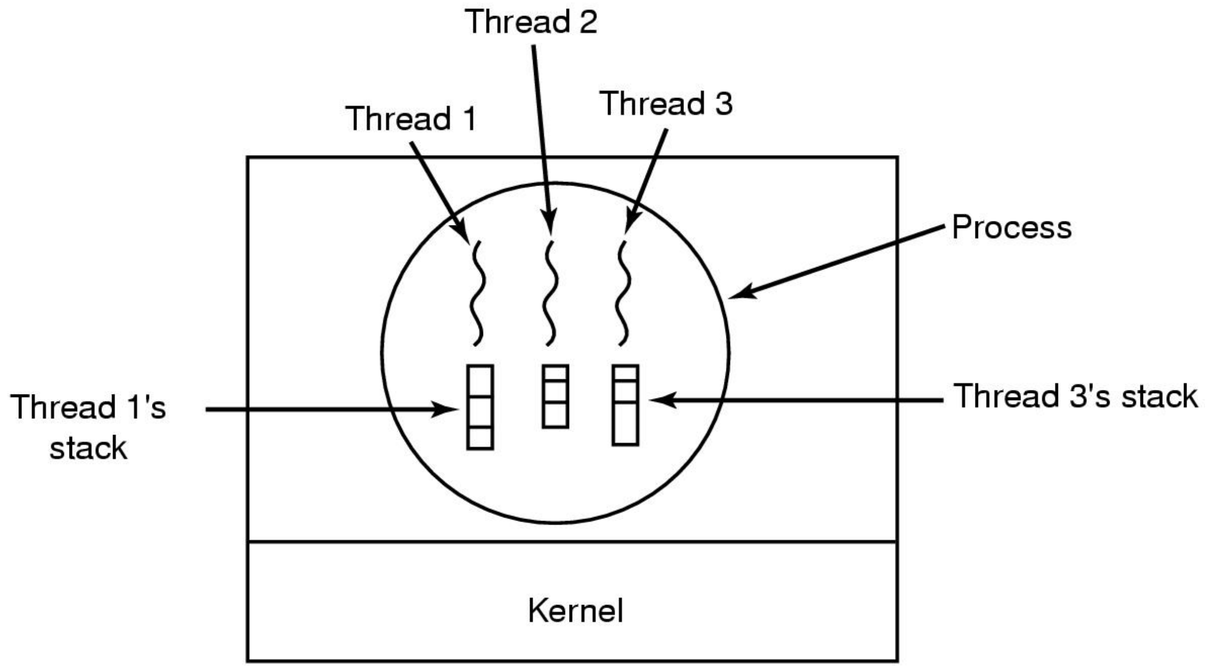
The Thread Model (2)



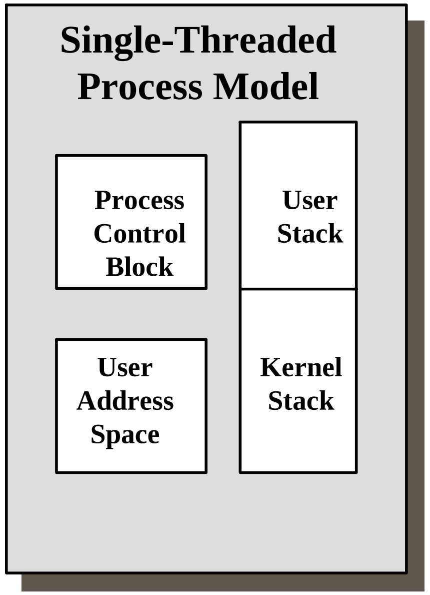
Per process items	Per thread items
Address space	Program counter
Global variables	Registers
Open files	Stack
Child processes	State
Pending alarms	
Signals and signal handlers	
Accounting information	

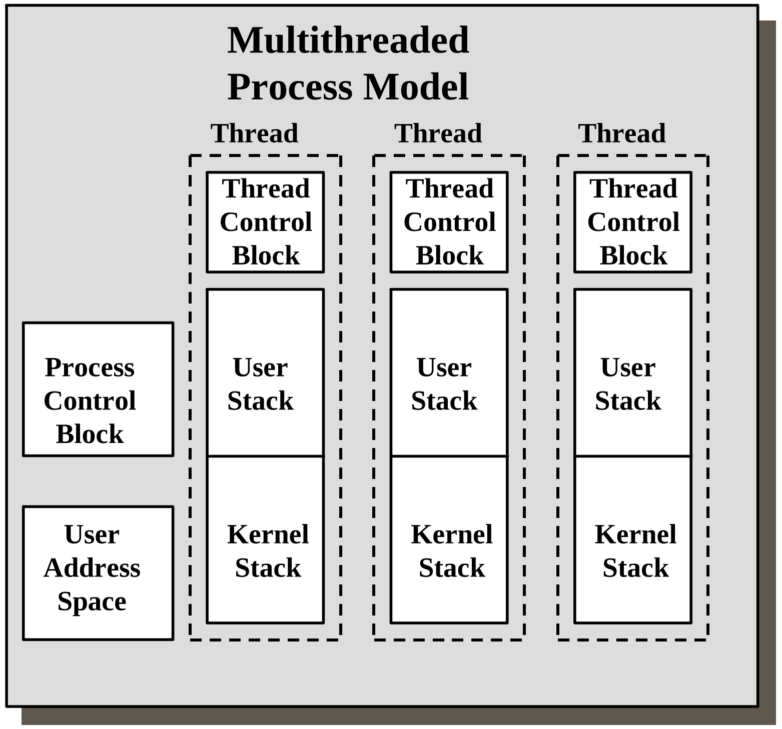
- Items shared by all threads in a process
- Items private to each thread

The Thread Model (3)

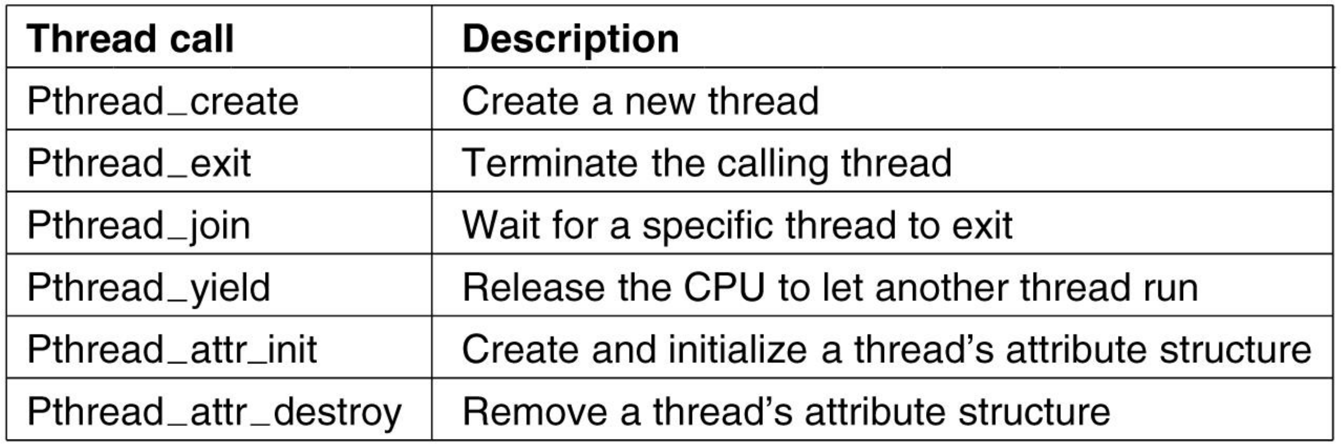


Each thread has its own stack


Single-Threaded Process Model
Process Control Block
User Address Space
User Stack
Kernel Stack


Multithreaded Process Model
Thread
Thread
Thread
Thread Control Block
Thread Control Block
Thread Control Block
Process Control Block
User Address Space
User Stack
User Stack
User Stack
Kernel Stack
Kernel Stack
Kernel Stack

POSIX Threads



Thread call	Description
Pthread_create	Create a new thread
Pthread_exit	Terminate the calling thread
Pthread_join	Wait for a specific thread to exit
Pthread_yield	Release the CPU to let another thread run
Pthread_attr_init	Create and initialize a thread's attribute structure
Pthread_attr_destroy	Remove a thread's attribute structure

Some of the Pthreads function calls.

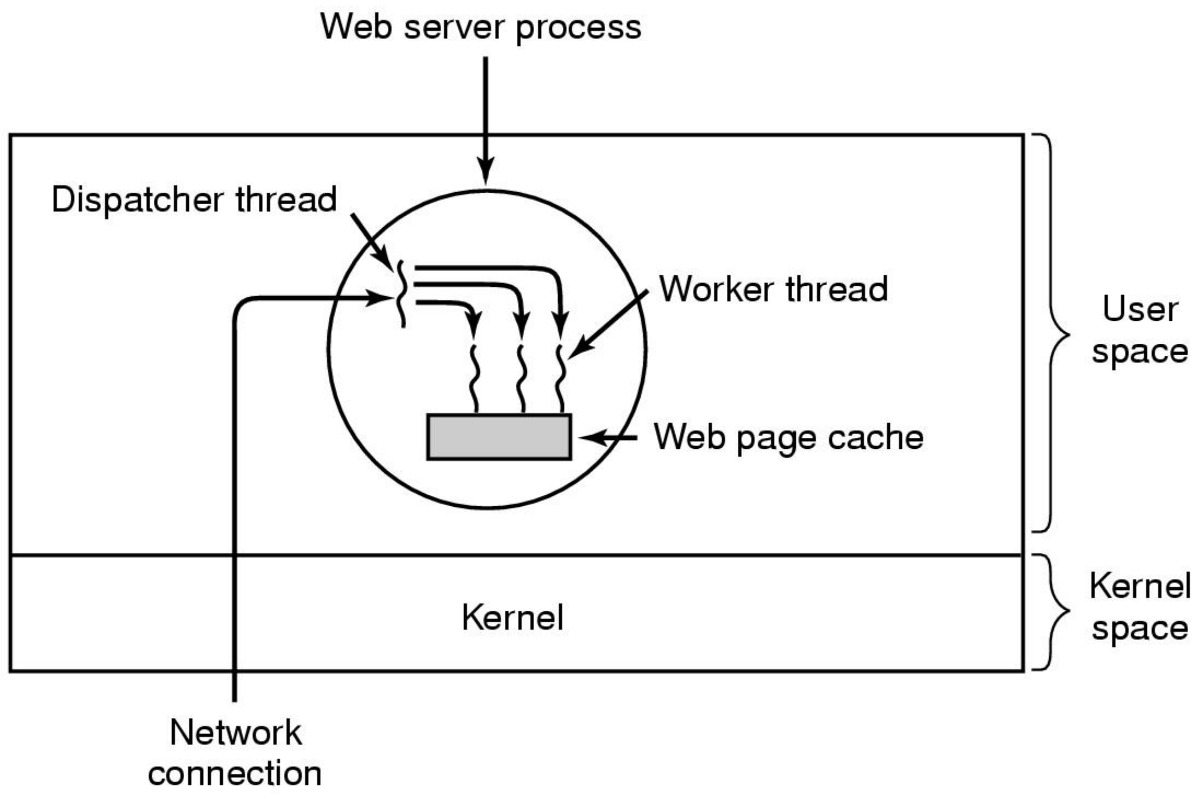
二 线程（Threads）

3、线程的使用

Thread Usage

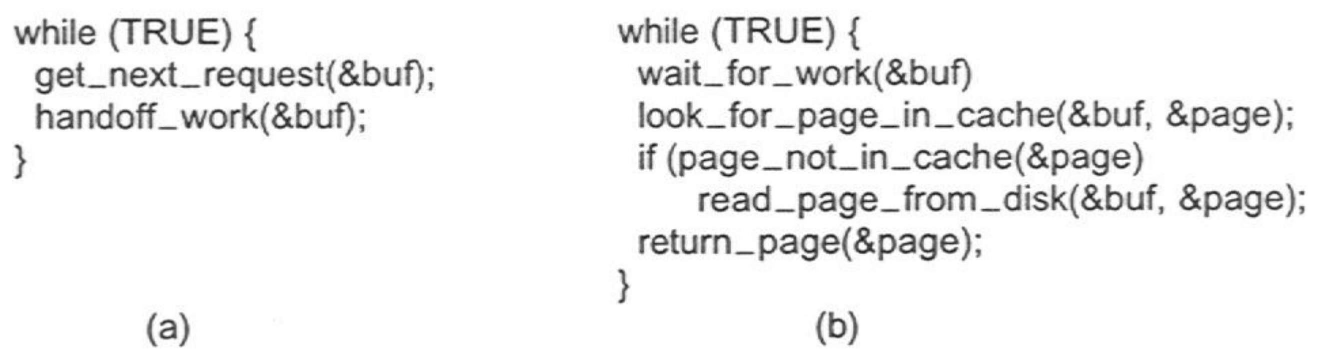
- Parallel entities
- Easy to create and destroy
- Substantial calculation & substantial I/O
- Useful on systems with CPUs

Thread Usage (2)



A multithreaded Web server

Thread Usage (3)



```
while (TRUE) {
  get_next_request(&buf);
  handoff_work(&buf);
}
```

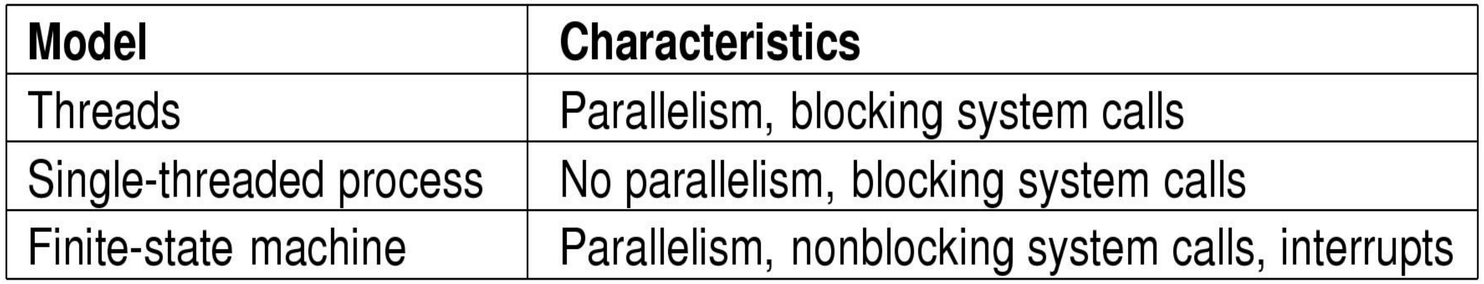
(a)

```
while (TRUE) {
  wait_for_work(&buf)
  look_for_page_in_cache(&buf, &page);
  if (page_not_in_cache(&page)
        read_page_from_disk(&buf, &page);
  return_page(&page);
}
```

(b)

- Rough outline of code for previous slide
  - (a) Dispatcher thread
  - (b) Worker thread

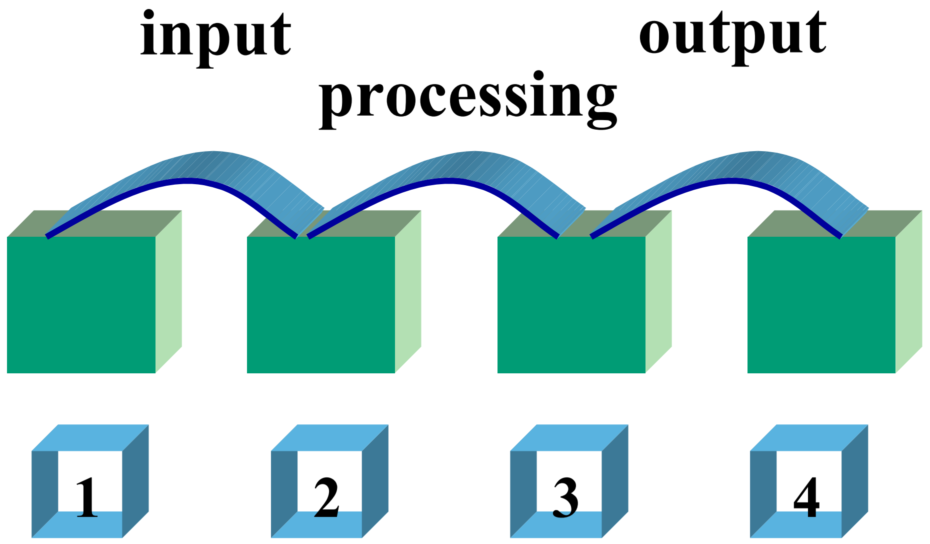
Thread Usage (4)



Model	Characteristics
Threads	Parallelism, blocking system calls
Single-threaded process	No parallelism, blocking system calls
Finite-state machine	Parallelism, nonblocking system calls, interrupts

Three ways to construct a server

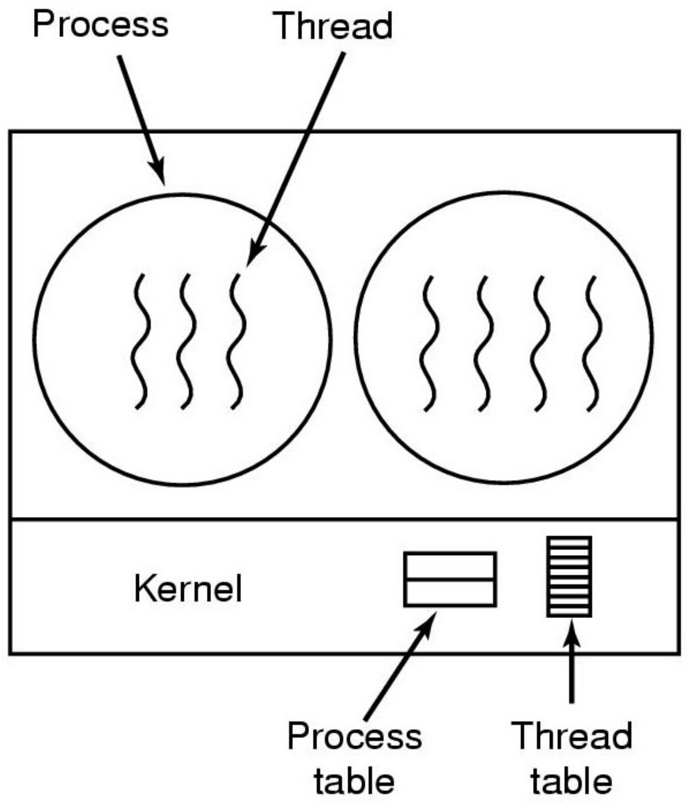
Thread Usage (5)



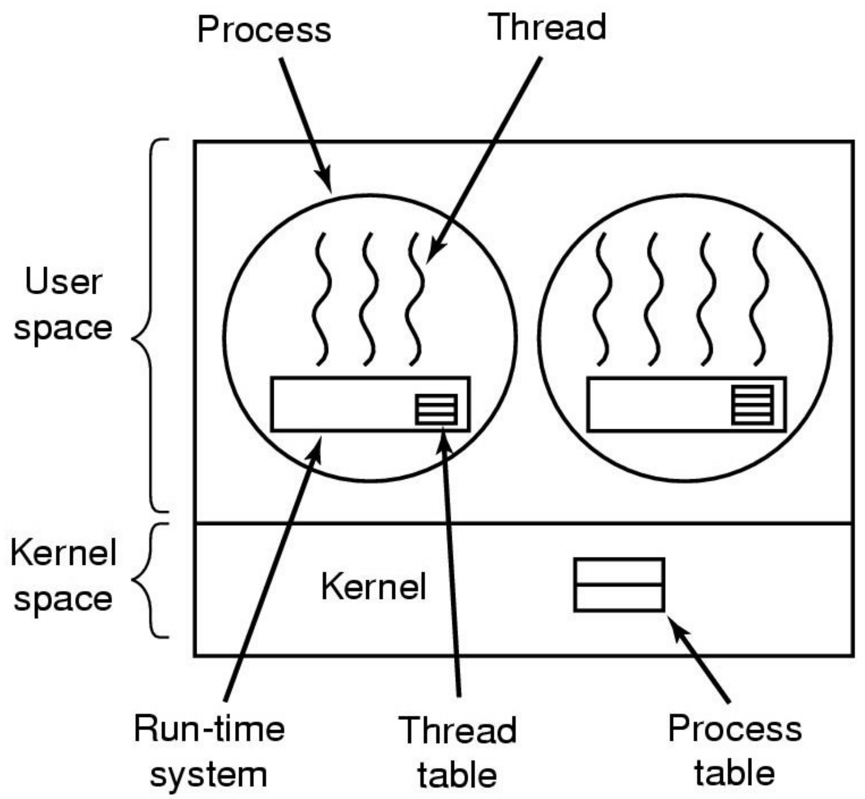
二 线程（Threads）

4、线程的实现

Implementations of Threads



A threads package managed by the kernel

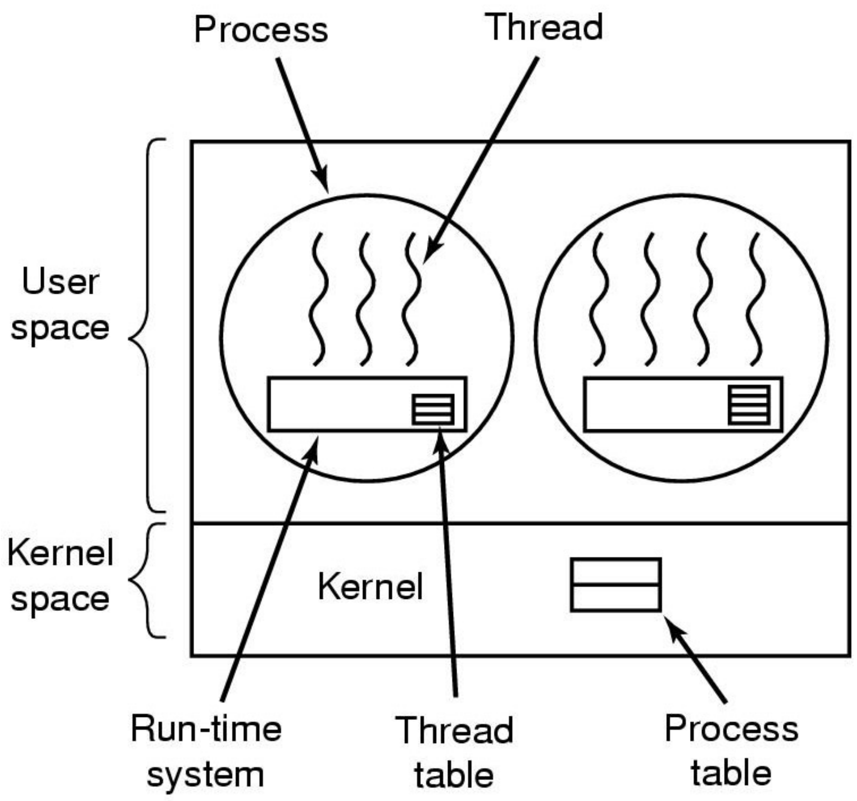


A user-level threads package

Implementing Threads in User Space

不依赖于OS内核，应用进程利用用户空间线程包提供的创建、调度和管理线程的函数来控制用户线程。

- 内核只管理进程(PT)，用户线程的维护由应用进程完成(TB)；
- 用户线程切换不需到内核态执行，切换开销更小；
- 用户线程调度算法可针对应用来优化；
- 时间片等额分配给进程，时钟中断对用户级线程不起作用，只能靠线程释放CPU才进行线程调度；
- 当线程执行系统调用阻塞会如何？

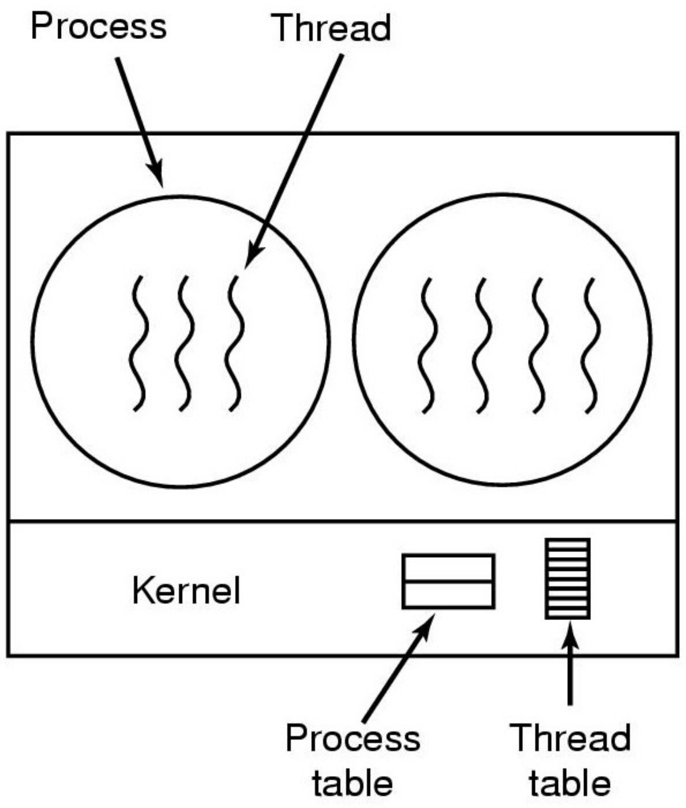


A user-level threads package

Implementing Threads in the Kernel

依赖于OS内核，通过执行内核提供的相应函数来进行线程的创建、撤销等操作。

- 进程和线程的上下文信息均由内核来维护（PT、TB）；
- 线程切换由内核完成，故存在用户态/核心态的切换开销；
- 时间片等额分配给内核线程，时钟中断对内核级线程有效，多内核线程的进程获得更多的CPU时间；
- 内核线程发起系统调用而阻塞，不会影响其他线程的运行。



A threads package managed by the kernel

小 结

- 线程的概念、作用；
- 线程和进程的区别；
  - 资源；调度；并发性；切换
- TCB；
- 用户级线程、内核级线程。