

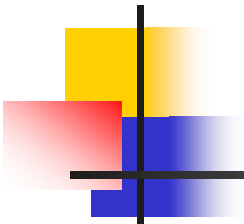
嵌入式Linux学习七步曲

Sailor_forever(扬帆)

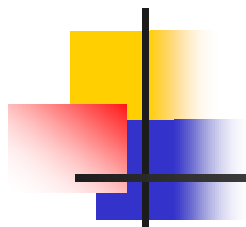
自由传播 版权所有 翻版必究



八一卦-我是who

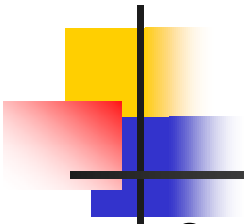
- 
-
- n 目前就职于通信行业某外企研发中心
 - n 参与校园招聘和社会招聘的技术面试工作
 - n 5年嵌入式软件开发经验，擅长嵌入式Linux开发；
 - n 接触的软硬件平台包括ARM，DSP，PowerPC，uC/OS-II，Linux，VxWorks及OSE

八一卦-我是who



- n 嵌入式Linux七步曲 学习群 交流讨论 资源共享
- n 群号 107900817
- n 7steps2linux@gmail.com
- n http://blog.csdn.net/sailor_8318

嵌入式水平小调查



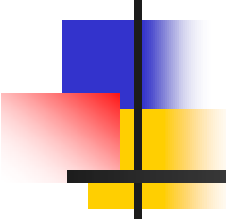
n 0—3个月

n 3—6 个月

n 1年左右

n 2年以上

n 多少人参加过系列交流会？



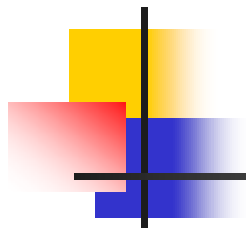
嵌入式Linux学习七步曲

- 
- 1 Linux主机开发环境
 - 2 嵌入式Linux交叉开发环境
 - 3 Linux系统bootloader移植
 - 4 Linux的内核移植
 - 5 Linux的内核及驱动开发
 - 6 文件系统制作
 - 7 Linux的高级应用编程

Key To Success

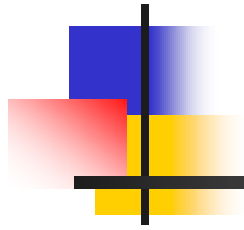
- n Google、Baidu
- n 理论 + 实践（开发板）
- n 勤于思考，善于总结
- n 多上相关技术论坛，他山之石可以攻玉
- n 良好的文档撰写习惯
- n Passion！



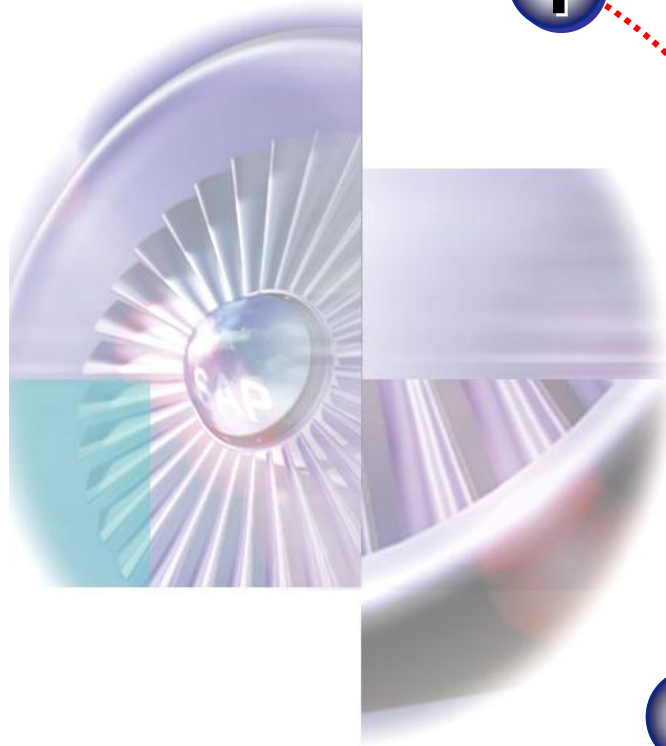


CHAPTER

7 Linux的高级应用 编程



主要内容



1

多进程编程

2

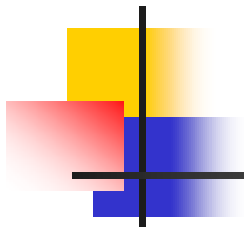
多线程编程

3

网络编程

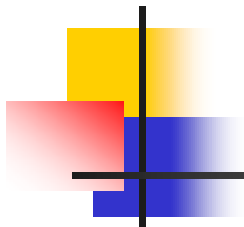
4

应用程序的调试技巧



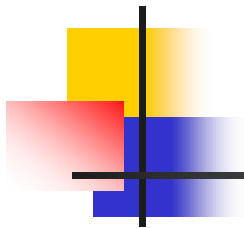
多进程编程

- n 进程组成
- n 进程控制
- n 进程间的通信机制



进程组成

- n 何谓进程？
- n 进程和程序的区别
- n 进程的特性



何谓进程？

n 定义

n 由代码段（text）、用户数据段（user segment）以及系统数据段（system segment）共同组成的一个动态执行环境

n 代码段可共享

n 用户数据段存放全局变量

n 系统数据段为进程的管理控制信息

进程和程序的区别

n 程序

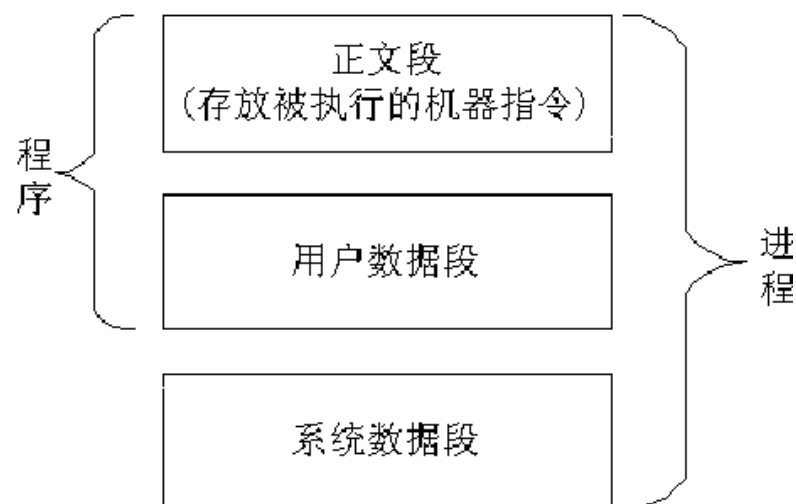
- n 静态对象，纯粹的数据。
编译后形成的可执行代码

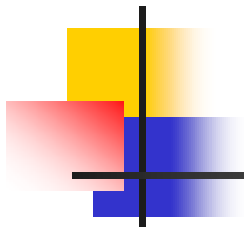
- n 由代码段、数据段、**BSS**段等组成，是进程的一部分

n 进程

- n 程序的动态实例

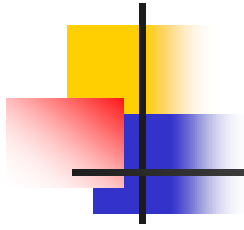
- n 同一个程序可以有多个
动态实例，多份拷贝

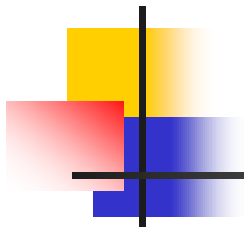




进程的特性

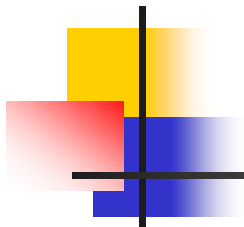
- n 进程的状态
 - n 根据OS的调度处在不同的状态
- n 进程的地址空间
 - n 用户空间
 - n 内核空间
 - n 独立栈





进程控制

- n 父子进程
- n fork
- n Exec
- n system
- n 孤儿进程
- n 僵尸进程



父子进程

- n 亲缘关系

- n 描述了系统进程创建的关系

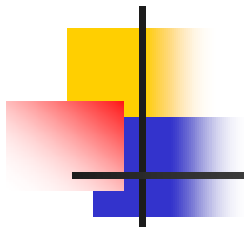
- n 祖先进程

- n Linuxrc启动脚本

- n /sbin/init

- n 1号进程

- n ps -ef



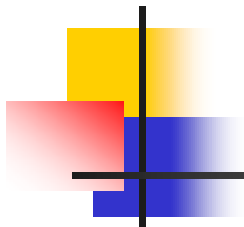
fork

n 功能

- n 创建一个子进程
- n Fork的返回值是父子进程的分岔点
- n 对于父进程，返回值为子进程的pid
- n 子进程，返回值0

n Fork产生的父子进程的关系

- n 共享代码段
- n 拷贝数据段和堆栈段
- n 子进程继承父进程的所有资源
- n 只有在数据段发生变化时才从物理上进行分离



exec

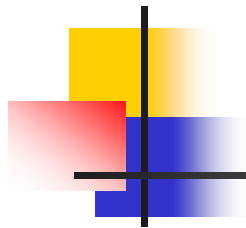
n 功能

- n 改变当前进程的行为，替换为另一个进程

n exec产生的进程与原进程的关系

- n 进程id不变，只留下躯壳

- n 更新代码段，创建新的数据段和堆栈



system

n 功能

- n 在父进程里面启动一个新的命令，并等待该命令执行完成

n 流程

- n 函数先调用fork()
- n 然后再调用exec()来执行用户的登录shell，通过它来查找可执行文件的命令并分析参数
- n wait()函数族之一来等待子进程的结束



孤儿进程

n 定义

n 父进程先于子进程消亡，子进程即变为孤儿进程

n 解决办法

n 对系统无影响

n 孤儿进程将被init进程(进程号为1)所收养



僵尸进程

n 定义

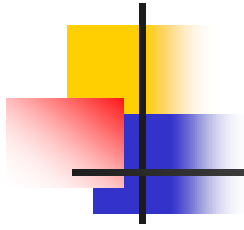
- n 子进程消亡时，父进程未回收task_struct资源，只剩下空壳，故为僵尸

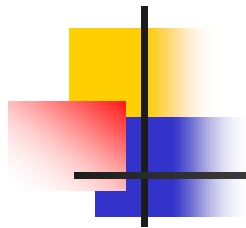
n 原因

- n 子进程结束后向父进程发出SIGCHLD信号，父进程忽略了它
- n 父进程没有调用wait()或waitpid()函数来等待子进程的结束

n 解决方案

- n 父进程可以通过wait系列的系统调用（如wait4、waitid）来等待某个或某些子进程的退出，并获取它的退出信息
- n 杀死父进程可以让init进程回收僵尸进程的资源





进程间的通信机制

- n 进程间的约束关系
- n 通信机制的来源
- n 通信机制的类型



进程间的约束关系

n 同步

- n 指系统中一些进程需要相互合作，共同完成一项任务

n 互斥

- n 由于各进程要求共享资源，但某一资源同时只允许一个访问者对其进行访问，具有唯一性和排它性

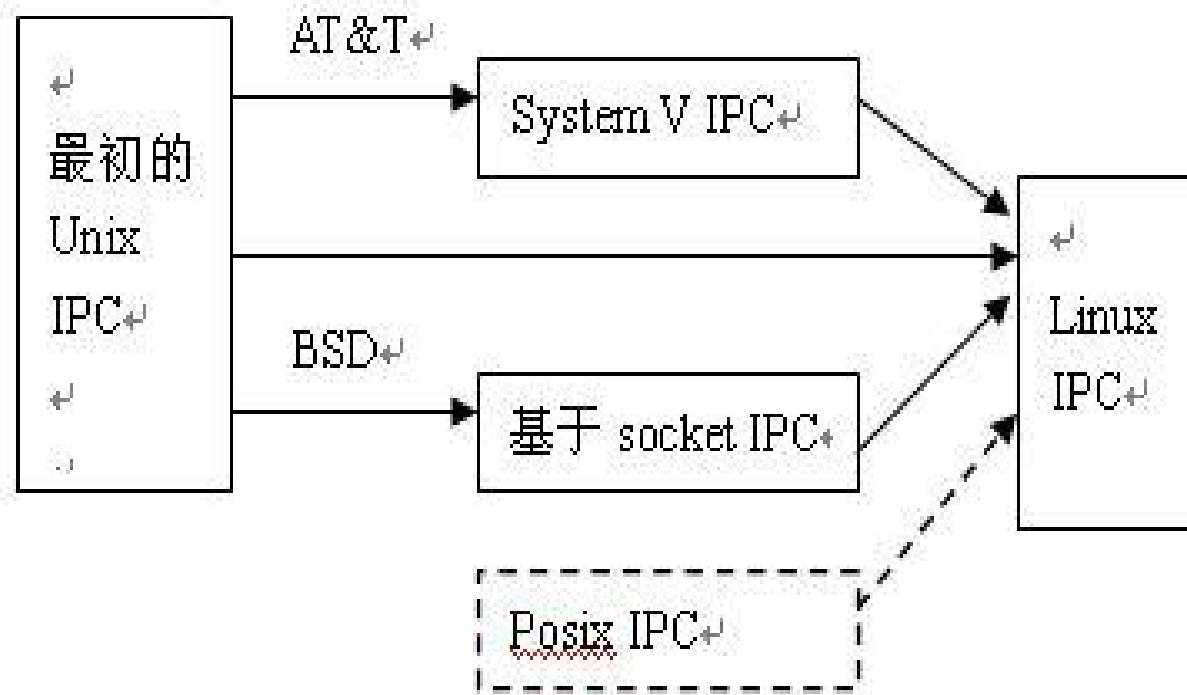
n 临界区

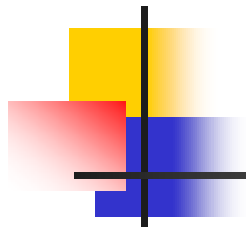
- n 访问共享资源的代码

n 死锁

- n 指多个进程互不相让，都得不到足够的资源
- n 各进程按照相同的顺序获取互斥资源可以防止死锁

通信机制的来源





通信机制的来源

- n Unix IPC

- n 管道、FIFO、信号

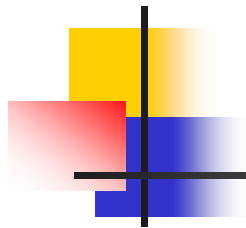
- n System V IPC

- n 消息队列、信号灯、共享内存

- n Posix IPC

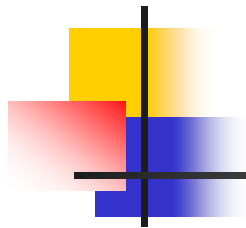
- n 消息队列、信号灯、共享内存

- n socket



通信机制的类型

- n 管道pipe
- n 管道FIFO
- n 共享内存
- n 消息队列
- n 信号灯
- n 信号
- n socket



管道pipe

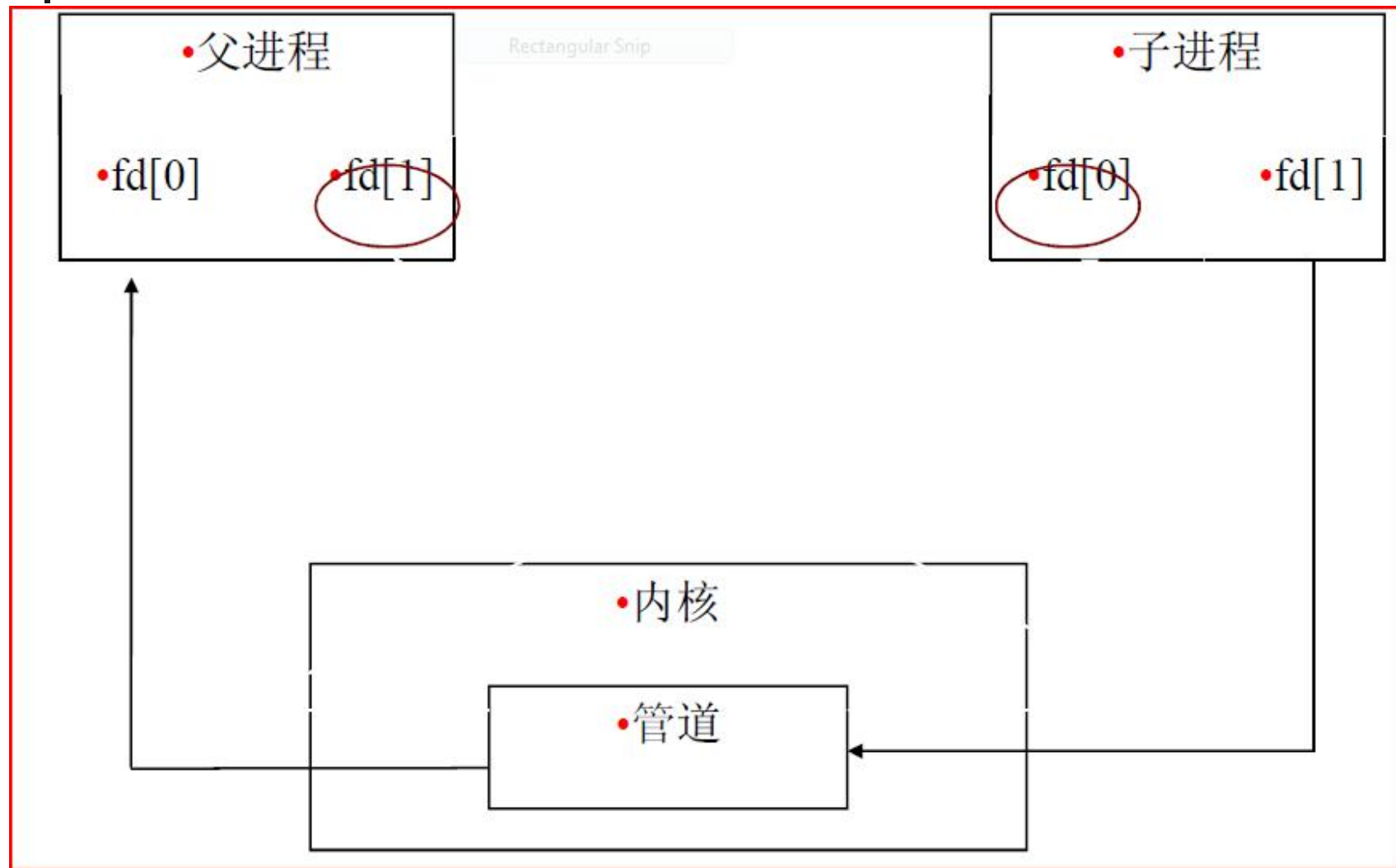
n 特点

- n 是一种单向的数据流，一个进程写入数据另外一个进程读取数据，典型的生产者消费者模型
- n 基于文件系统，但是其仅存在于内存中，无文件实体
- n 在最后一个访问管道的进程终止后，管道就被自动删除

n 适用场合

- n 具有亲缘关系的进程
- n 单处理器

管道pipe





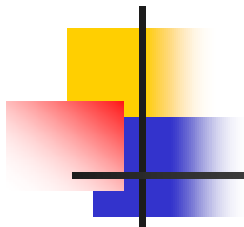
管道FIFO

n 特点

- n 类似普通文件，存在文件实体
- n 不支持lseek文件定位操作
- n Mkfifo命令或者函数建立
- n 可以使用路径来指示FIFO
- n 严格的先进先出，从开始处返回数据，写入时总是将数据添加到末尾

n 适用场合

- n 任意进程间的通信
- n 单处理器



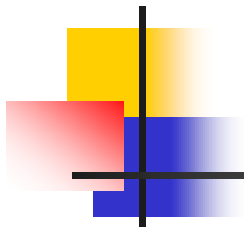
共享内存

n 特点

- n 内核的专用内存区
- n 需要访问的进程将其映射到私有地址空间
- n 多进程直接读取内存，无需拷贝数据
- n 因为共享，所以需要互斥，通常和信号灯配合使用
- n 可以通过使用shmctl函数设置共享存储内存的某些标志位如SHM_LOCK、SHM_UNLOCK等来实现

n 适用场合

- n 任意进程间的快速通信
- n 单处理器



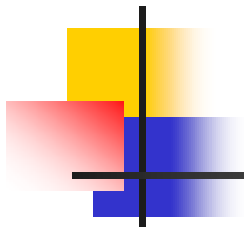
消息队列

n 特点

- n 一个消息列表，用户可以添加及读取消息
- n 类似FIFO，但可以根据msg类型实现随机读取
- n 可根据消息类型实现不同的功能
- n 操作类型包括创建、发送、接收及控制
- n 消息存在于内核中，由队列id标识

n 适用场合

- n 任意进程间的通信
- n 单处理器



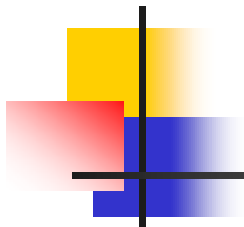
信号灯

n 特点

- n 实现资源的互斥访问
- n 无法携带其他更多的辅助信息

n 适用场合

- n 任意进程间的互斥通信
- n 单处理器



信号

n 特点

- n 是一种异步机制，是软件层次上对中断的一种模拟
- n 每一个进程可以自定义对信号的处理流程，甚至改变信号的行为
- n 忽略、捕捉、默认行为
- n **Sigkill**和**sigstop**不能忽略

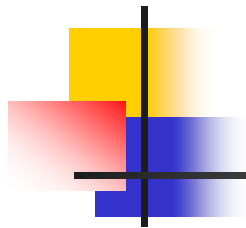
n 适用场合

- n 任意进程间的通信，通常用于内核向进程发送消息
- n 单处理器



信号

信号名	含义	默认操作
SIGHUP	该信号在用户终端连接(正常或非正常)结束时发出,通常是在终端的控制进程结束时,通知同一会话内的各个作业与控制终端不再关联。	终止
SIGINT	该信号在用户键入INTR字符(通常是Ctrl-C)时发出,终端驱动程序发送此信号并送到前台进程中的每一个进程。	终止
SIGQUIT	该信号和SIGINT类似,但由QUIT字符(通常是Ctrl-\)来控制。	终止
SIGILL	该信号在一个进程企图执行一条非法指令时(可执行文件本身出现错误,或者试图执行数据段、堆栈溢出时)发出。	终止
SIGFPE	该信号在发生致命的算术运算错误时发出。这里不仅包括浮点运算错误,还包括溢出及除数为0等其它所有的算术的错误。	终止
SIGKILL	该信号用来立即结束程序的运行,并且不能被阻塞、处理和忽略。	终止
SIGALRM	该信号当一个定时器到时的时候发出。	终止
SIGSTOP	该信号用于暂停一个进程,且不能被阻塞、处理或忽略。	暂停进程
SIGTSTP	该信号用于交互停止进程,用户可键入SUSP字符时(通常是Ctrl-Z)发出这个信号。	停止进程
SIGCHLD	子进程改变状态时,父进程会收到这个信号	忽略
SIGABORT		



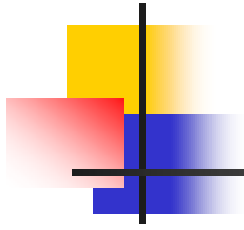
socket

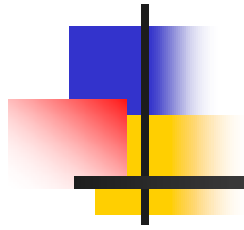
n 特点

- n 基于套接字的编程，利用TCP/IP协议栈
- n 可以实现任意类型的通信，消息可以自由定义

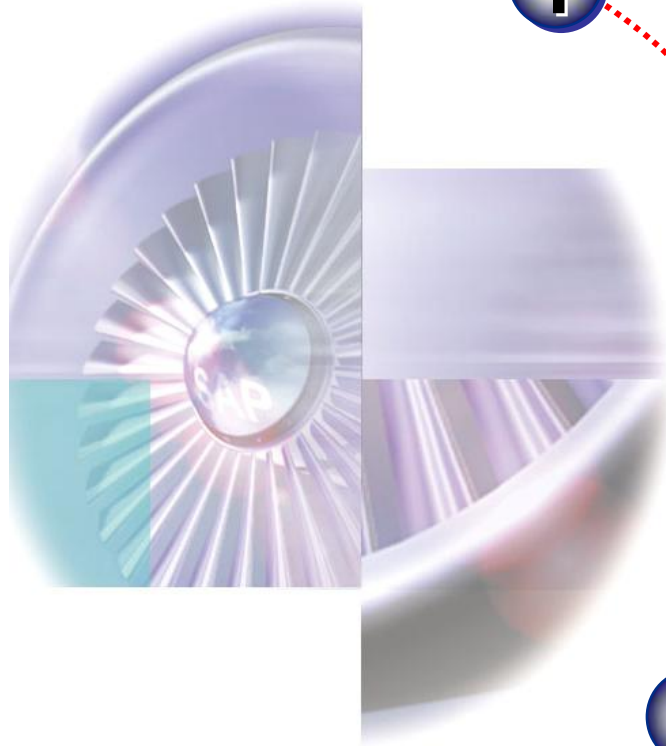
n 适用场合

- n 任意进程间的通信
- n 单处理器及多机





主要内容



1

多进程编程

2

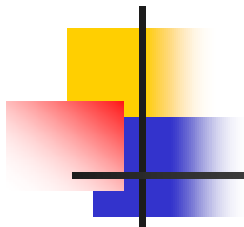
多线程编程

3

网络编程

4

应用程序的调试技巧



多线程编程

- n 线程的特点
- n Linux的线程模型
- n 线程的通信机制



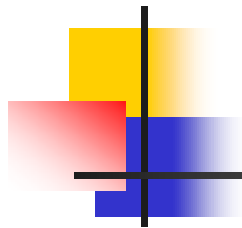
进程和用户线程的区别

n 线程

- n 进程的子集，由线程库调度，程序执行的最小单位
- n 多线程模型设计使程序更简洁明了
- n 更好的支持SMP以及减小上下文切换开销

n 进程

- n 资源分配和管理的最小单位
- n 具备独立的地址空间
- n 创建销毁慢
- n 内核调度的基本单位
- n 上下文切换开销大



进程和用户及内核线程的区别

n 进程

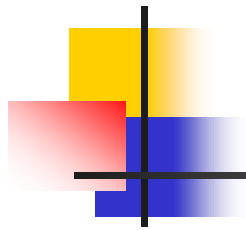
- n 内核调度的基本单位
- n 包括用户空间和内核空间两部分

n 用户线程

- n 线程库调度，只有用户空间

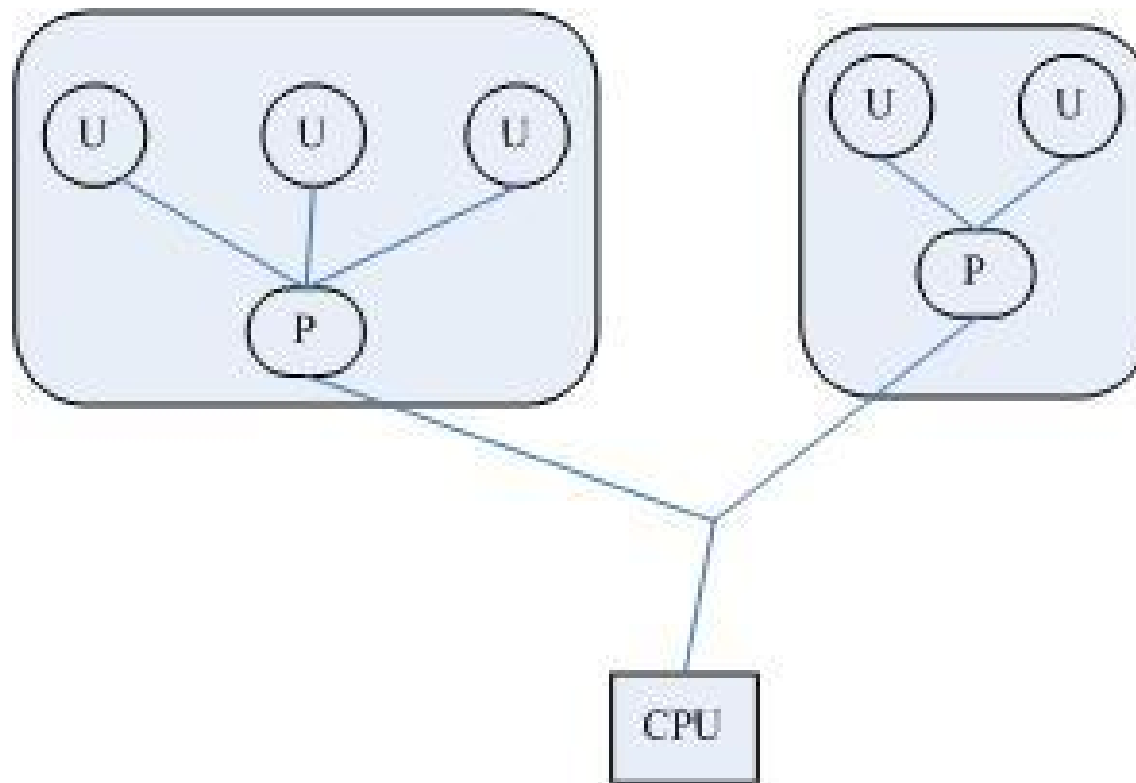
n 内核线程

- n 内核调度，只有内核空间
- n 其他和进程一样



Linux的线程模型

n 一对多模型





线程间的通信机制

n Thread-specific Data

- n 仅供本线程使用的全局变量

n 互斥锁

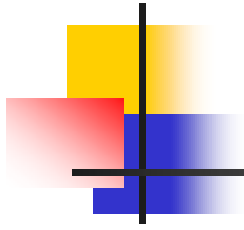
- n pthread_mutex_t
- n 实现线程间的互斥访问

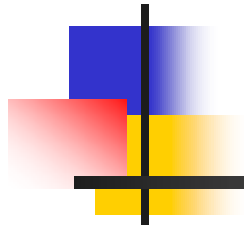
n 条件变量

- n pthread_cond_t
- n 利用共享的全局变量进行同步
- n 需要和线程锁配合使用以防止竞态

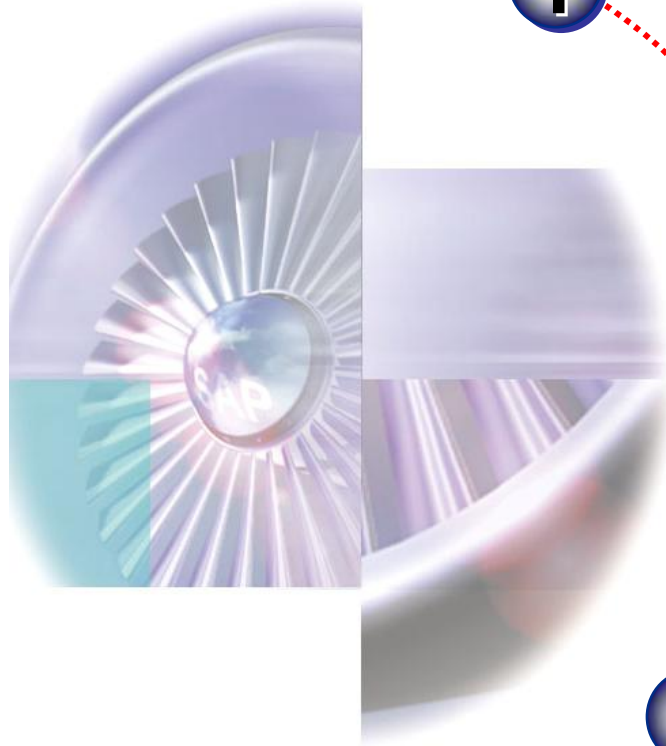
n 信号灯

- n 可以有多个值，主要用于同步操作
- n 即使没有人等待灯，释放灯也是有意义的





主要内容



1

多进程编程

2

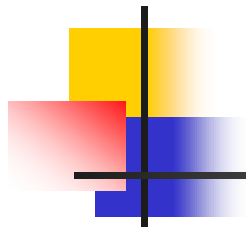
多线程编程

3

网络编程

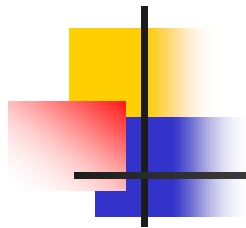
4

应用程序的调试技巧



网络编程

- n 网络编程的基本流程
- n 并发服务器的设计模型



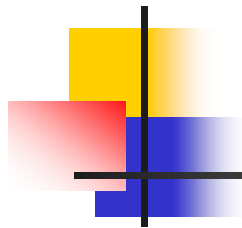
网络编程的基本流程

n TCP

n 演示

n UDP

n 演示



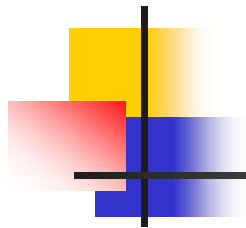
并发服务器的设计模型

n 多进程

- n 每个进程服务一个客户端。优势是有各自独立的地址空间，可靠性高，但进程调度开销大，无法资源共享，进程间通信机制复杂

n 多线程

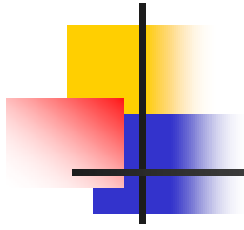
- n 每个线程服务一个客户端。优势是开销小，通信机制简单，可共享内存。但共享地址空间，可靠性低，一个服务器出现问题时可能导致系统崩溃，同时全局共享可能带来竞争，共享资源需要互斥，对编程要求高

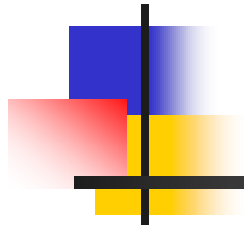


并发服务器的设计模型

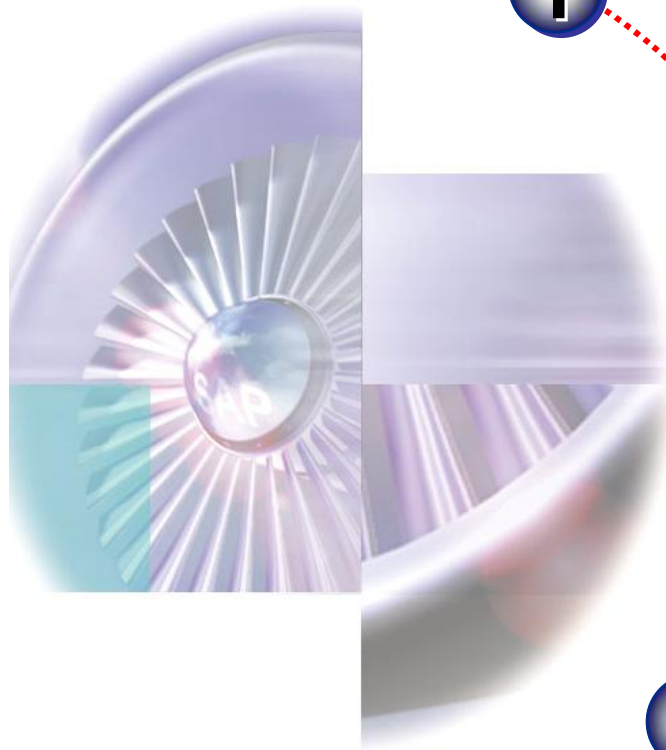
n 单进程select

- n 占有的进程及线程资源少，通信机制简单。但监听服务器及各个子服务器揉和在一起，程序结构复杂不清晰，编程麻烦





主要内容



1

多进程编程

2

多线程编程

3

网络编程

4

应用程序的调试技巧



应用程序的调试技巧

- n 错误打印
- n 内存泄露检查
- n Core dump
- n 自动调用GDB
- n 自动显示调用栈



错误打印

n Abort

- n 异常终止程序，不做清除工作

n Exit

- n 正常终结目前进程的执行，并把参数 `status` 返回给父进程

n Atexit

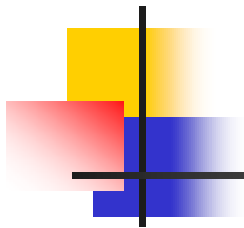
- n 注册程序正常终止时要被调用的函数

n Strerror

- n 返回错误原因的描述字符串

n perror

- n 返回错误原因的描述字符串，并把它输出到标准错误输出流



错误打印

n __FILE__

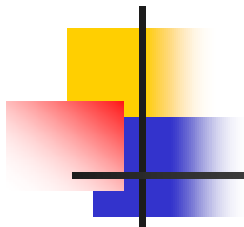
n Log对应的源文件名

n __LINE__

n Log对应的行号

n __FUNCTION__

n Log对应的函数名



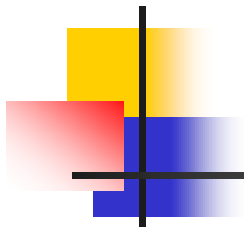
内存泄露检查

n 功能

- n 检查是否有不对称的动态内存申请释放以及指针越界操作

n tools

- n Mtrace
- n memwatch



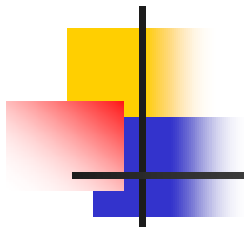
Core dump

n 功能

- n 当出现非法地址访问时会出现segmentation fault，使能core dump后会自动记录程序崩溃时的内存现场至log文件
- n 可以利用gdb工具读取该log文件获得程序崩溃的位置及原因

n tools

- n `sysctl -w "kernel.core_pattern=/xx/%e-core.%p"` 设置产生的core dump文件的路径及命名规则
- n `gdb EXECUTABLE_FILE CORE_FILE` 分析core dump文件
- n `bt` 打印出错的栈调用信息



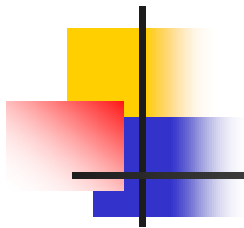
自动调用GDB

n 功能

- n 当出现非法地址访问时会出现segmentation fault，内核会向对应的进程发送SIGSEGV，其默认行为是打印segmentation fault
- n 可以改变SIGSEGV信号的默认行为，在处理函数里自动调用gdb启动调试

n tools

- n Getpid获得当前运行进程的pid号
- n 根据/proc/pidxxx/cmdline获得gdb启动时的参数
- n 获得SIGSEGV信号自动调用“gdb axxx”
- n bt 打印出错的栈调用信息



自动显示调用栈

n 功能

- n 嵌入式平台上一般没有gdb等调试工具，另外单步调试对于多进程多线程程序的调试不方便
- n 当出现非法地址访问时会出现segmentation fault，可以改变SIGSEGV信号的默认行为，在处理函数里自动打印栈信息，模拟bt命令

n tools

- n Backtrace获得当前函数调用的级别
- n backtrace_symbols获得部分符号化的调用栈
- n Objdump反汇编程序，分析栈中返回地址对应的上一条指令地址

