華中農業大學
HUAZHONG AGRICULTURAL UNIVERSITY

计算机操作系统

倪福川
fcni_cn@mail.hzau.edu.cn

华中农业大学 信息学院

第二章　进程的描述与控制

2.6 进 程 通 信

进程通信是指进程之间的信息交换。

- 低级进程通信　效率低　不透明
  - 互斥、同步　信号量
- 高级进程通信　方便　高效传送大量数据
  - 共享存储器
  - 管道通信
  - 消息传递
  - 客户机-服务器

2.6.1 进程通信的类型

1. 共享存储器系统(Shared-Memory System)

进程之间通过共享某些数据结构或共享存储区进行通信：

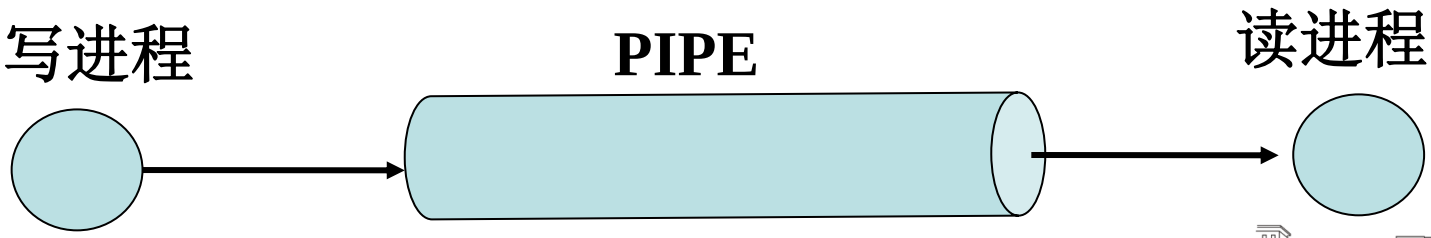
(1) 基于共享数据结构的通信方式。

(2) 基于共享存储区的通信方式。

2. 管道(pipe)通信系统

“管道”，连接一个读进程和一个写进程以实现它们之间通信的一个共享文件，又名pipe文件。

以字节流方式传送的通信通道。由OS核心的缓冲区来实现，是单向的，是一个先进先出（FIFO）的队列，由一个进程写，一个进程读



管道通信机制：

① 互斥，即当一进程正对pipe读/写操作时，其它(另一)进程必须等待。

② 同步，数据写入pipe，写进程睡眠等待，直到读进程取走数据后再把它唤醒；pipe空时，读进程睡眠等待，直至写进程将数据写入后才将之唤醒。

③ 确定对方是否存在，只有确定了对方已存在时才能进行通信。

3. 客户机-服务器系统(Client-Server system)

1) 套接字(Socket)　网络通信程序接口

2) 远程过程调用和远程方法调用

允许运行于一台主机(本地)系统上的进程调用另一台主机(远程)系统上的进程

采用面向对象编程，称远程方法调用

远程过程调用(RPC)的主要步骤：

(1) 本地过程调用者以一般方式调用远程过程在本地关联的客户存根，传递相应的参数，然后将控制权转移给客户存根；

(2) 客户存根执行，完成包括过程名和调用参数等信息的消息建立，将控制权转移给本地客户进程；

(3) 本地客户进程完成与服务器的消息传递，将消息发送到远程服务器进程；

(4) 远程服务器进程接收消息后转入执行，并根据其中的远程过程名找到对应的服务器存根，将消息转给该存根；

(5) 该服务器存根接到消息后，由阻塞状态转入执行状态，拆开消息从中取出过程调用的参数，然后以一般方式调用服务器上关联的过程；

(6) 在服务器端的远程过程运行完毕后，将结果返回给与之关联的服务器存根；

(7) 该服务器存根获得控制权运行，将结果打包为消息，并将控制权转移给远程服务器进程；

(8) 远程服务器进程将消息发送回客户端；

(9) 本地客户进程接收到消息后，根据其中的过程名将消息存入关联的客户存根，再将控制权转移给客户存根；

(10) 客户存根从消息中取出结果，返回给本地调用者进程，并完成控制权的转移。

4. 消息传递系统(Message passing system)

以格式化的消息 (message)为单位，将通信的数据封装在消息中，并利用一组通信命令(原语)，在进程间进行消息传递，完成进程间的数据交换。

其实现方式的不同，可分成两类：

(1) 直接通信方式　发送原语

(2) 间接通信方式　发送/接受进程

通过邮箱完成通信

4.1 消息传递方式

消息传递：是相互合作的并发进程交换信息的一种方式，进程间的数据交换以消息为单位。

消息队列：每个进程有一个与之相关的消息队列

当进程向一个满队列发送消息时，它将被挂起；

当进程从一个空队列读取时也会被挂起。

4.2 消息传递方式—消息

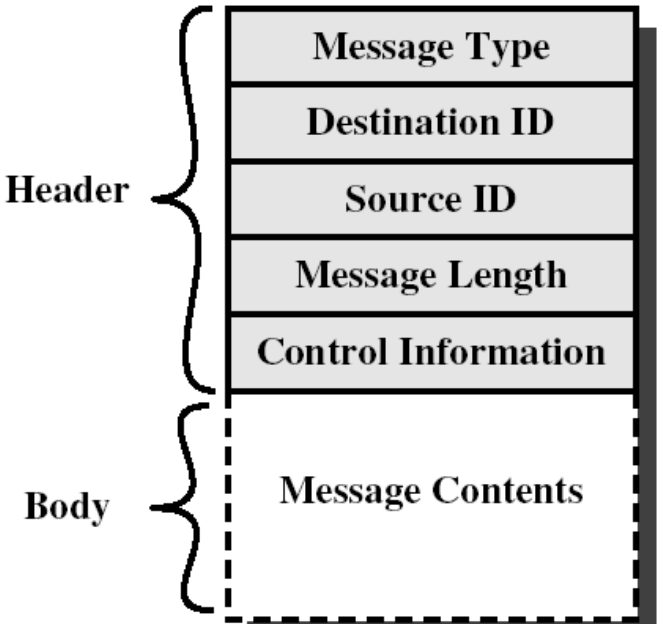
消息：一段文本。消息格式设计与应用环境和要求有关

- 固定长度消息：可以减小处理和存储的开销
- 基于文件的：传送大量的数据
- 可变长度消息：灵活

消息的一般格式

消息头：源标识、目的标识、
长度域、类型域、控制域

消息体



4.3 消息传递通信的实现方式

1. 直接消息传递系统

采用直接通信方式，发送进程利用OS所提供发送命令(原语)，直接把消息发送给目标进程。

Send(P,message):发送消息到进程P

Receive(Q,message):接收来自进程Q的消息。

(1) 对称寻址方式。发送进程和接收进程必须命名对方

(2) 非对称寻址方式。只要发送者命名接收者，而接收者不需要命名发送者

进程的同步方式

进程通信，需要同步机制协调通信。

发送进程/接收进程，完成消息发送/接收后，可能继续发送/接收或者阻塞。

①发送进程阻塞，接收进程阻塞

②发送进程不阻塞，接收进程阻塞

③发送进程阻塞，接收进程不阻塞

通信链路

在发送进程和接收进程之间建立一条通信链路。

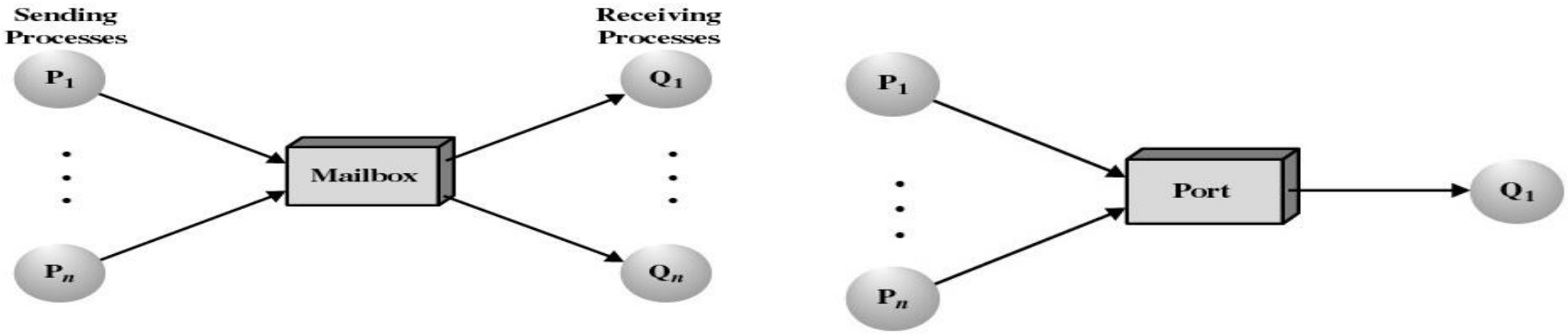
(1) 发送进程在通信之前用显式“建立连接”命令(原语)请求系统为之建立一条链路，使用完后拆除链路。 计算机网络

(2) 发送进程无须明确提出建立链路请求，利用发送命令(原语)，系统自动建立一条链路。 单机

2. 间接消息传递系统

消息通过邮箱或端口A来发送和接收。

Send（A,message）；
receive（A,message）；



信箱通信

1) 信箱的结构

(1) 信箱头：信箱的描述信息, id ,owner,size等

(2) 信箱体：存放消息的信箱格

2) 信箱通信原语

系统提供若干原语：

(1) 邮箱的创建和撤消。

(2) 消息的发送和接收。

Send（A,message）；

receive（A,message）；

3) 信箱的类型

邮箱可由操作系统创建，也可由用户进程创建，创建者是邮箱的拥有者。

邮箱分三类：

(1) 私用邮箱。

(2) 公用邮箱。

(3) 共享邮箱。

4.4 直接消息传递系统实例

1. 消息缓冲队列通信机制中的数据结构

(1) 消息缓冲区。 消息缓冲队列

(2) PCB中有关通信的数据项。

消息缓冲队列 队首指针

互斥信号量 资源信号量

2. 发送原语

```
void send(receiver，a)  {
        getbuf(a.size, i)；
        copy(i.sender，a.sender)；
        i.size=a.size；
        copy(i.text，a.text)；
        i.next=0；
        getid(PCBset，receiver.j)；
        wait(j.mutex)；
        insert(&j.mq，i)；
        signal(j.mutex)；
        signal(j.sm)；
    }
```


消息缓冲通信

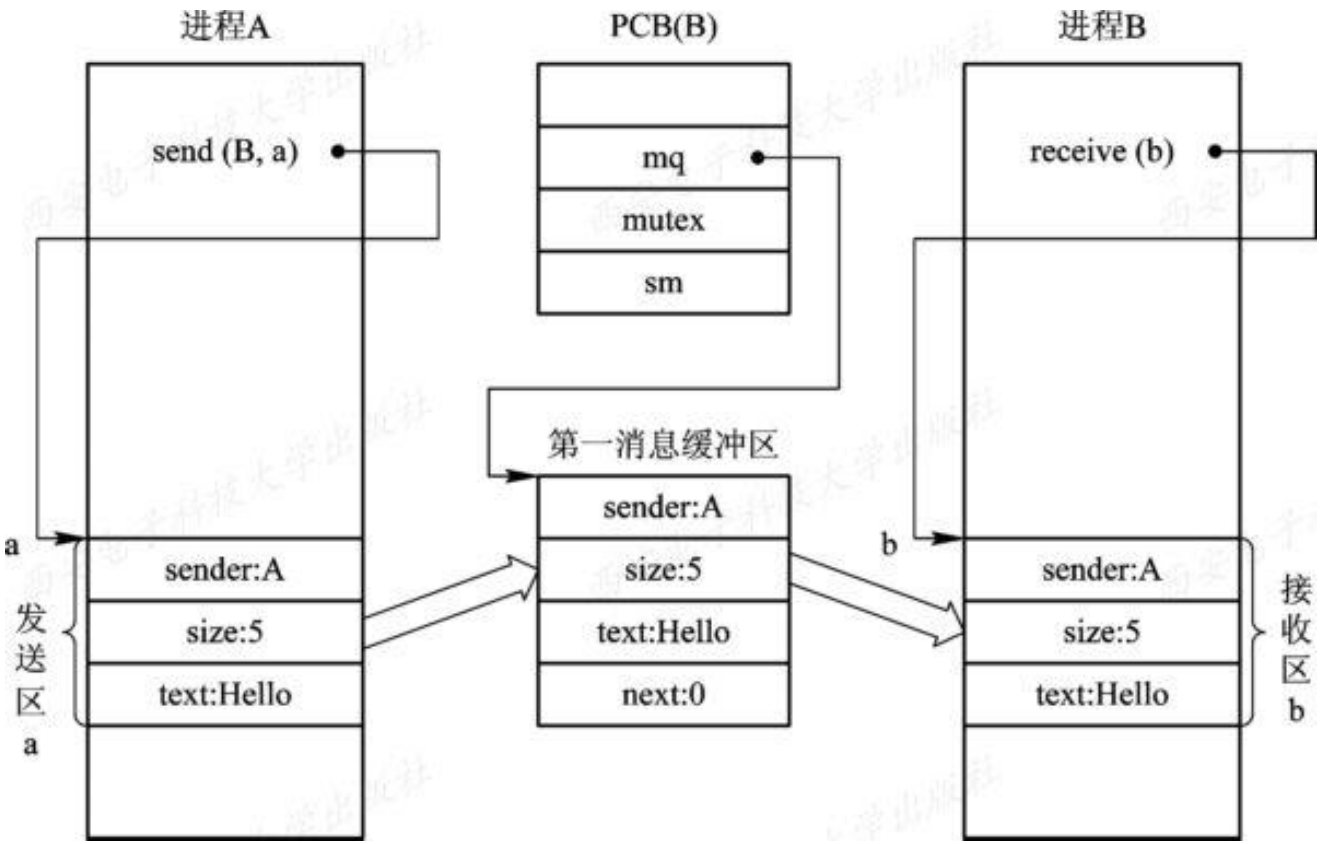


图2-17　消息缓冲通信

3. 接收原语

```
void receive(b)   {
        j = internal name;
        wait(j.sm);
        wait(j.mutex);
        remove(j.mq，  i);
        signal(j.mutex);
        copy(b.sender，  i.sender);
        b.size =i.size;
        copy(b.text，  i.text);
        releasebuf(i);
    }
```


5 信号(signal)

通知发生一个进程通信事件的软件机制，给进程的“软件”中断；

所有信号没有优先级，平等对待；

对于同时发生的信号，一次只给进程一个信号，而没有特别的次序。

单向、异步的；

进程可发送信号，指定信号处理例程；

在接收到信号时就被调用，“捕获”信号。

信号处理例程的参数是接收到信号的编号。

利用信号机制实现进程间通信

信号
发送进程

信号
接收进程

出现某事件后
发送信号

检测到发送给
自己的信号

相应
信号处理
程序

Linux系统 64种信号，表示不同事件。 Kill -l

选择scorpion@NFC-SurfacePro: ~

```
scorpion@NFC-SurfacePro:~$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
scorpion@NFC-SurfacePro:~$ ~
```


2.7 线程(Threads)的基本概念

2.7.1 线程的引入

为了减少程序在并发执行时所付出的时空开销，使OS具有更好的并发性。

进程使多个程序能并发执行，以提高资源利用率和系统吞吐量

1. 进程的两个基本属性

进程两个基本属性：

① 进程是拥有资源的独立单位；

② 进程又是独立调度也是分派的单位

2. 程序并发执行所需付出的时空开销

(1) 创建进程，分配除处理机以外所有资源，如内存空间、I/O设备，以及建立相应的PCB；

(2) 撤消进程，对占有资源回收，撤消PCB；

(3) 进程切换，进程上下文切换，保留当前进程的CPU环境，设置新选中进程的CPU环境

进程数目不宜过多；切换频率也不宜过高

3. 线程——作为调度和分派的基本单位

尽量减少系统开销，将进程两属性由OS分开处理

不把作为调度和分派的基本单位也同时作为拥有资源的单位，以做到“轻装上阵”；

对于拥有资源的基本单位，不对之施以频繁的切换。

2.7.2 线程与进程的比较 P76

1. 调度的基本单位
2. 并发性
3. 拥有资源
4. 独立性
5. 系统开销
6. 支持多处理机系统

2.7.3 线程的状态和线程控制块

1. 线程运行的三个状态

(1) 执行状态，已获得处理机而正在运行；

(2) 就绪状态，已具备了各种执行条件，只须再获得CPU便可立即执行；

(3) 阻塞状态，在执行中因某事件受阻而处于暂停状态，

2. 线程控制块TCB

控制和管理线程的信息记录在线程控制块中。

① 线程标识符
② 一组寄存器　PC　PSW　通用寄存器
③ 线程运行状态
④ 优先级
⑤ 专有存储区　存放现场信息　统计信息
⑥ 信号屏蔽
⑦ 栈指针　用户栈　系统栈

3. 多线程OS中的进程属性

在一个进程中多个线程能并发执行，此时进程就不再作为一个执行的实体。

多线程OS中进程的属性：

(1) 进程是拥有资源的基本单位。

(2) 多个线程可并发执行。

(3) 进程已不是可执行的实体。

多线程OS　线程是调度的基本单位

2.8 线 程 的 实 现

2.8.1 线程的实现方式

各系统的实现方式并不完全相同

- 用户级线程；
- 内核支持线程；
- 同时实现两种类型的线程。

1. 内核支持线程KST

内核支持线程KST的创建、阻塞、撤消和切换等，在内核空间实现；

在内核空间为内核线程设置一个线程控制块。

优点：

(1) 多处理器系统中，能同时调度同一进程中多个线程并行执行；

(2) 如线程阻塞，可调度其它线程运行，也可运行其它进程中线程；

(3) 很小的数据结构和栈，切换比较快，开销小；

(4) 内核本身用多线程，提高执行速度和效率。

缺点：

线程切换 开销大

2. 用户级线程ULT(User Level Threads)

在用户空间中实现线程创建、撤消、同步与通信等功能，无需内核支持，与内核无关。

线程的任务控制块在用户空间，无需内核帮助，内核完全不知道用户级线程存在

用户级线程优点：

(1) 线程切换不需转换到内核空间

(2) 调度算法可是进程专用

(3) 实现与OS平台无关 属用户程序，可共享。

用户级线程缺点：

(1) 系统调用的阻塞问题 线程阻塞,所在的进程内的所有线程阻塞。

(2) 不能利用多处理机多重处理 一个进程仅有一个线程能执行，其它线程只能等待

3. 组合方式

用户级线程和内核支持线程进行组合

内核支持多个内核支持线程的建立、调度和管理，同时，也允许用户应用程序建立、调度和管理用户级线程。

时分多路复用

阻塞一个线程，不再阻塞整个进程

连接方式

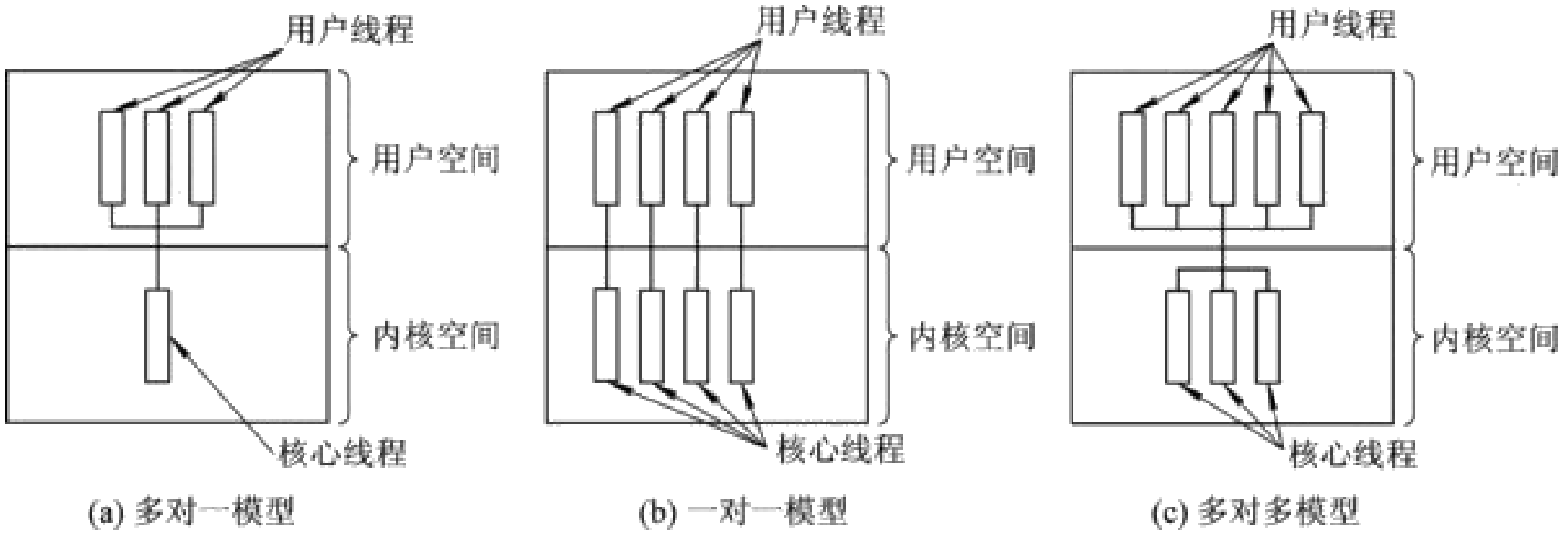


图2-18 多线程模型

2.8.2 线程的实现

1. 内核支持线程的实现

创建新进程，分配任务数据区PTDA(Per Task Data Area)，包括若干个线程控制块TCB空间。

图2-19 任务数据区空间

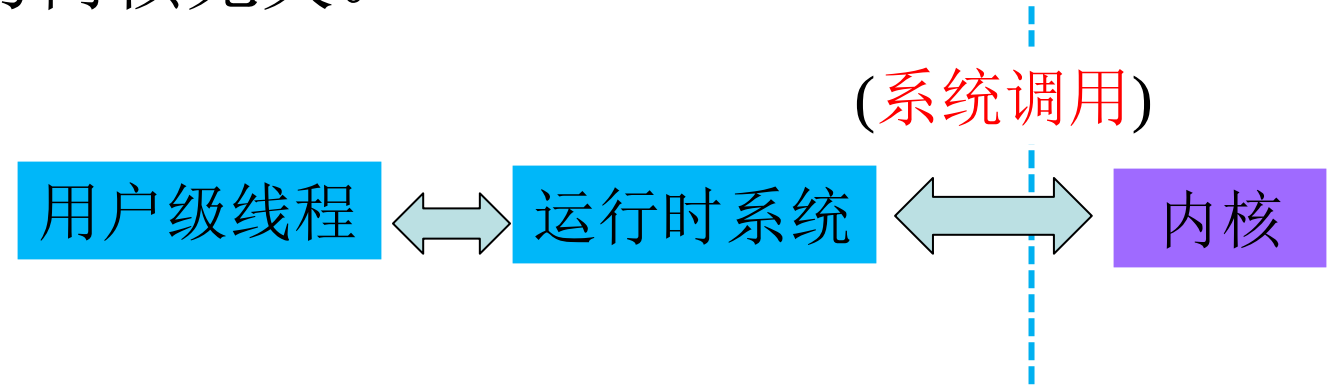
中间系统

2. 用户级线程的实现

1) 运行时系统(Runtime System)

用于管理和控制线程的函数(过程)的集合。

驻留用户空间，用户级线程与内核接口，与内核无关。



2) 内核控制线程　　组合方式

轻型进程LWP(Light Weight Process):

①有特定数据结构：线程标识符、优先级、状态，栈和局部存储区等。

②共享进程所拥有资源

③通过系统调用获得内核服务

用户级线程连接到LWP，便具有内核支持线程的所有属性

LWP实现用户级线程与内核隔离

用户进程连接轻型进程访问内核

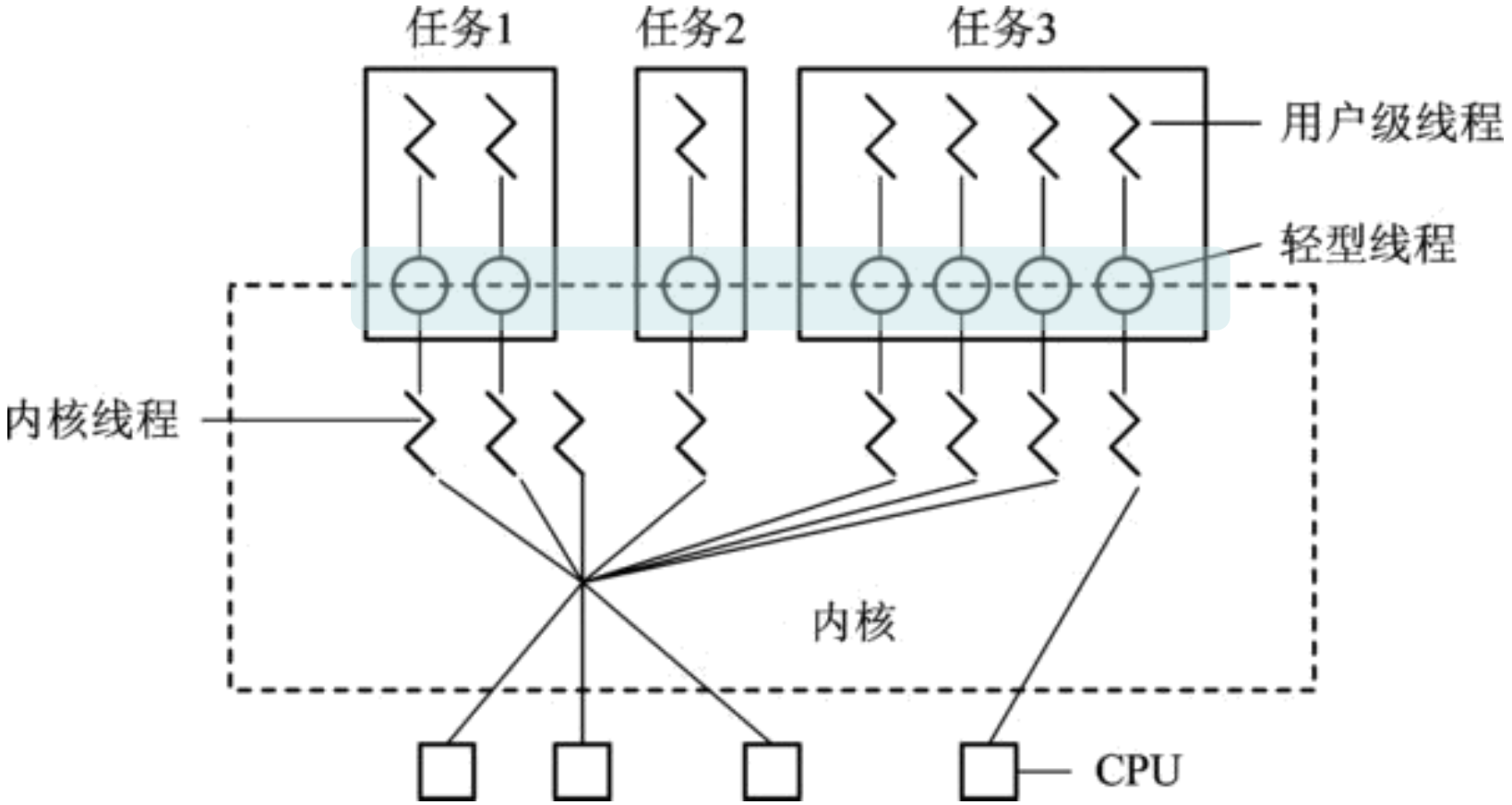


图2-20　利用轻型进程作为中间系统

2.8.3 线程的创建和终止

1. 线程的创建

应用程序启动，仅有“初始化线程”，用于创建新线程。

创建新线程：线程创建函数(或系统调用)，相应的参数(线程主程序入口指针、堆栈大小，调度的优先级等）

返回一个线程标识符

2. 线程的终止

线程完成任务，或是在运行中出现异常而被强行终止，调用相应函数(或系统调用) 终止。

多数OS，线程被中止后并不立即释放所占资源，只有其它线程执行分离函数后，才与资源分离，资源才能利用。

主观题 10分

设置

HUAZHONG AGRICULTURAL UNIVERSITY

在本课堂上，你想学习那些内容？
你有那些建议？

正常使用主观题需2.0以上版本雨课堂

作答