# 手持设备的基本概念和应用

## 基本概念 – 理解

* A *mobile device*(also known as *cellphone device*, *handheld device*, *handheld computer*, *"Palmtop"* or simply *handheld*) is a pocket-sized computing device, typically having a display screen with touch input or a miniature keyboard.

衣袋大小的，通常有一个拥有手触输入或迷你键盘的屏幕的计算机设备

## 应用 – 举例

personal digital assistant (PDA)个人数字助理（掌上电脑）:输入输出功能被结合在手触屏幕上

Enterprise digital assistants企业数字助理

## 相关知识和技术

# 4种嵌入式软件体系结构

* 轮转结构:轮转结构(round-robin)最简单的一种结构

主循环依次检查每个I/O设备，并为需要服务的设备提供服务。 不存在中断

优点 结构简单

缺点 最坏响应时间 缺乏优先级 结构缺乏可扩展性

* 带中断的轮转结构:

中断程序处理硬件特别紧急的需求，然后设置标记；

主循环轮询这些标记，然后根据这些需求进行后续的处理。

优点:为硬件操作提供优先级。实质上，中断程序中的所有操作拥有比主程序中任务代码更高的优先级。

缺点 :结构带来一定的复杂度 共享数据问题

优先级机制不彻底！所有任务代码以同样的优先级来执行。 任务的最坏响应时间：

* 函数队列调度结构

更复杂精细的结构

中断程序在一个函数指针队列中添加一个函数指针，以供主程序调用。

主程序仅需要从该队列中读取相应的指针并且调用相关函数。

引进任务优先级：通过对函数指针排队实现

优点 :提供了任务优先级

缺点:程序复杂 最坏响应时间？

* 实时操作系统RTOS

明确提出“任务”概念

中断程序和任务代码之间的必要信号发送是通过实时操作系统处理的，并不需要使用共享变量来达到这个目标。

代码中并没有用循环来决定下一步要做什么。实时操作系统内部的代码决定什么任务代码可以运行。

操作系统可以在一个任务运行期间将其挂起，以便运行另一个任务。

结构可扩展性强。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 结构种类 | 是否允许优先级 | 任务代码的最坏响应时间 | 代码改变时响应时间的稳定性 | 简单性 |
| 轮转结构 | 不允许 | 所有任务代码的总和 | 差 | 很简单 |
| 带中断的轮转结构 | 中断程序有优先级次序，所有任务代码在同一个优先级上 | 所有任务代码的执行时间的总和(加上中断程序的执行时间) | 中断程序响应时间的稳定性好；任务代码响应时间的稳定性差 | 必须处理中断程序和任务代码的共享数据 |
| 函数队列调度结构 | 中断程序有优先级次序，任务代码也有优先级次序 | 最长函数的执行时间(加上中断程序的执行时间) | 相对较好 | 必须处理共享数据，并且要编写函数排队代码 |
| 实时操作系统结构 | 中断程序有优先级次序，任务代码也有优先级次序 | 0(加上中断程序的执行时间)(对优先级最高的任务来说) | 很好 | 最复杂（尽管多数复杂部分是在操作系统内部） |

### RTOS or RTK

RTOS实时操作系统:

* 明确提出“任务”概念
* 中断程序和任务代码之间的必要信号发送是通过实时操作系统处理的，并不需要使用共享变量来达到这个目标。
* 代码中并没有用循环来决定下一步要做什么。实时操作系统内部的代码决定什么任务代码可以运行。
* 操作系统可以在一个任务运行期间将其挂起，以便运行另一个任务。
* 结构可扩展性强。

RTK实时动态差分法：

一种常用的GPS测量方法。采用了载波相位动态实时差分方法-----可以实时地提供测站点在指定坐标系中的三维定位结果。

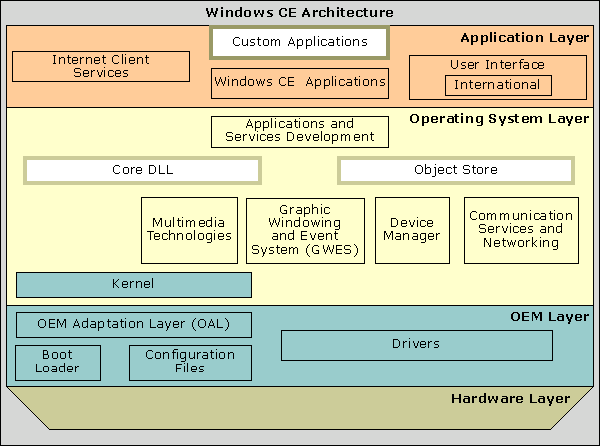
# Windows CE体系结构

## 体系结构的特点

### 分层结构

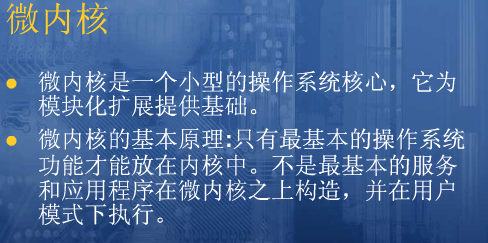
* 体系结构的特点
  + 分层结构（使系统有良好的可扩展性和可维护性）
    - Hardware, OEM, OS layer, Application Layer
  + 微内核特点：内核只实现那些必须由内核实现的基本功能，而将图形系统文件系统，设备驱动及通信等功能放在内核之外，以系统服务的形式提供各种功能。（这种结构的优点是有一个精炼的内核，便于裁剪与移植，而且系统服务运行在用户地址空间，因个别驱动程序的错误不至于导致整个系统崩溃，其不足之处是在运行中用户状态与内核态须频繁切换，从而导致系统效率不如单体内核。）

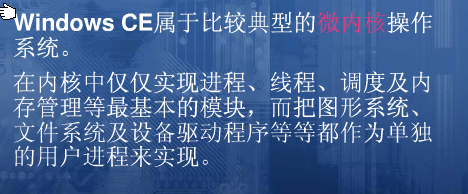
### Hardware, OEM, OS layer, Application Layer

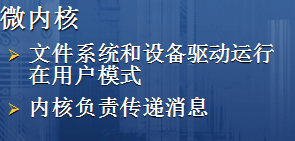


## 微内核特点

* The kernel is represented by New Kernel (NK.exe) module – while system is running.内核被NK.exe模块代替，当系统运行时
  + Core of the operating system操作系统核心
  + Base level functions in kernel: process, thread and memory management基本内核功能：进程，线程，内存管理
  + Includes some file management functions包括一些文件管理函数
  + Kernel services allow applications to use the core functions内核服务器应用程序使用内核函数
* Nk.exe主要由nk.lib和oal.lib组成







## CE6 differences

Two modes of privilege, Kernel and User内核和用户模式

*1.With CE6, many critical OS components moved into the kernel process.* 在CE6，许多关键的操作系统组件搬进了内核进程。

* + *filesys.exe --> filesys.dll*
  + *device.exe --> device.dll*
  + *gwes.exe --> gwes.dll*
* *Primarily for performance reason.* 主要用于性能的原因

*2.New virtual memory layout*新的虚拟内存布局

3.Two modes of privilege, Kernel and User内核用户模式，2种特权模式

内核模式：基本组件的基本服务部分

用户模式：用户程序和DLL

WinCE启动流程

1. CPU加电，跳转到复位向量
2. [可选] 引导程序从Startup()开始执行
3. 执行OAL中的Startup()
4. KernelStart() [ KernelInitialize() For x86 ]
5. Kernel调用 OAL中的OEMInit()
6. 完成内核初始化
7. 内核加载Filesys.exe
8. FileSys初始化注册表
9. 内核加载在HKEY\_LOCAL\_MACHINE\Init 中列出的应用程序

# 存储管理和文件系统





**CE 数据库**模型是针对小型高效存储而优化的小型平面结构。通过函数进行操作





# 应用程序的开发

（Why Platform Manager?

WinCE应用程序开发与Windows桌面应用程序开发的主要不同：

* 在开发桌面应用程序时，应用程序同时在桌面操作系统中运行;
* 在开发WinCE应用程序时，需要将WinCE应用程序download到WinCE目标设备上运行，并且调试的情况也是一样。）

#include <windows.h>

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR szCmdLine, int nShow)

{

MessageBox(NULL, TEXT("Hello, Win32"), TEXT("HelloMsg"), MB\_OK);

return 0;

}

* WinMain() prototype

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPWSTR szCmdLine, int nShow)

Why?

* + Under Windows CE, the command-line string is a Unicode string. In all other versions of Windows, the string is always ASII.

# 系统定制与开发：

* 系统定制与开发的整体流程：

系统定制：1得到并安装bsp ;2 定制操作系统; 3下载到开发板上运行调试：3.1得到并安装bootLoader 3.2配置网络连接 3.3配置调试串口 3.4配置platform builder 3.5连接设置 3.6下载运行映像 ; 4发布操作系统

开发应用程序大致可以分三个步骤：

1 安装合适的sdk 2 编写代码和调试3 发布应用程序

* Build System – 4 steps
  + 与Build有关的文件：Source，DIRS，Makefile

与系统初始化有关的文件：.bib, .reg, .dat, .db

1Sysgen 阶段（生成阶段）2 Feature build 阶段 （编译阶段）

3 Release copy 阶段（release 文件夹复制阶段） 4 Make Image 阶段（镜像打包阶段）

# BSP

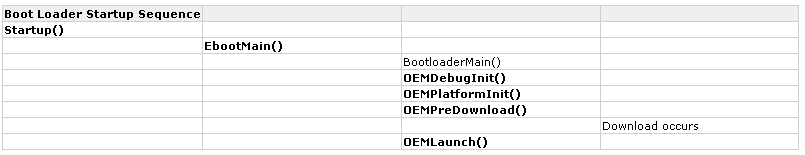
* BSP的概念与组成：4部分

BSP：Board Support Package

* + 主板硬件和操作系统之间的一层软件系统。严格地说，BSP属于操作系统的一部分
  + 解决操作系统跨不同CPU体系结构的方法之一：抽象操作系统和硬件之间的交互接口



* Bootloader的简单分析:功能：初始化目标硬件设备，控制启动过程，下载并执行操作系统映像
  + 代码框架：blcommon oem 代码 eboot 存储管理 EDBG 驱动程序
  + 程序流程



* 黑体字标识的函数需要由OEM厂商来实现.
* 用户需做的：

1实现OEM的应用程序接口（API）.

2连接Microsoft提供的库

引导程序 – StartUp函数

* 硬件复位和运行时复位需要执行的第一条指令
* 设置为超级用户模式
* 执行必须的硬件初始化:
  + CPU 内存控制器 系统时钟 串口 缓存 快表 (TLBs)
* 根据使用的CPU修改Startup.s

引导程序 – EbootMain

* EbootMain是C代码运行的入口
* 调用BLCOMMON库
* BLCOMMON 库 源文件在 Blcommon.c 文件中，路径为%\_WINCEROOT%\Public\Common\Oak\Drivers\Ethdbg directory

引导程序 – OEMDebugInit

* 用来初始化串行口，作为调试输出
* OEMDebugInit初始化完成后, 一个Windows CE的标记会出现，表示这个接口可以使用了.

引导程序 – OEMPlatformInit

* 各种OEM 硬件平台初始化函数，包括时钟, PCI接口,或者NIC接口.
* NIC接口用于下载映象，另外服务于后面一些函数.

引导程序 – OEMPreDownload

* 在加载一个运行时映象时首先被BLCOMMON调用.
* 查找硬件设备的IP地址，并与宿主机相连
* 如果出错返回-1

引导程序 – OEMLaunch

* OEMLaunch 是引导程序的最后一个需要运行的函数.
* 负责跳转的到需要运行的映象.
* 跳转到由dwLaunchAddr指定的第一条指令，这条指令在运行时映象的启动函数里.

**BSP的概念**

BSP：Board Support Package

A board support package (BSP) is the software that implements and supports running an operating system on a standard development board (SDB).

主板硬件和操作系统之间的一层软件系统。严格地说，BSP属于操作系统的一部分

解决操作系统跨不同CPU体系结构的方法之一：抽象操作系统和硬件之间的交互接口

**组成：**

|  |  |
| --- | --- |
| 内容 | 描述 |
| 引导程序 | 加载操作系统映象 |
| OEM 抽象层 (OAL) | 连接内核映象，支持硬件的初始化和管理 |
| 设备驱动 | 支持相关外围设备以及动态安装的设备 |
| 配置文件 | 可以通过对环境变量、.bib文件和 and .reg文件的修改来重新配置BSP |

Bootloader的简单分析

代码框架



程序流程



