# 手持设备的基本概念和应用

## 基本概念 – 理解

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

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

# 四种嵌入式软件体系结构

## 轮转结构

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 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:为中断程序、硬件操作提供优先级。

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。

## 函数队列调度结构

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

## 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

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，但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——介绍见上 2.4

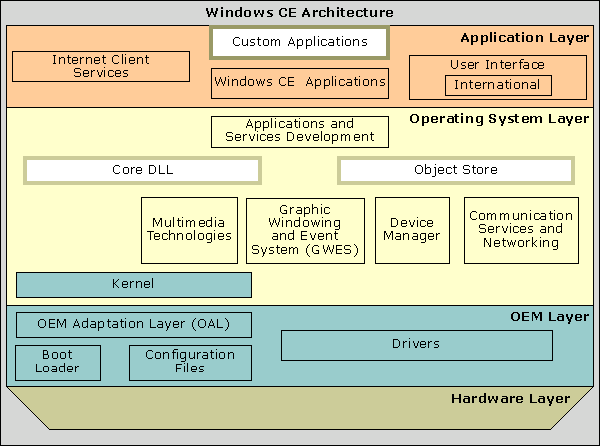
Real Time Kernel实时内核

# Windows CE体系结构

## 体系结构的特点——分层结构

分层结构（使系统有良好的可扩展性和可维护性）

Hardware——OEM——OS——Application



## 微内核特点

微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之外，以系统服务的形式提供各种功能，在用户模式下执行，内核用来传递消息。

优点：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

不足：在运行中用户态与内核态须频繁切换，从而导致系统效率不如单体内核。

WinCE 属于比较典型的微内核操作系统。内核中仅实现进程、线程、调度及内存管理等最基本模块。而把图形、文件和设备系统等都作为单独的用户进程实现。

## CE6 differences

内核和用户模式

* 在CE6，许多关键的操作系统组件搬进了内核进程。并由dll实现。
  + filesys.exe --> filesys.dll
  + device.exe --> device.dll
  + gwes.exe --> gwes.dll （ui）
  + 主要由于性能的原因
* New virtual memory layout新的虚拟内存布局
* 两种特权模式
  + 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

由Filesys.exe管理（->Filesys.dll in CE6）

* 对象存储的概念
  + （物理上）对象存储是一个被Filesys.exe管理的内存堆；如果有后备电源，它可用于为应用程序提供永久存储。
  + （逻辑上）可分为：RAM文件系统、注册表和CE数据库。
* 操作系统使用对象存储技术完成以下任务：
  + 管理栈和内存堆；按需求压缩或展开文件；无缝地集成基于ROM的数据和基于RAM的数据。
* 文件系统
  + Built-in 文件系统
    - ROM-RAM文件系统（\目录）
    - ROM-only文件系统（\Windows目录）
  + 可安装的文件系统（\Storage Card, \CD Driver, etc.）
* 注册表
  + 基于RAM的注册表
  + 基于Hive的注册表
* CE数据库(略)

XIP：Execute In Place（本地执行）

* + 允许程序代码不进入RAM，直接在ROM/Flash中执行
* 优点：
  + 代码段不必先加载到物理内存中，节省内存。（在Windows CE中，OS分配虚拟地址空间给代码段，并将其映射到ROM/Flash中）
* 缺点：
  + 只支持允许线性访问的ROM/Flash（e.g. NOR Flash）
  + 执行速度相对较慢；不适合实时性要求高的场合。

norF 容量小，线性访问。优于直接存放程序，可以放置启动代码。

nandF 容量较大，优于存放数据，主要存储硬盘，存放操作系统和文件系统等。三星的处理器，可以复制nandF前4k到ram，并开始引导。

# 应用程序的开发

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。）

# 系统定制与开发：

## 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

1 安装合适的sdk

2 编写代码和调试

3 发布应用程序

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

配置文件的用途：.bid

## Build System – 4 steps

生成阶段——System Generation

编译阶段——Build

release 文件夹复制阶段——Building the Release Directory

镜像打包阶段——Making Image——NK。BIN

Nb0；nb1 。等等。每块大小有限，nb0不够大了，就出了nb1.

在实验中，因WinCE从NandFlash启动，BSP build后还产生了一个文件stepldr.nb1（应烧写在NandFlash的block 0）。

# BSP

## 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

|  |  |
| --- | --- |
| 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 | 修改环境变量.bib文件和.reg文件来配置BSP |

## 运行流程

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（bootloaderMain）

EbootMain是C代码运行的入口

调用Blcommon.c 文件中的BLCOMMON库

引导程序 – OEMDebugInit

用来初始化串行口，作为调试输出

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

引导程序 – OEMLaunch

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.



# 关于作业

* 每个作业的基本流程
  + 系统开发和应用开发
  + 开发工具（远程调试工具）；配置文件

Visual Studio Remote Tools

* 作业背后的idea
  + Windows CE和桌面Windows开发和调试的异同
  + 作业中涉及的Windows CE基本概念和工作流程