

The logo features the word "FAR SIGHT" in white, bold, sans-serif capital letters. A red, stylized vertical bar is positioned between the "R" and "S". The text is centered within a dark green, textured, downward-pointing triangle that has a 3D effect with a lighter green top edge and a darker green bottom edge. The triangle is set against a light gray background that forms a larger, inverted triangle shape.

FAR SIGHT

嵌入式培训专家

应用最广泛的嵌入式实时操作系统
——VxWorks介绍

www.farsight.com.cn

- ✓ VxWorks操作系统介绍
- ✓ 为实时性需求而设计的操作系统
 - ∅ 多任务和任务间通信
 - ∅ 中断处理
 - ∅ 内存管理
 - ∅ IO系统
- ✓ 走进VxWorks BSP 开发
- ✓ VxWorks最新技术
 - ∅ 像Windows中一样使用进程和动态连接库 (VxWorks RTP)
 - ∅ 对多核的支持 (VxWorks SMP)
- ✓ 课程培训目标

The logo features the words "FAR SIGHT" in a white, serif font. A red, stylized vertical bar separates the two words. The text is centered within a dark green, textured, downward-pointing triangle that has a 3D effect with a lighter green highlight on its top edge.

FAR SIGHT

嵌入式培训专家

VxWorks操作系统介绍

华清远见

- √ 网络设备
- √ 工业自动化
- √ 汽车
- √ 航空航天
- √ 国防工业
- √ 消费电子



VxWorks 的应用



FAR SIGHT

VxWorks的特点

- √ 实时性
- √ 稳定性
- √ 可裁减性
- √ 友好的开发调试环境
- √ 广泛的运行环境支持

实时性

√ 硬实时

- ∅ 有一个刚性的、不可改变的时间限制
- ∅ 它不允许任何超出时限的错误

√ 软实时

- ∅ 时限是一个柔性灵活的，可以容忍偶然的超时错误
- ∅ 只能提供统计意义上的实时

√ 非实时

- ∅ 对时间没有什么特定的要求

非实时

软实时

硬实时



计算机仿真

用户界面

网络视频

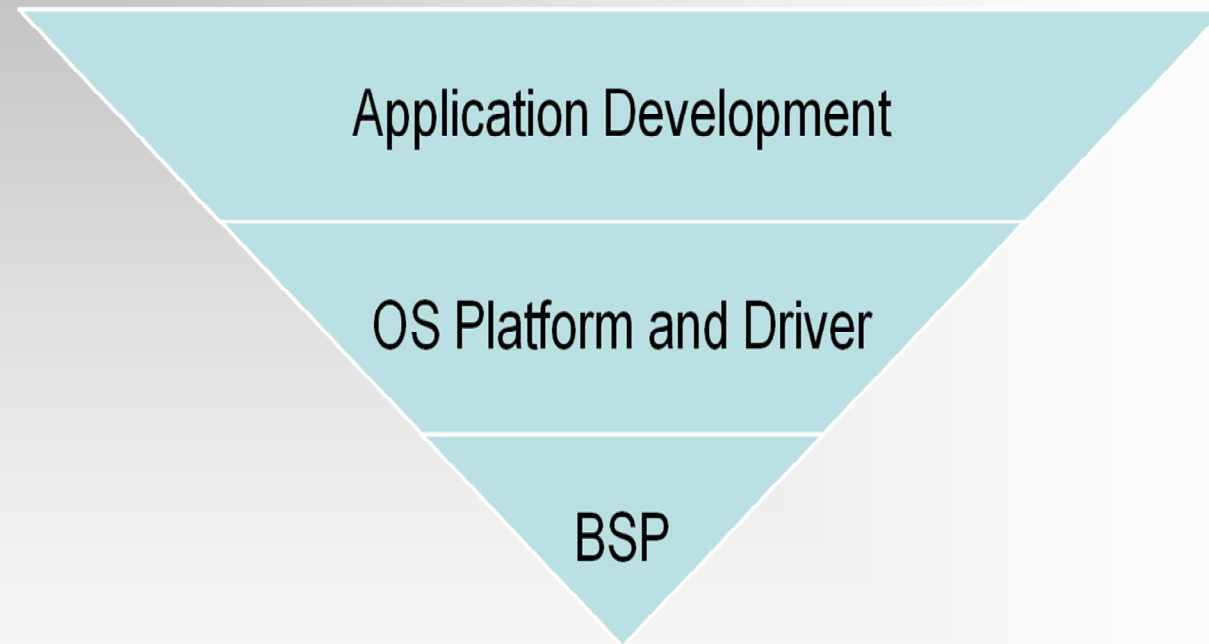
Cruise 控制

电信

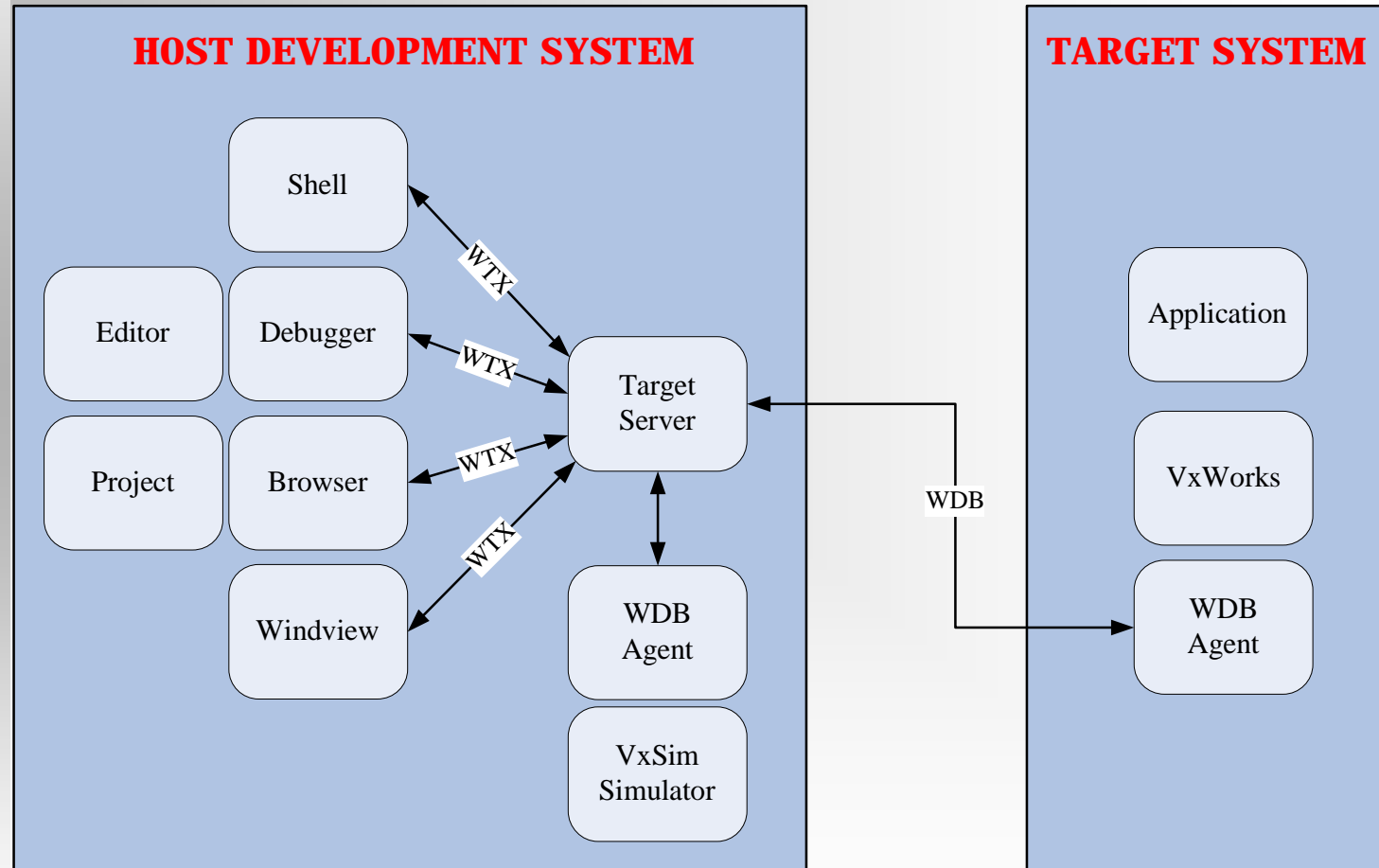
飞行控制

电子引擎

对开发人员的要求



WorkBench 开发环境



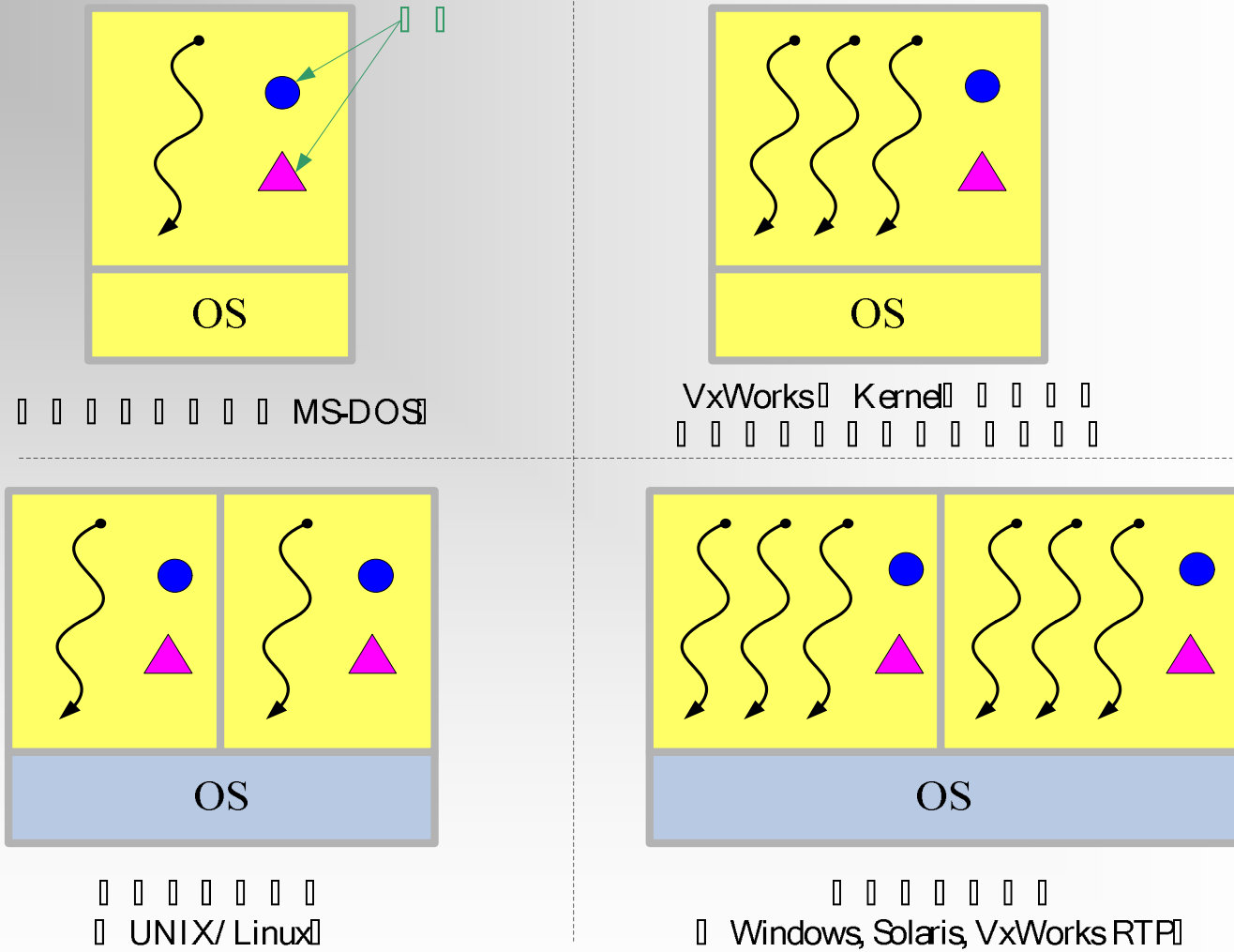
The logo features the words "FAR SIGHT" in a white, serif font. A red, stylized vertical line separates the two words. The text is centered within a dark green, textured, downward-pointing triangle that has a 3D effect with a lighter green top edge. The triangle is set against a light gray background that forms a larger, inverted triangle shape.

FAR SIGHT

嵌入式培训专家

为实时性需求而设计的操作系统

进程和线程



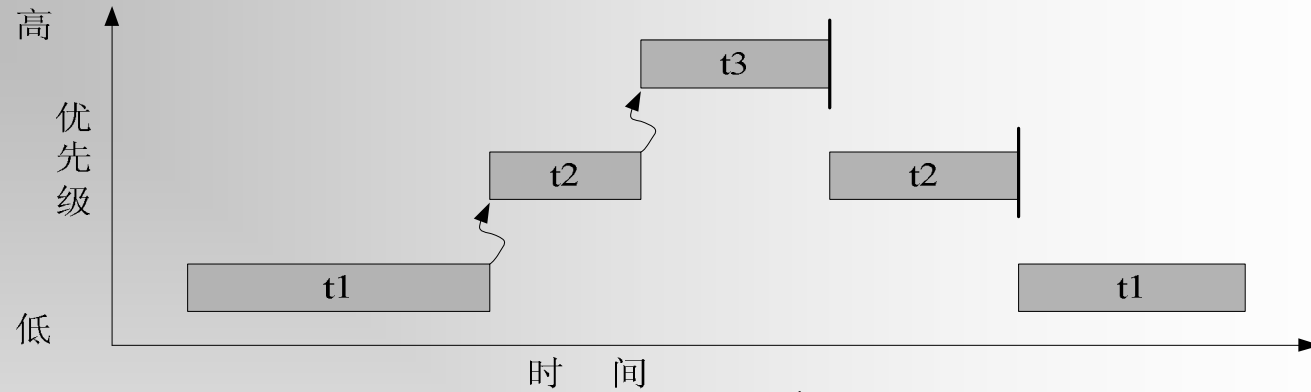
✓ 进行任务管理


∅ 任务是竞争系统资源的最小运行单元。任务可以使用或等待CPU、I/O设备及内存空间等系统资源，并独立于其它任务，与它们一起并发运行（宏观上如此）。VxWorks内核使任务能快速共享系统的绝大部分资源，同时有独立的上下文来控制个别线程的执行；

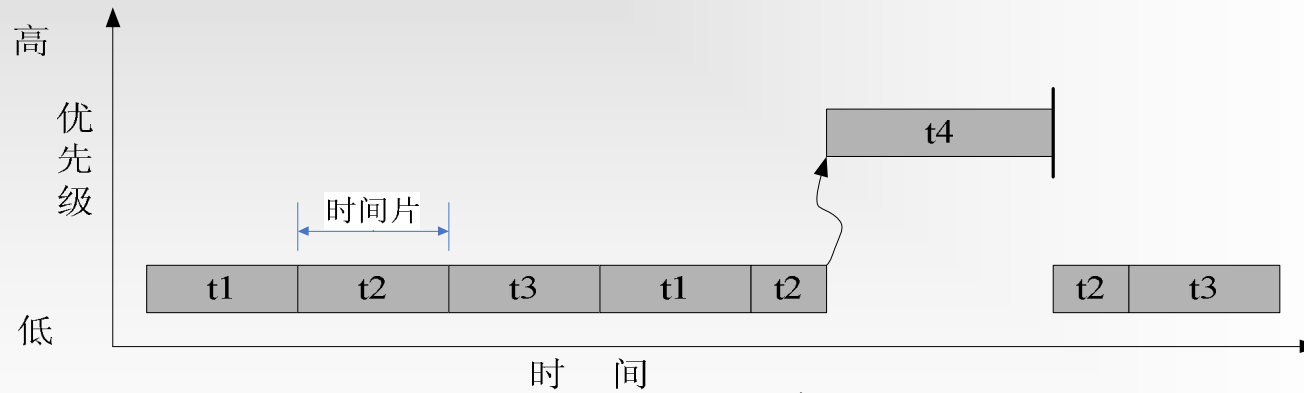
✓ VxWorks实时内核Wind提供了基本的多任务环境，系统内核根据某一调度策略让它们交替运行。

✓ 系统调度器使用**任务控制块**的数据结构（简记为TCB）来管理任务调度功能。

基于优先级的抢占式调度



注:  表示抢占 |  表示任务完成



注:  表示抢占 |  表示任务完成

√ 任务间通信手段

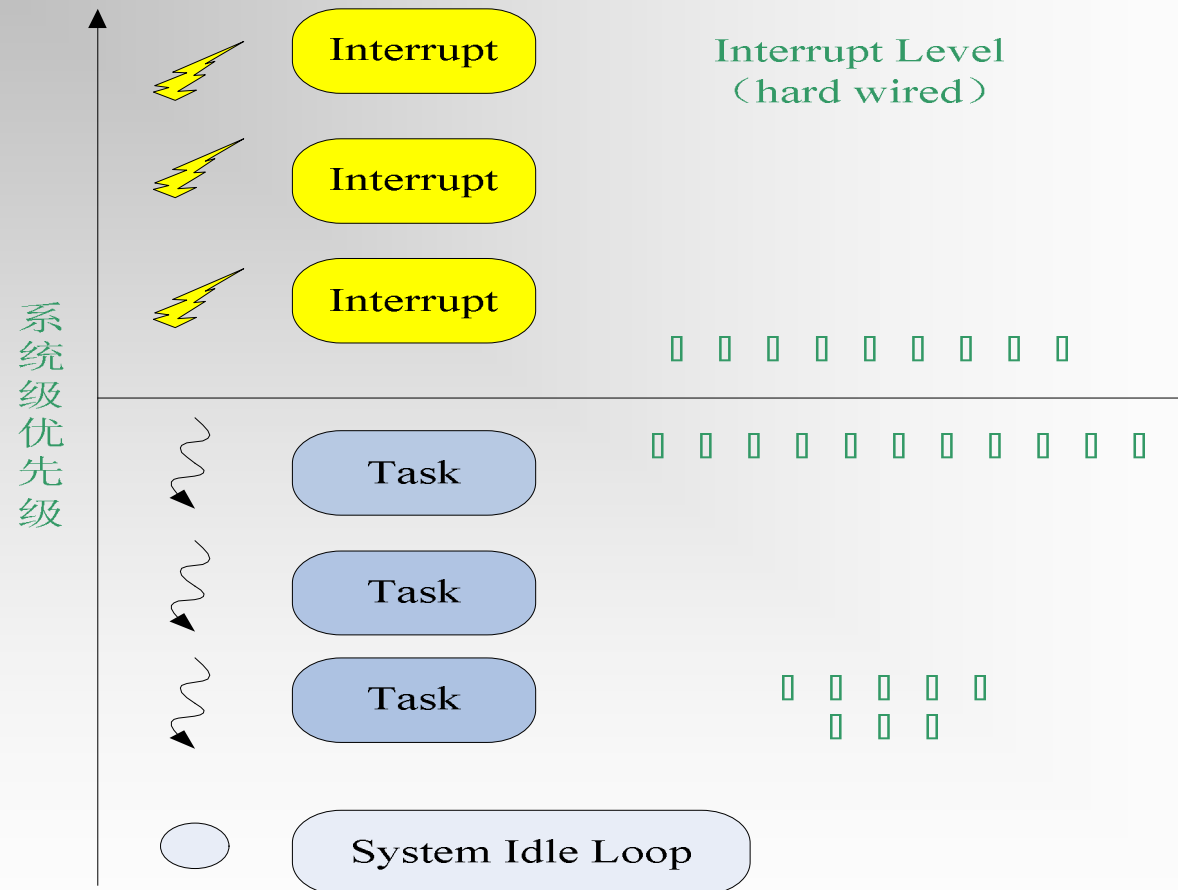
- ∅ 共享数据结构、共享内存；
- ∅ 信号量；
- ∅ 消息队列；
- ∅ 管道；
- ∅ 信号
- ∅ 事件与Task网络通信

VxWorks 中断处理

- ✓ 硬件中断处理是实时系统设计的最重要、最关键的问题。
- ✓ 为了获得尽可能快的中断响应时间，VxWorks的中断处理程序运行在特定的上下文中(在所有任务上下文之外)。因此，中断处理不会涉及任何任务上下文的交换。
- ✓ ISR拥有专用堆栈，在系统初始化时分配，大小由INT_STACK_SIZE决定。
- ✓ 通过内核工作队列(Kernel Work Queue)，最大程度较小中断处理延时。

中断和任务的优先级

√ 中断抢占最高优先级的任务

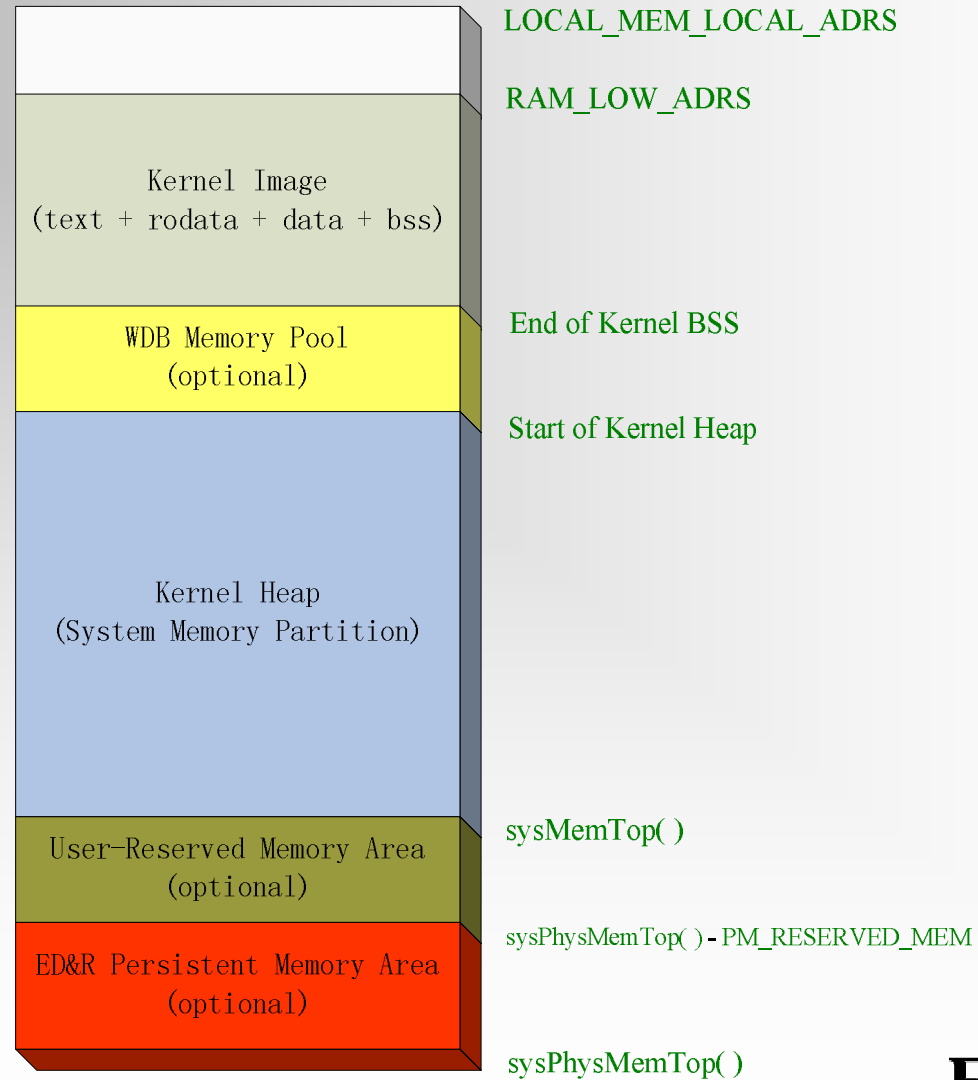


intLock and taskLock

- ✓ 谨慎使用
- ✓ taskLock增加任务的抢占延时
- ✓ intLock增加中断延时
- ✓ intLock在锁中断的同时会锁任务切换
- ✓ 尽量避免使用intLock

```
lockLevel = intLock();  
for(i=0;i<10000000;i++)  
    gDataCount++;  
  
taskDelay(10);  
intUnlock(lockLevel);
```

VxWorks的内存空间布局



- ✓ memPartLib 和 memLib
- ✓ malloc () 和 free();
- ✓ 其他的内存分配相关函数：
 - ∅ *void * calloc (nElems, size)* Allocate zeroed memory for an array. ;
 - ∅ *void * realloc (ptr, newSize)* Resize an allocated block. The block may be moved ;
 - ∅ *int memFindMax()* Returns the size of the largest free block in system memory ;

✓ 采用页表的方式管理内存 (vmBaseLib)

✓ 主要功能包括：

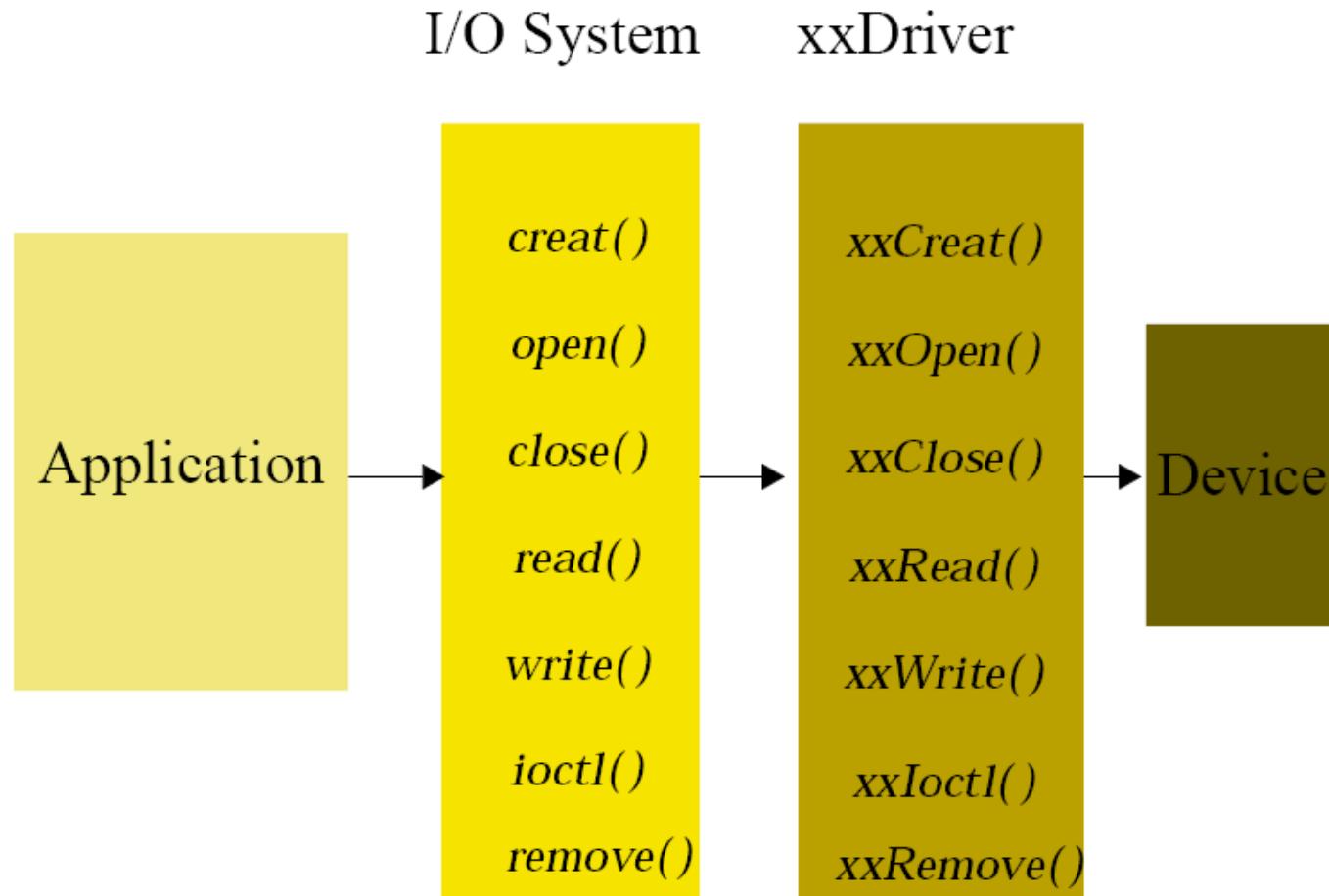
- ∅ 启动时设置内核模式内存空间的上下文
- ∅ 映射物理地址到虚拟地址
- ∅ 设置内存的Cache属性
- ∅ 设置内存的保护属性
- ∅ 内存映射的Enable/Disable
- ∅ TLB的Lock/Unlock
- ∅ 页大小的优化

✓ RTP的支持

- ∅ 进程内存空间上下文的管理

VxWorks IO系统

- ✓ 专门为实时系统设计，简单、灵活、高效
- ✓ 提供了对标准C库中basic和buffered I/O的支持
- ✓ VxWorks IO系统在把控制传递给设备驱动之前，只做最简单的基本处理，基本上只是把用户IO请求路由到正确的设备驱动程序入口。
- ✓ 驱动实现相当灵活，甚至可以完全绕开IO系统
- ✓ 驱动程序可以被动态的安装和卸载
- ✓ 驱动程序执行在执行该IO操作的任务的上下文中，可以被抢占。



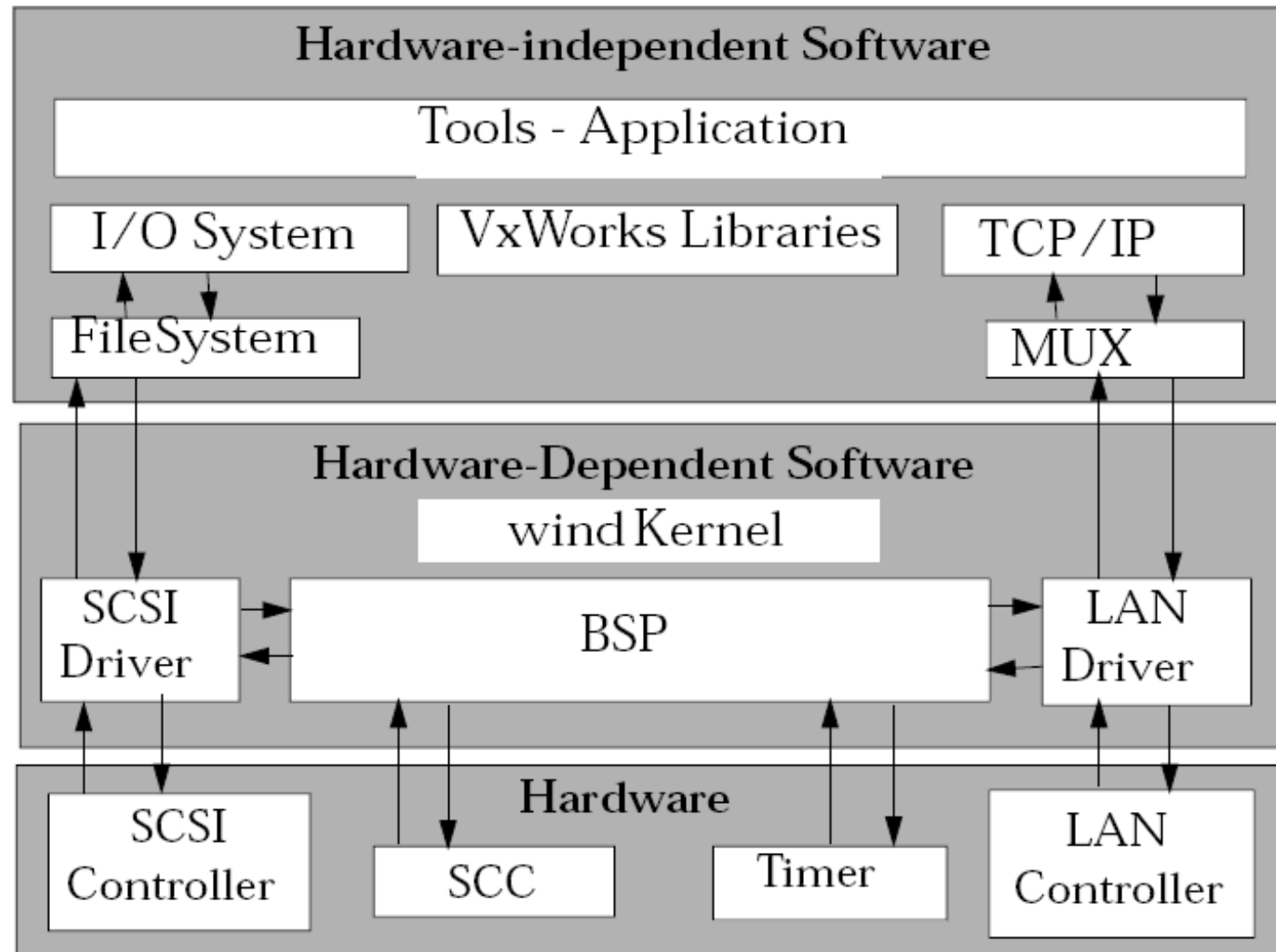
The logo features the words "FAR SIGHT" in a white, serif font. A red, stylized vertical line separates the two words. The text is centered within a dark green, textured, downward-pointing triangle that has a 3D effect with a lighter green top edge.

FAR SIGHT

嵌入式培训专家

走进VxWorks BSP 开发

- ✓ 屏蔽硬件单板差异，给VxWorks提供访问硬件的接口。
- ✓ BSP是一系列文件（函数）的集合，有些用来在操作系统内核启动前执行硬件初始化，有些被操作系统调用
- ✓ 和Bootrom概念的区别
 - ∅ Bootrom相当于PC的BIOS, Linux下的Bootloader.
 - ∅ 从生存周期上看，Bootrom完成引导后不再存在，BSP则一直生存。
 - ∅ 在VxWorks下，Bootrom是BSP的一个应用



√ 上电或复位后的初始化

- ∅ CPU初始化
- ∅ 内存控制器初始化，等等

√ 一般硬件的初始化和配置

- ∅ 串口、网口、硬盘等等
- ∅ 通过调用Device driver的接口函数完成，也需要BSP执行参数配置

√ BSP特定硬件的支持

- ∅ 中断控制器
- ∅ 时钟和定时器

√ 提供操作系统的配置参数

- ∅ 内存参数
- ∅ 总线配置 (PCI/VME)

三种VxWorks映像

√ Loadable

∅ 通过Bootrom进行引导.

√ ROM-based

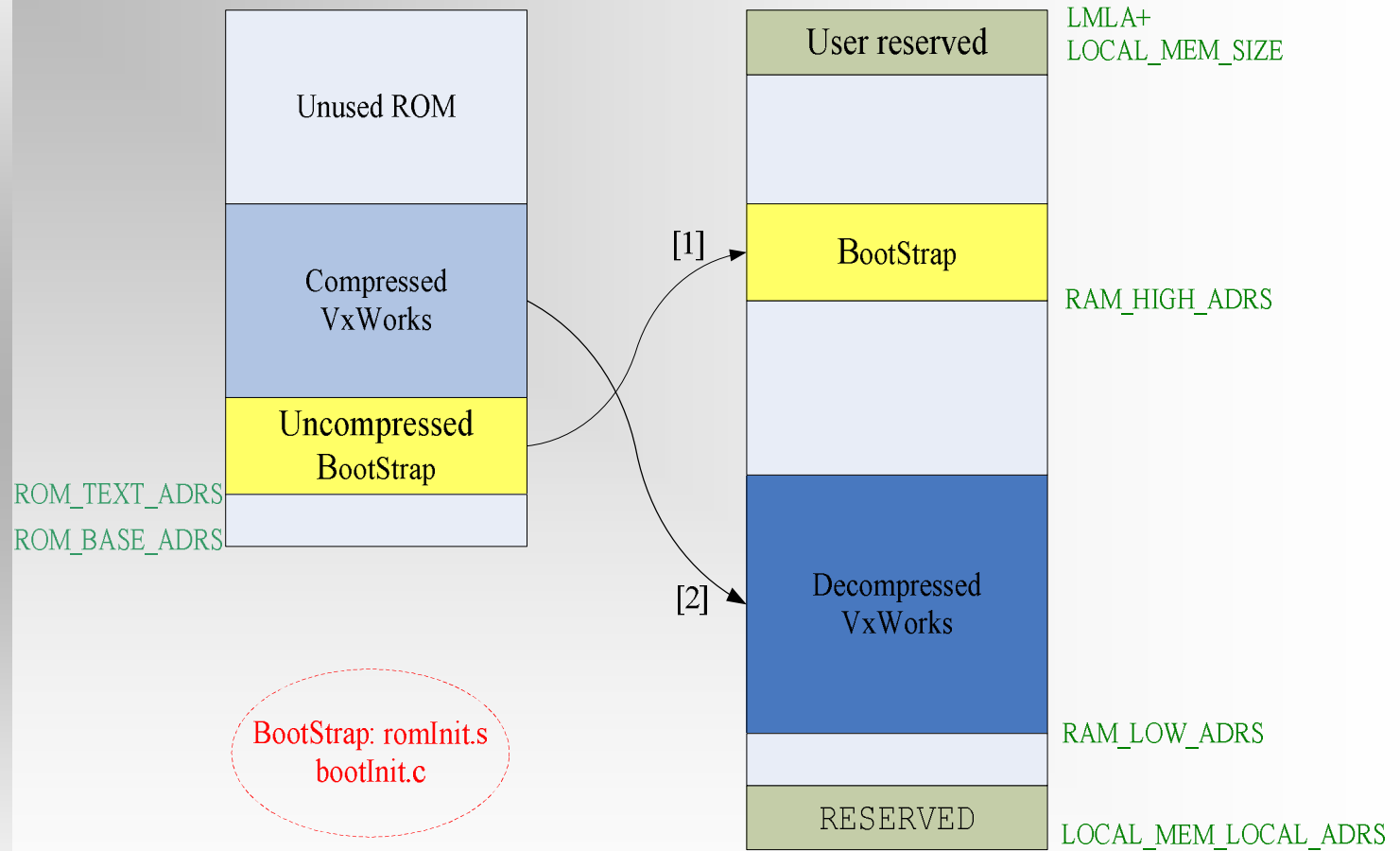
∅ 等于Bootrom和VxWorks的组合.

∅ 分为压缩和非压缩两种

√ ROM-Resident

∅ 相比ROM-based, 只是把数据段load进RAM中.

VxWorks romCompress启动过程



BSP开发流程

- ✓ 建立开发环境，得到参考BSP。
- ✓ 准备内核启动之前的初始化代码。
- ✓ 启动VxWorks内核，提供系统时钟和中断向量的安装。
- ✓ 启用WDB，使用WorkBench进行调试。
- ✓ 完成具体的硬件支持。

The logo features the words "FAR SIGHT" in a white, serif font. A red, stylized vertical bar separates the two words. The text is centered within a dark green, textured, downward-pointing triangle that has a 3D effect with a lighter green top edge.

FAR SIGHT

嵌入式培训专家

VxWorks最新技术

像Windows中一样使用进程和动态连接库
(VxWorks RTP)

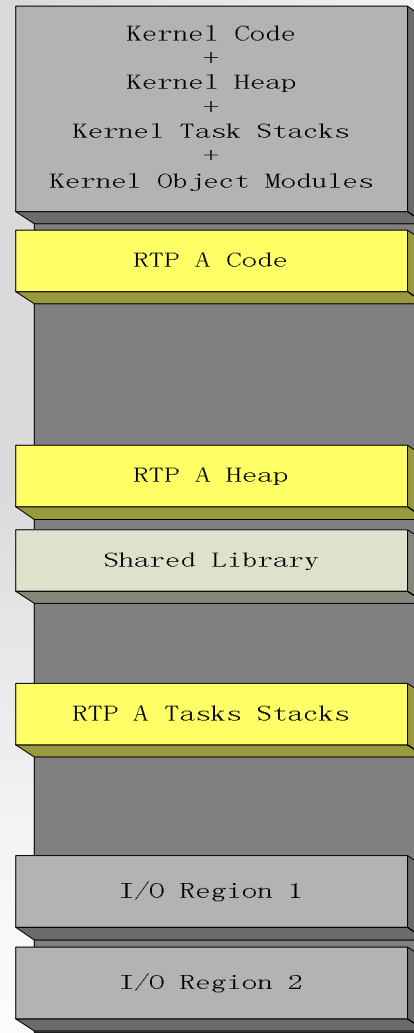
什么是RTP?

- ✓ Real Time Process
- ✓ 与其他操作系统进程很相像，为实时嵌入式系统专门优化
- ✓ 用户模式执行
- ✓ 每个进程有自己独立的地址空间
- ✓ 应用程序和操作系统独立编译
- ✓ 每个进程的命名空间相对独立

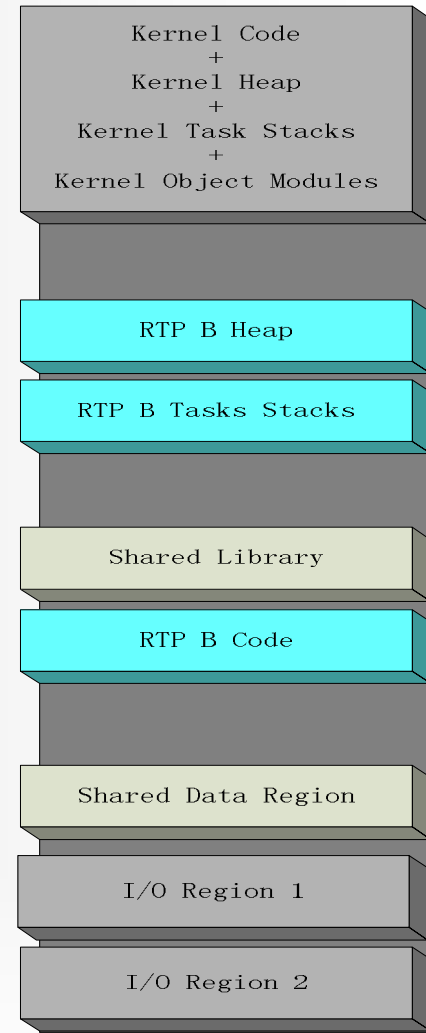
RTP模型下的内存映射




Kernel Memory View



RTP A Memory View



RTP B Memory View

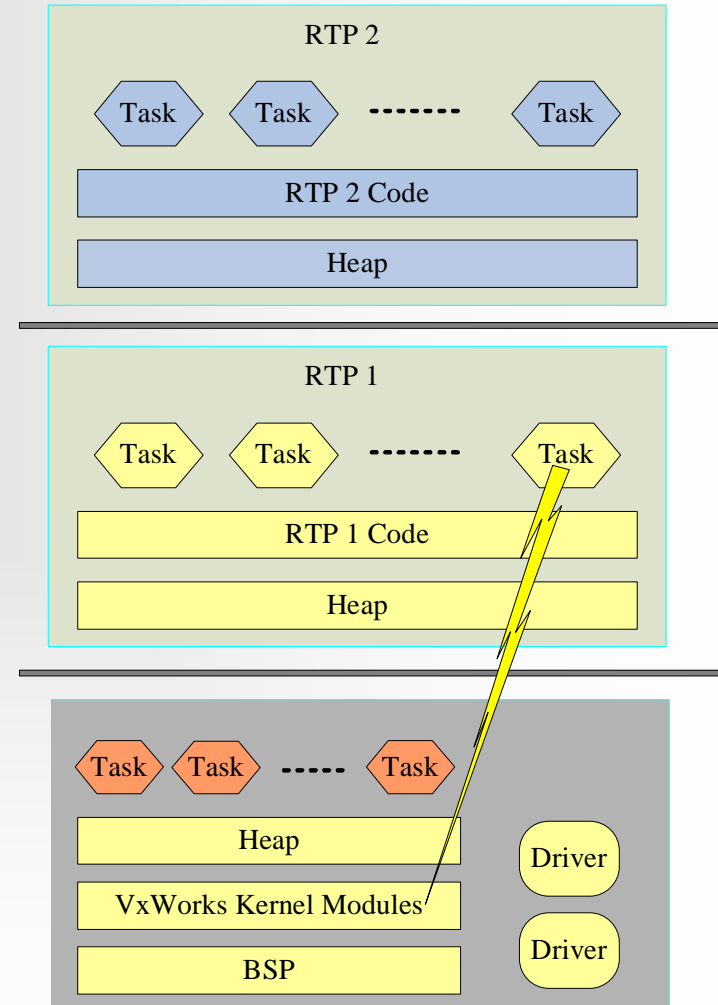
 Accessible in supervisor mode only

RTP地址模型

- ✓ 非重叠地址模型。所有RTP和Kernel共享一个4G的逻辑地址空间。每个RTP占用其中的一段或几段。
- ✓ 每个RTP有自己的上下文。Kernel空间加入到每个RTP的上下文中，降低系统调用的开销。
- ✓ 不同上下文之间的访问保护通过保护属性完成。
- ✓ RTP映像是浮动映像。RTP的地址再加载时由虚存管理器确定。
- ✓ 不支持页面交换。

RTP的实质

- ✓ RTPs是用户态的应用程序
 - ∅ 类似DOS/Windows .exe & Unix executable
 - ∅ 和Kernel的开发独立
- ✓ RTPs实际上是一个“资源容器”
 - ∅ 程序（代码&数据）
 - ∅ 堆栈
 - ∅ 堆
 - ∅ 属于这个RTP的任务
 - ∅ 由RTP创建的各种软件对象
 - ∅ 由RTP管理的各种资源
- ✓ RTP特性
 - ∅ 内存保护（MMU使能）
 - ü 代码/数据/堆栈保护
 - ∅ 内核隔离
 - ü 通过系统调用/TRAP使用内核功能
 - ü 不需要静态链接
 - ∅ 本身不被调度
 - ü RTP中的任务被全局调度
 - ∅ 当所有的任务终止时，资源被回收
 - ∅ 独立于MMU的具体实现



RTP映像和RTP的执行

✓ RTP映像

- Ø 是一个全连接的浮动映像，当使用动态链接库时是部分连接的
- Ø WorkBench提供了构建RTP映像的Project模版
- Ø 构建RTP映像与构建Kernel映像基本相同，只是
 - ü 无需指定内存地址
 - ü 连接的是VxWorks用户库
- Ø RTP映像需要作为一个VxWorks可执行文件（.vxe）存放在文件系统中
 - ü 没有文件系统时，也可以通过ROMFS将.vxe文件绑定到VxWorks映像中

✓ RTP可以通过API调用，Shell命令启动执行

- Ø VxWorks6在用户空间和Kernel空间提供了相同的API和编程模式
 - ü 用户空间的API是系统服务的Stub。Stub通过TRAP进入Kernel，在Kernel执行真正的系统服务
- Ø 系统调用无需上下文切换（Kernel与用户空间在一个上下文中）
- Ø 每个Task有两个堆栈——用户栈和核心栈
- Ø 用户可以方便地增加系统服务

共享库与内核模块

✓ 内核模块 (Downloadable Kernel Modules)

- ∅ 可以在运行时动态加载、卸载
- ∅ 可以和Kernel一起静态链接，或者存放于文件系统中
- ∅ 内核空间执行
- ∅ 通过系统符号表实现动态加载，主要目的是便于内核模式的开发与调试

✓ 共享库 (Shared Library)

- ∅ 相当于Windows中的DLL
- ∅ 可以在运行时动态加载、卸载
- ∅ 用户模式执行
- ∅ 主要目的是实现在进程之间共享代码，应用程序无需包含共享库代码，可以减小程序尺寸。
- ∅ 实现上有细微的不同

Kernel mode Vs. RTP

✓ Size

∅ 对RTP和MMU的支持会导致系统尺寸较大

✓ Speed

∅ 过多的系统调用会降低效率

✓ Kernel-only features

∅ watchdog timers, ISRs, and VxMP

✓ Hardware access

∅ RTP不能直接访问硬件

The logo features the words "FAR SIGHT" in white, bold, sans-serif capital letters. A red, stylized vertical bar separates the two words. The text is centered within a dark green, textured, downward-pointing triangle that has a 3D effect with lighter green highlights on its sides.

FAR SIGHT

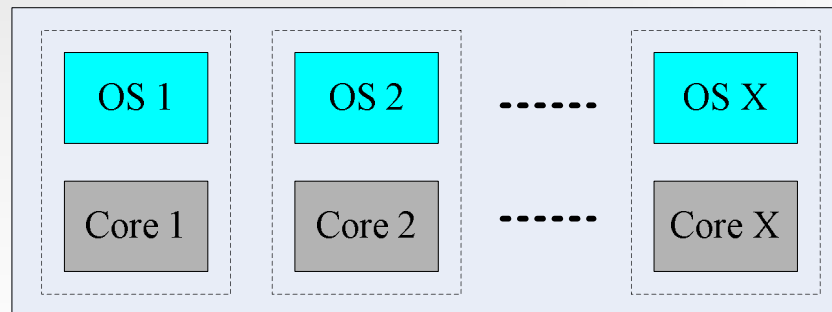
嵌入式培训专家

VxWorks最新技术

对多核的支持 (VxWorks SMP)

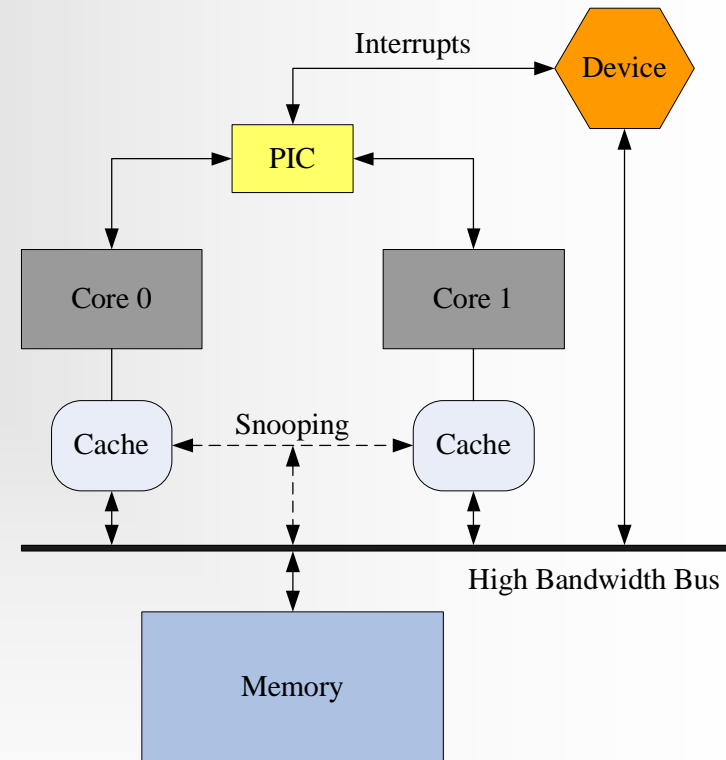
多处理技术的几个概念

- ✓ UP (Uniprocessor)
- ✓ AMP (Asymmetric Multiprocessing)
 - ∅ 分为同质和异质两种
 - ∅ 每个核都有自己专用的主内存，不同核之间可通过共享内存通信
 - ∅ 每个CPU内核运行一个独立的操作系统或同一操作系统的独立实例 (instantiation)
 - ∅ 相互连接进行通信和同步
 - ∅ 工作在设计时进行分割，并分配给不同的处理器
 - ∅ 移植到不同的硬件平台比较困难



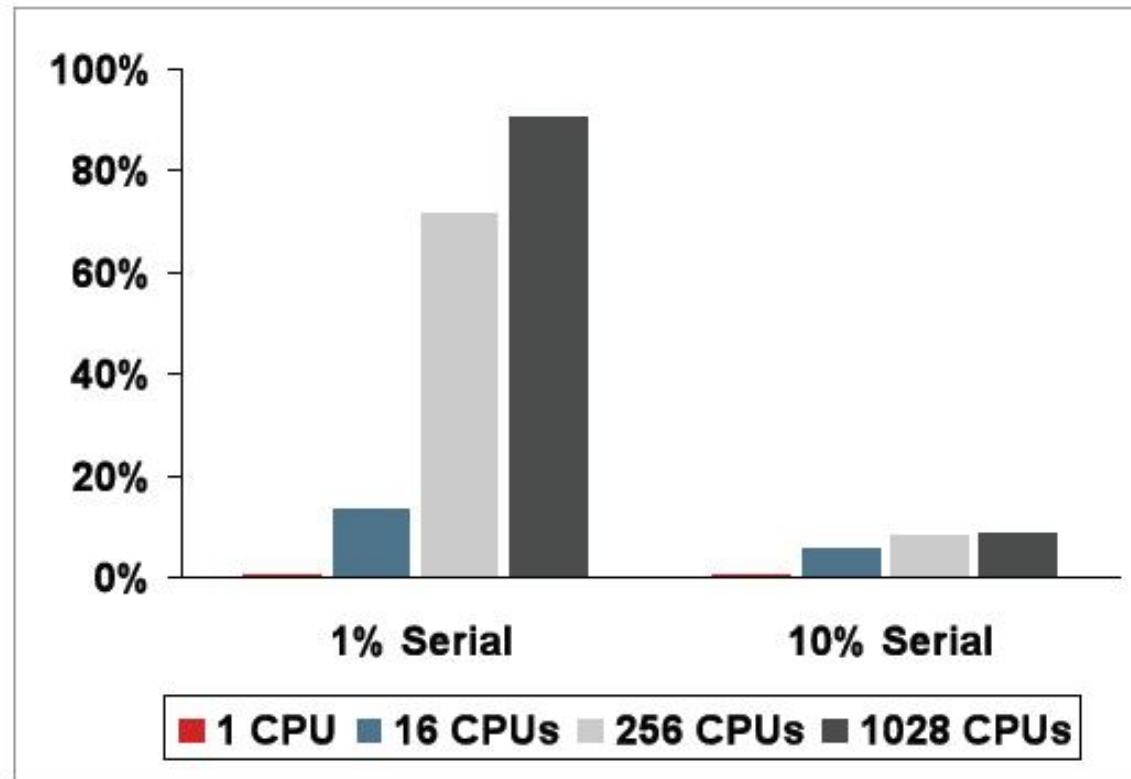
SMP (Symmetric Multiprocessing)

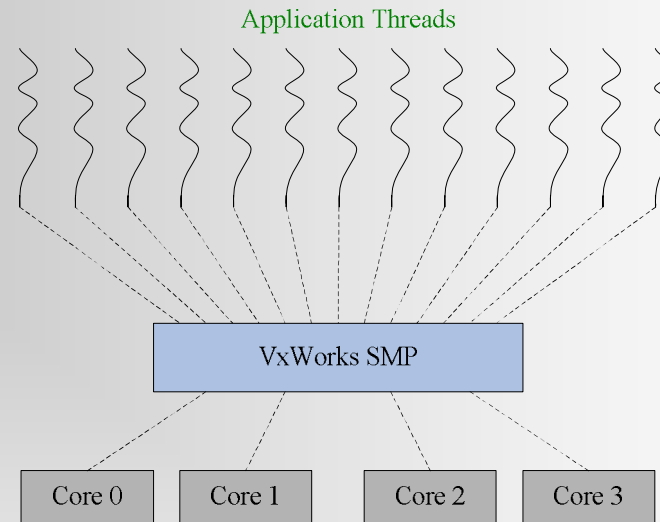
- ✓ 每一个核都可以公平地访问系统设备，如内存。每个设备对不同的核具有相同的物理地址，每个核到设备的“距离”是相同的
- ✓ 因为任务可能会被调度到不同的CPU上，当任务切换CPU时，它所需要的内存可能不在新CPU的缓存中，导致任务切换CPU时的较大开销
- ✓ 中断可以通过可编程中断控制器路由到任何一个核上



Amdahl's Law

All jobs include some work that has to be serial.
Parallel speedup = $1 / (\text{Serial}\% + (1 - \text{Serial}\%) / N)$





- ✓ 一个单独的组件，需要另外购买
- ✓ 硬件的抽象使编程更加容易
- ✓ 所有的核作为资源由操作系统控制
- ✓ 任务中断可以被绑定到特定的核（Affinity）
- ✓ 操作系统可以实现不同核之间的负载均衡
- ✓ 保持和UP VxWorks的最大兼容，便于代码重用
- ✓ 支持主流的多核平台
- ✓ 提供完整的多核开发、调试环境（WorkBench）

VxWorks SMP的特性

✓ VxWorks SMP 的调度器

- ∅ 基于优先级的任务抢占（与UP兼容）
- ∅ 同时有多个任务在执行

✓ 与VxWorks UP保持API兼容

✓ 新的高效同步机制

- ∅ Spin-locks
- ∅ Atomic operators
- ∅ Reader-writer semaphores

✓ 任务与CPU的绑定

- ∅ 绑定一个任务到特定的核
- ∅ 提高性能，方便移植

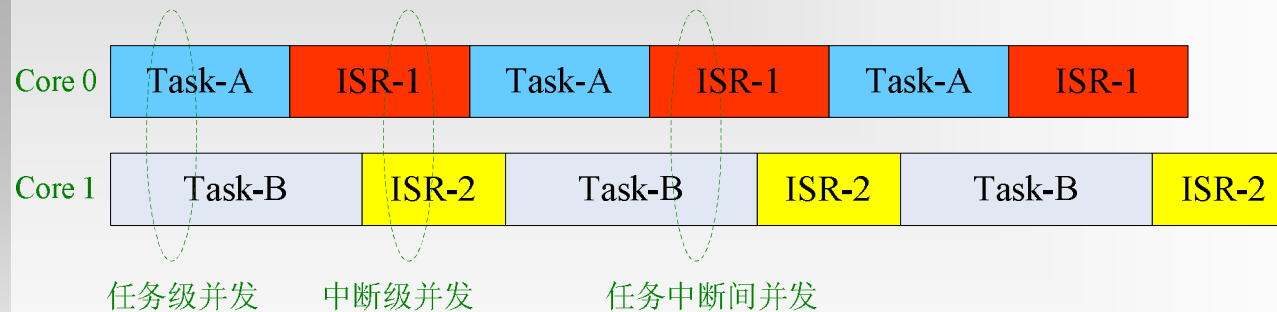
SMP导致并发性

Single CPU System (UP)



UP 系统中ISR与任务严格按照优先级串行执行

Two-Way SMP System



VxWorks SMP 导致任务之间、ISR之间以及任务与ISR之间的并发执行

SMP 带来的开发上的挑战

- ✓ 在任意时间，存在多个任务同时执行
- ✓ 多线程之间的通信和同步更加复杂
- ✓ 如何移植传统的UP应用程序到SMP上，
以充分利用多核资源
- ✓ 多核的调试更加困难
- ✓ 开发人员必须以不同的方式思考

VxWorks SMP API的变化

✓ 下面的API在SMP中不存在

- ∅ intLock/intUnlock
- ∅ taskLock/taskUnlock
- ∅ taskRtpLock/taskRtpUnlock

✓ 新的同步机制 - spin-locks

- ∅ Task-only – 用于多任务间的同步
- ∅ ISR-callable – 用于任务和中断间的同步

✓ Atomic Operators

- ∅ vxAtomicAdd vxAtomicAnd vxAtomicDec
- ∅ vxAtomicInc vxAtomicGet vxAtomicSet

✓ 任务和CPU的绑定

- ∅ taskCpuAffinitySet and taskCpuAffinityGet

✓ Reader/Writer Semaphores

- ∅ semRTake, semWTake

VxWorks 培训的目标

- ✓ 系统地讲解VxWorks的开发流程
- ✓ 补全工程师所缺的相关知识
- ✓ 让工程师具有独立把握整个系统的能力

华清远见

Questions!



FAR  SIGHT