第二章　进程管理

翁楚良

https://chuliangweng.github.io

2023 春　ECNU

第二章进程管理 提纲

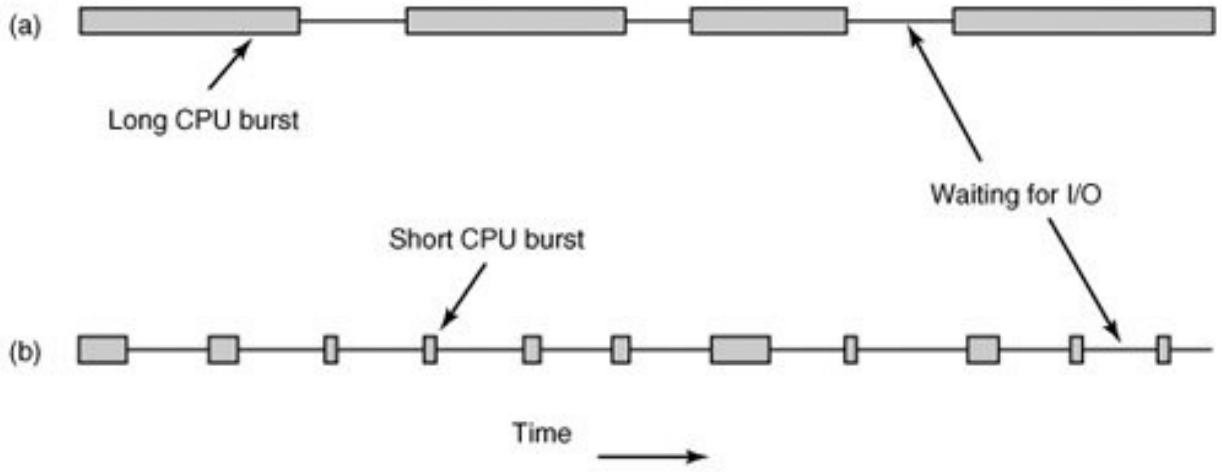
- 2.1 进程
- 2.2 进程间通信
- 2.3 经典并发问题
- 2.4 进程调度
- 2.5 MINIX3进程概述
- 2.6 MINIX3进程实现
- 2.7 MINIX3系统任务
- 2.8 MINIX3时钟任务

2.4 进程调度

- 调度简介
- 批处理系统调度
- 交互式系统调度
- 实时系统调度
- 调度策略与机制
- 线程调度

进程行为

- 进程在运行中，会交替地需要CPU进行计算、需要进行I/O操作
- 按CPU计算时间与I/O时间的比例，进程可以分为：
  - 计算密集型：进程的大部分时间花在CPU运算上
  - I/O密集型：进程的大部分时间花I/O操作上



调度时机

- 进程退出
- 进程在I/O或信号量上阻塞
- 创建新进程
- I/O中断发生
- 时钟中断发生
  - 抢占式调度
  - 非抢占式调度

调度算法

- 当多个进程竞争CPU资源时，操作系统需要决定进程运行的先后次序，操作系统中实现这一功能的部分即为调度器，确定进程运行次序的算法即为调度算法。
- 不同的应用环境需要相应合适的调度算法
  - 批处理系统
  - 交互式系统
  - 实时系统

调度算法的目标

- 共同的目标
  - 公平性、策略强制性、平衡性
- 批处理系统
  - 最大化吞吐量、最小化周转时间、最大化CPU利用率
- 交互式系统
  - 最小化响应时间、均衡性
- 实时系统
  - 满足截止时间、可预测性

性能指标

- 面向用户的调度性能指标
  - 周转时间：作业从提交到完成（得到结果）所经历的时间。包括：在输入队列中等待，CPU上执行，就绪队列和阻塞队列中等待，结果输出等待 — 批处理系统
  - 响应时间：用户输入一个请求（如击键）到系统给出首次响应（如屏幕显示）的时间 — 分时系统
  - 截止时间：开始截止时间和完成截止时间 — 实时系统，与周转时间有些相似。
  - 公平性：不因作业或进程本身的特性而使上述指标过分恶化。如长作业等待很长时间。

性能指标

- 面向系统的调度性能指标
  - 吞吐量
    - 单位时间内所完成的作业数，跟作业本身特性和调度算法都有关系
  - CPU利用率
    - CPU作计算的时间与总时间的比值
  - 各种设备的均衡利用：
    - 如CPU繁忙的作业和I/O繁忙（指次数多，每次时间短）的作业搭配

2.4 进程调度

- 调度简介
- 批处理系统调度
- 交互式系统调度
- 实时系统调度
- 调度策略与机制
- 线程调度

先来先服务(FCFS, First Come First Service)

- 按照作业提交或进程变为就绪状态的先后次序，分派CPU
- 当前作业或进程占用CPU，直到执行完或阻塞，才出让CPU（非抢占方式）
- 在作业或进程唤醒后（如I/O完成），并不立即恢复执行，通常等到当前作业或进程出让CPU。是最简单的算法。
- FCFS的特点
  - 比较有利于长作业，而不利于短作业。
  - 有利于CPU繁忙的作业，而不利于I/O繁忙的作业。

短作业优先(SJF, Shortest Job First)

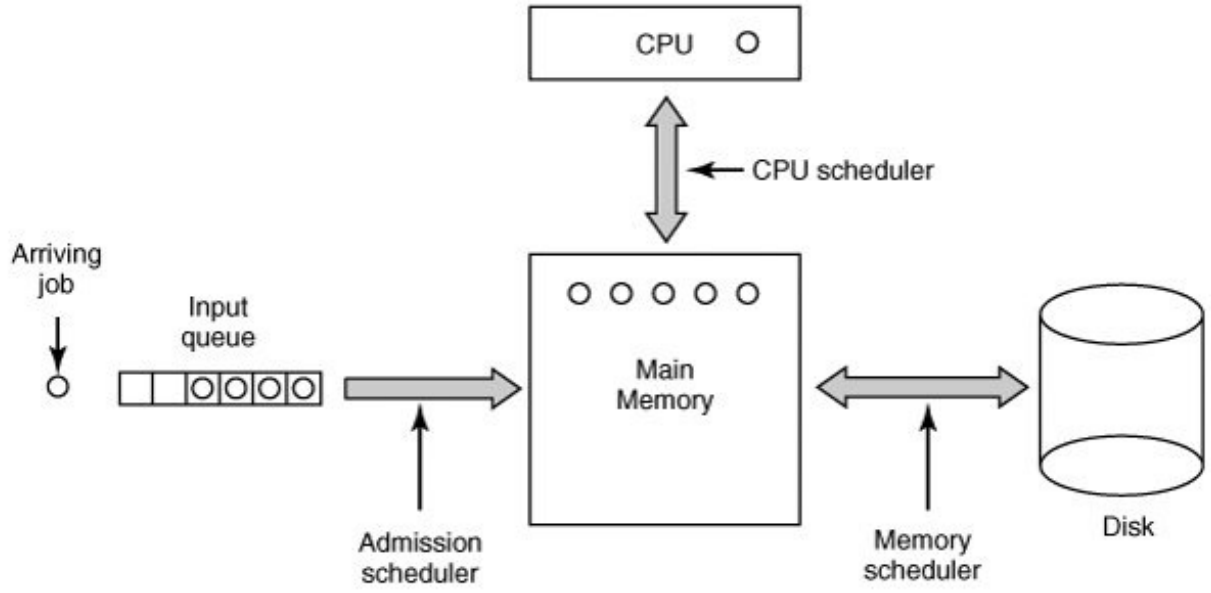
- 对预计执行时间短的作业（进程）优先分派CPU。通常后来的短作业不抢先正在执行的作业。
- 优点
  - 比FCFS改善平均周转时间，缩短作业的等待时间；
  - 提高系统的吞吐量；
- 缺点
  - 对长作业非常不利，可能长时间得不到执行；
  - 未能依据作业的紧迫程度来划分执行的优先级；
  - 难以准确估计作业（进程）的执行时间，从而影响调度性能。

SJF的变型

- "最短剩余时间优先"SRT(Shortest Remaining Time)
  - 允许比当前进程剩余时间更短的进程来抢占
- "最高响应比优先"HRRN(Highest Response Ratio Next)
  - 响应比R = (等待时间 + 要求执行时间) / 要求执行时间
  - 是FCFS和SJF的折衷

三级调度(在三个层次上进行调度)

- 准入调度器
  - 选取哪些作业允许进入系统，未选中的作业临时保存在输入队列中
- 内存调度器
  - 确定哪些进程驻留内存，哪些进程需要暂存到磁盘中
- CPU调度器
  - 选取将要运行的进程



2.4 进程调度

- 调度简介
- 批处理系统调度
- 交互式系统调度
- 实时系统调度
- 调度策略与机制
- 线程调度

时间片轮转(Round Robin)

- 其基本思路是通过时间片轮转，提高进程并发性和响应时间特性，从而提高资源利用率；
- 将系统中所有的就绪进程按照FCFS原则，排成一个队列。
- 每次调度时将CPU分派给队首进程，让其执行一个时间片。时间片的长度从几个ms到几百ms。
- 在一个时间片结束时，发生时钟中断。
- 调度程序据此暂停当前进程的执行，将其送到就绪队列的末尾，并通过上下文切换执行当前的队首进程。
- 进程可以未使用完一个时间片，就出让CPU（如阻塞）。

时间片长度的确定

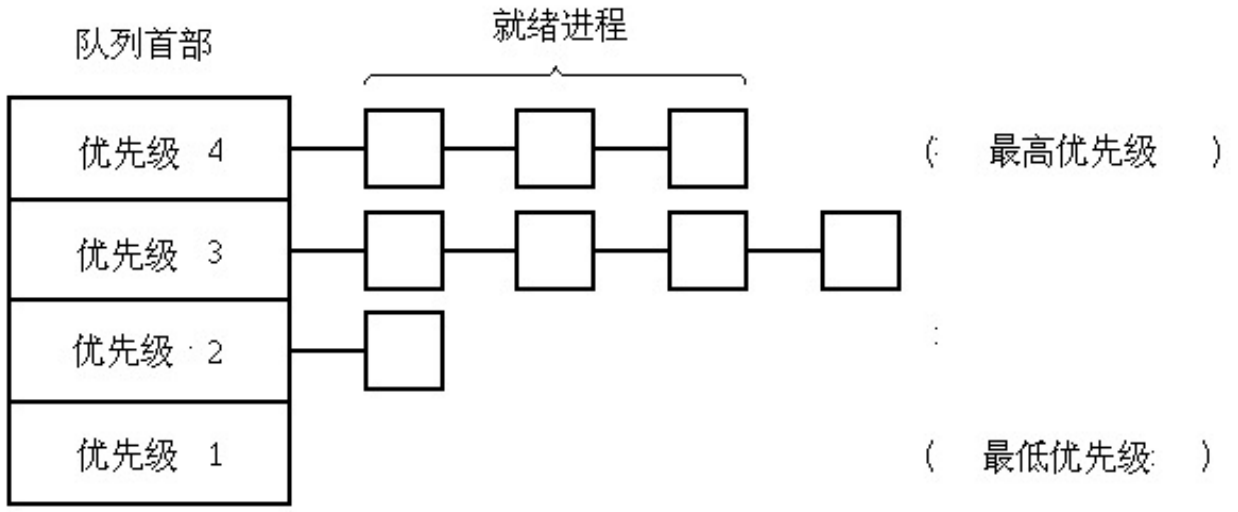
- 时间片长度变化的影响
  - 若过长，该算法退化为FCFS算法，进程在一个时间片内都执行完，响应时间长。
  - 若过短，用户的一次请求需要多个时间片才能处理完，上下文切换次数增加，降低CPU效率。
- 对响应时间的要求：
  - 响应时间=进程数目× 时间片长度
- 时间片长度的影响因素：
  - 就绪进程的数目
  - 系统的处理能力

优先级调度

- 其基本思想是给每个进程赋予一个优先级，优先调度运行优先级最高的就绪进程。
- 静态优先级
  - 创建进程时就确定，直到进程终止前都不改变。通常是一个整数。确定优先级的依据：
    - 进程类型（系统进程优先级较高）
    - 对资源的需求（对CPU和内存需求较少的进程，优先级较高）
    - 用户要求（紧迫程度和付费多少）
- 动态优先级
  - 在创建进程时赋予的优先级，在进程运行过程中可以自动改变，以便获得更好的调度性能。例如：
    - 在就绪队列中，等待时间延长则优先级提高，从而使优先级较低的进程在等待足够的时间后，其优先级提高到可被调度执行；
    - 进程每执行一个时间片，就降低其优先级，从而一个进程持续执行时，其优先级降低到出让CPU。

多类优先级调度

- 在各类之间采用优先级调度，而各类进程内部采用时间片轮转调度
- 四类优先级的系统，调度算法如下：
  - 只要存在优先级为第4类的就绪进程，就按照时间片轮转法使其运行一个时间片，此时不理会较低优先级的进程。若第4类进程为空，则运行第3类进程。若第4、第3类均为空，则按时间片法运行第2类进程。
  - 如果对优先级不经常进行调整，则低优先级进程很可能会发生饥饿。



多重队列

- 设置多个就绪队列，分别赋予不同的优先级，如逐级降低。每个队列执行时间片的长度也不同，规定优先级越低则时间片越长，如逐级加倍。
- 新进程进入内存后，先投入最高优先级队列的末尾，按FCFS算法调度；若按在最高优先级队列中一个时间片未能执行完，则降低投入到次高优先级队列的末尾，同样按FCFS算法调度；如此下去，降低到最后的队列，则按"时间片轮转"算法调度直到完成。
- 仅当较高优先级的队列为空，才调度较低优先级的队列中的进程执行。如果进程执行时有新进程进入较高优先级的队列，则抢先执行新进程，并把被抢先的进程投入原队列的末尾。

彩票调度算法

- 其基本思想是为进程发放针对系统各种资源（如CPU时间）的彩票，当调度程序需作出决策时，随机选择一张彩票，持有该彩票的进程将获得系统资源
- 中奖的机会与其持有的彩票数成正比
  - 被调度的机会与持有的票数成正比
- 合作进程如果愿意的话可以交换彩票
- 自动地按照正确的比例分配CPU资源

公平分享调度

- 以分配用户的资源比例为基础，确定用户中进程获得的CPU时间比例
- 如两个用户，需要保证各使用一半的CPU资源
  - 若用户1运行4个进程(A,B,C,D)，用户2运行1个进程(E)，则可能的调度序列为：
    - A, E, B, E, C, E, D, E, A, E, B, C, E, D, E, (轮转调度)
  - 若用户1占用CPU时间是用户2的两倍，则可能序列为：
    - A, B, E, C, D, E, A, B, E, C, D, E, (轮转调度)

2.4 进程调度

- 调度简介
- 批处理系统调度
- 交互式系统调度
- 实时系统调度
- 调度策略与机制
- 线程调度

实时系统调度

- 实时系统通常分为硬实时(hard real time)系统和软实时(soft real time)系统
  - 硬实时系统必须满足的时间限制
  - 软实时系统偶尔可以超过时间限制
- 可调度的实时系统
  - 针对多个周期的事件流，若有周期性事件 m 个，事件 i 的周期为 P_i，其中每个事件需要 C_i 秒的CPU时间来处理，则需满足以下条件:

$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$$

- 实时调度算法可以是动态或静态

2.4 进程调度

- 调度简介
- 批处理系统调度
- 交互式系统调度
- 实时系统调度
- 调度策略与机制
- 线程调度

调度策略与机制

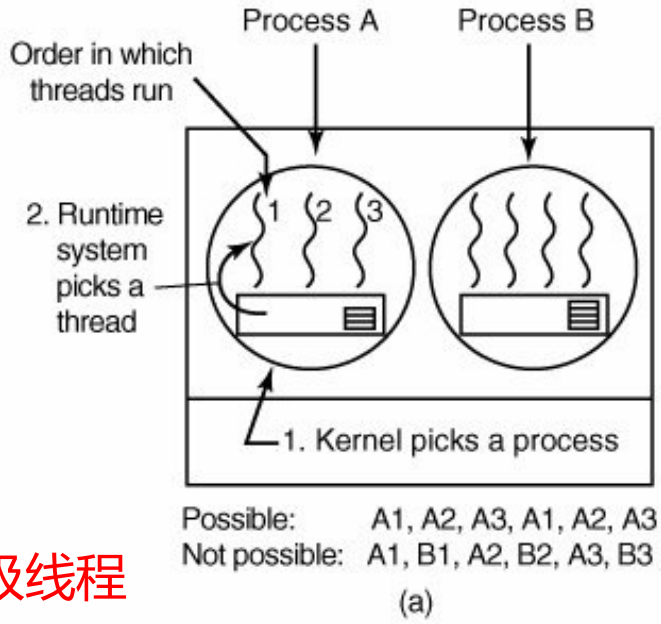
- 将调度问题分解为调度策略与调度机制
  - 将调度算法以某种形式参数化(调度机制)，而参数可以由用户进程依据相应的策略设置 (调度策略)。
- 例：数据库管理系统
  - 主进程完全需要掌握哪个子进程最重要（或最紧迫），哪个最不重要。
  - 若内核使用优先级调度算法，且提供一条系统调用，则该主进程可以使用它来设置或改变其子进程的优先级。

2.4 进程调度

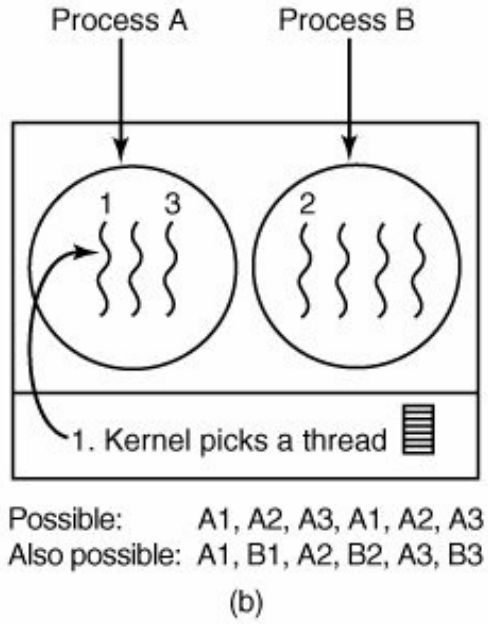
- 调度简介
- 批处理系统调度
- 交互式系统调度
- 实时系统调度
- 调度策略与机制
- 线程调度

线程调度

- 若系统中进程包含了多线程，则系统中存在两个层次上的并行性：进程级和线程级。
- 线程的实现方式决定线程级的调度模式



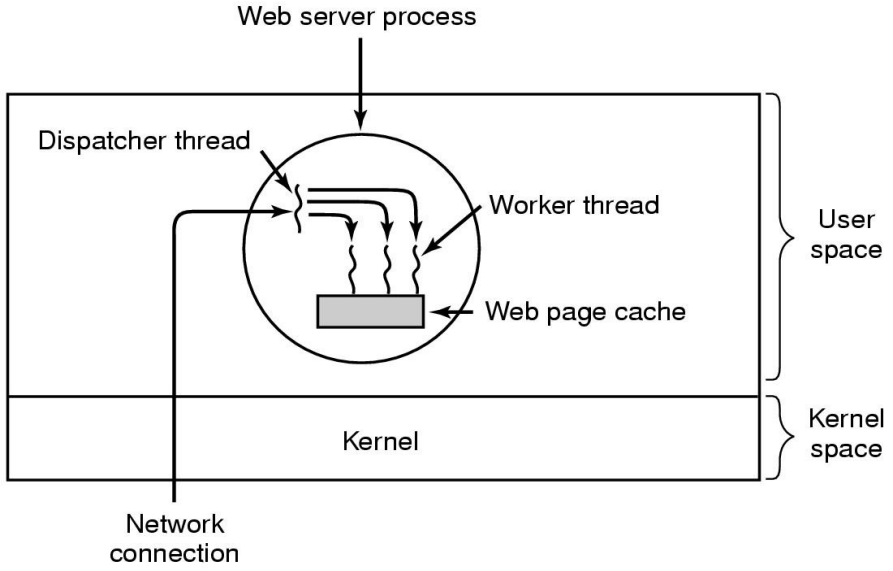
用户级线程



内核级线程

线程实现方式对调度的影响

- 用户级线程的切换只需要几条指令，无需上下文切换、修改内存映射、高速缓存失效等，时间快，而内核级线程相反
- 用户级线程的阻塞会引起整个进程的阻塞
- 在用户级线程方式下，可以为应用定制特定的调度器



第二章进程管理 提纲

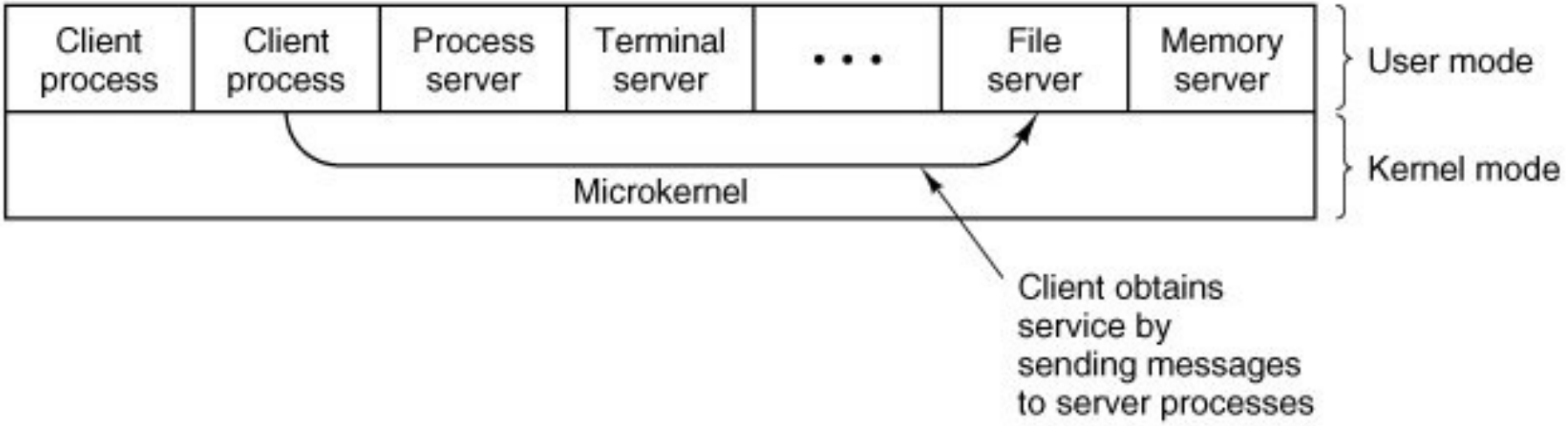
- 2.1 进程
- 2.2 进程间通信
- 2.3 经典并发问题
- 2.4 进程调度
- 2.5 MINIX3进程概述
- 2.6 MINIX3进程实现
- 2.7 MINIX3系统任务
- 2.8 MINIX3时钟任务

MINIX进程概述

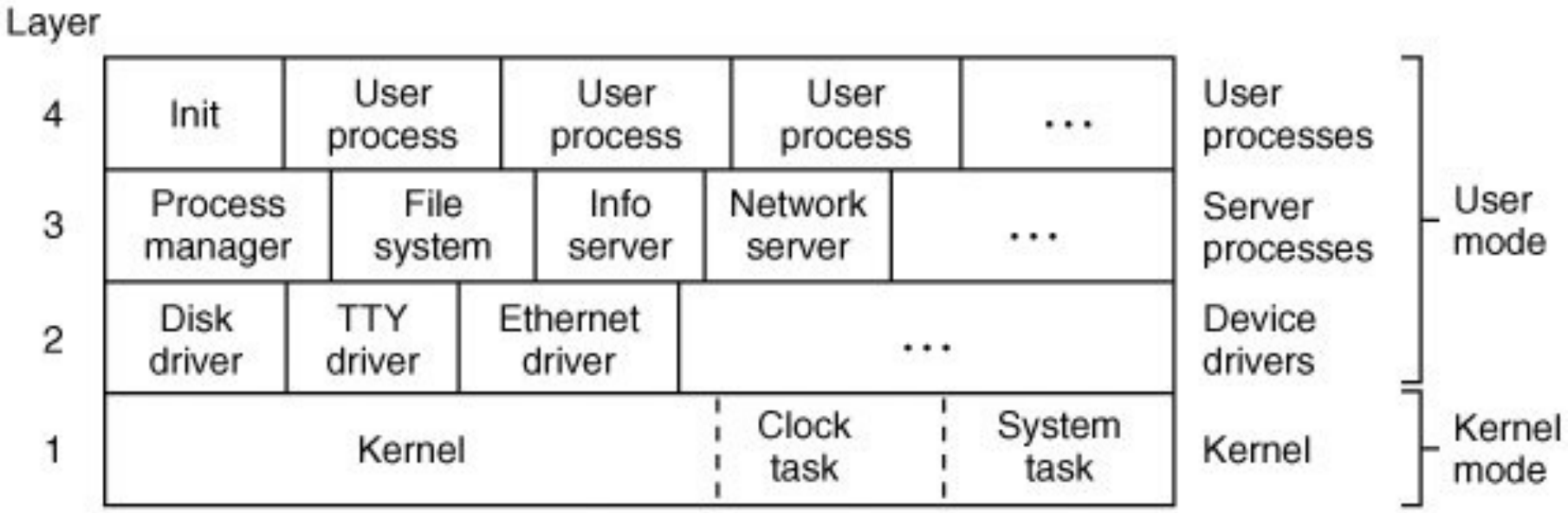
- MINIX3系统结构
- MINIX3进程管理
- MINIX3进程间通信
- MINIX3进程调度

MINIX 3 系统结构类型

- 微内核结构, 是一组进程的集合，内核功能较少, 更多的功能由用户进程实现, 进程间通过IPC机制进行通信



MINIX 3的四层结构



内核层

- 处于最底层，运行于内核模式
- 负责进程的调度、处理所有进程间通信、支持对I/O端口和中断的访问
- 时钟任务模块，产生时钟信号
- 系统任务模块，读写I/O端口、跨地址空间复制数据
- C语言+汇编语言实现

设备驱动层

- 调用内核中的系统任务模块，读写I/O端口的数据
- 不同类型设备，需要相应设备驱动程序
  - 磁盘、打印机、网络接口
- 也可以调用其它内核调用，……

服务器层

- 进程服务器
  - 创建、终止进程：fork, exec, wait
  - 信号相关的系统调用：alarm, kill
  - 管理内存：brk
- 文件系统服务器
  - 执行文件系统的调用：read, mount, chdir
- 再生服务器
  - 启动或重启相关的设备驱动程序等
- 其它……

用户进程层

- 包括所有的用户进程
  - Shell、编辑器、编译器等工具程序
  - 用户运行的程序
  - 守护进程：周期性运行的后台进程
  - Init进程

相关说明

- 设备驱动层和服务器层的进程统称为系统进程，它们是操作系统的组成部分
- 一般情况下的进程优先级
  - 内核进程>设备驱动层进程>服务器层进程>用户进程层进程

MINIX进程概述

- MINIX3系统结构
- MINIX3进程管理
- MINIX3进程间通信
- MINIX3进程调度

MINIX3启动

- 启动顺序
  - Bootstrap → boot (boot image) → Kernel → 系统任务、时钟任务 → 系统进程 → init进程 → 其它用户进程及……
- 引导映像(boot image)
  - 由boot装载至内存的适当位置
  - 包括多个部分：内核、进程管理器、文件系统、…
  - 每个部分都是独立的程序

进程树初始化

- init进程
  - 引导映像最后加载的一个进程
  - PID为1，但不是系统中运行的第一个进程
  - 执行/etc/rc脚本，结合再生服务器进程，启动其它的驱动程序和服务器
  - 执行/etc/rc脚本，启动守护进程
  - 读取其它配置文件，进行系统检查、启动终端设备、启动用户登录界面、启动shell、……

MINIX3重要组成部分

Component	Description	Loaded by
kernel	Kernel + clock and system tasks	(in boot image)
pm	Process manager	(in boot image)
fs	File system	(in boot image)
rs	(Re)starts servers and drivers	(in boot image)
memory	RAM disk driver	(in boot image)
log	Buffers log output	(in boot image)
tty	Console and keyboard driver	(in boot image)
driver	Disk (at, bios, or floppy) driver	(in boot image)
init	parent of all user processes	(in boot image)
floppy	Floppy driver (if booted from hard disk)	/etc/rc
is	Information server (for debug dumps)	/etc/rc
cmos	Reads CMOS clock to set time	/etc/rc
random	Random number generator	/etc/rc
printer	Printer driver	/etc/rc

MINIX3 进程列表

F	S	UID	PID	PPID	PGRP	SZ	RECV	TTY	TIME	CMD
0	R	0	(-4)	0	0	72		?	2h01	idle
10	W	0	(-3)	0	0	72	ANY	?	0:00	clock
0	R	0	(-2)	0	0	72		?	0:10	system
2	W	0	(-1)	0	0	72		?	0:00	kernel
10	W	0	0	5	90	120	ANY	?	0:03	pm
10	W	0	4	5	0	4956	mem	?	0:02	fs
10	W	0	5	1	0	48	ANY	?	0:00	rs
10	W	0	8	5	0	304	system	?	0:00	mem
10	W	0	9	5	0	76	ANY	?	0:00	log
10	W	0	7	5	0	84	ANY	?	0:01	tty
10	W	0	6	5	0	24	ANY	?	0:00	ds
10	S	0	1	5	1	16	(wait) pm	?	0:00	init
10	W	0	12	5	0	48	ANY	?	0:00	/bin/pci
10	W	0	14	5	0	24	ANY	?	0:00	/bin/floppy
10	W	0	17	5	0	88	ANY	?	0:00	/bin/at_wini
10	W	0	24	5	0	272	ANY	?	0:00	/sbin/is
10	W	0	26	5	0	16	ANY	?	0:00	/sbin/cmos

MINIX进程概述

- MINIX3系统结构
- MINIX3进程管理
- MINIX3进程间通信
- MINIX3进程调度

MINIX3 IPC原语

- 内核将消息从发送者复制到接收者
- 系统进程、时钟任务、系统任务只允许与特定的进程通信
- 当一个进程发送消息，而目标进程尚未开始接收消息时，发送进程将阻塞

```
send(dest, &message);

receive(source, &message);

sendrec(src_dst, &message);
```

notify原语

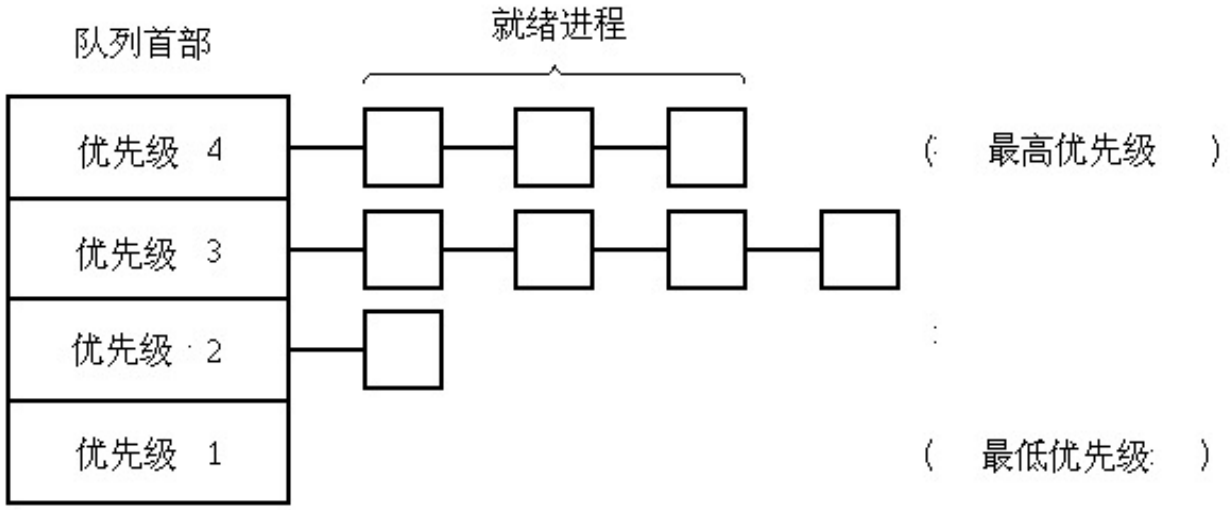
- 用于一个进程向另一个进程发送通知消息
- notify调用不会阻塞
- 通知中可以是最简单的内容：发送者身份和时间戳
- 对于系统进程，可以采用bitmap for pending notifications实现
  - 每个系统进程有这样一个位图，其中的每一位对应于一个系统进程，用于记录其它系统进程发给它的通知
  - 当系统进程调用receive时，通过检查自己的位图，可以知道其它系统进程是否发通知给它

MINIX进程概述

- MINIX3系统结构
- MINIX3进程管理
- MINIX3进程间通信
- MINIX3进程调度

MINIX3进程调度(1)

- MINIX 3调度器采用多级排队系统，定义16个队列
  - 系统任务和时钟任务处于最高优先级
  - IDLE进程在系统没有其它进程时运行，具有最低的优先级
  - 一个进程可以根据需要在不同优先级的队列中移动
  - 只有高优先级的队列中没有就绪进程，才运行低优先级的进程队列



MINIX3进程调度(2)

- 不同优先级中的进程的时间片长度不一
  - 驱动程序进程和服务器进程通常需要运行至其阻塞，为此它们分配较大的时间片
- 使用一个改进的轮转调度算法
  - 进程转为非就绪时，它的时间片没有用完，则表明发生I/O阻塞，则该进程再次转为就绪，将其放在队首，分配的时间片为上次时间片中的剩余时间
  - 一个用完时间片的进程，则以正常的时间片轮转调度方式放在队尾