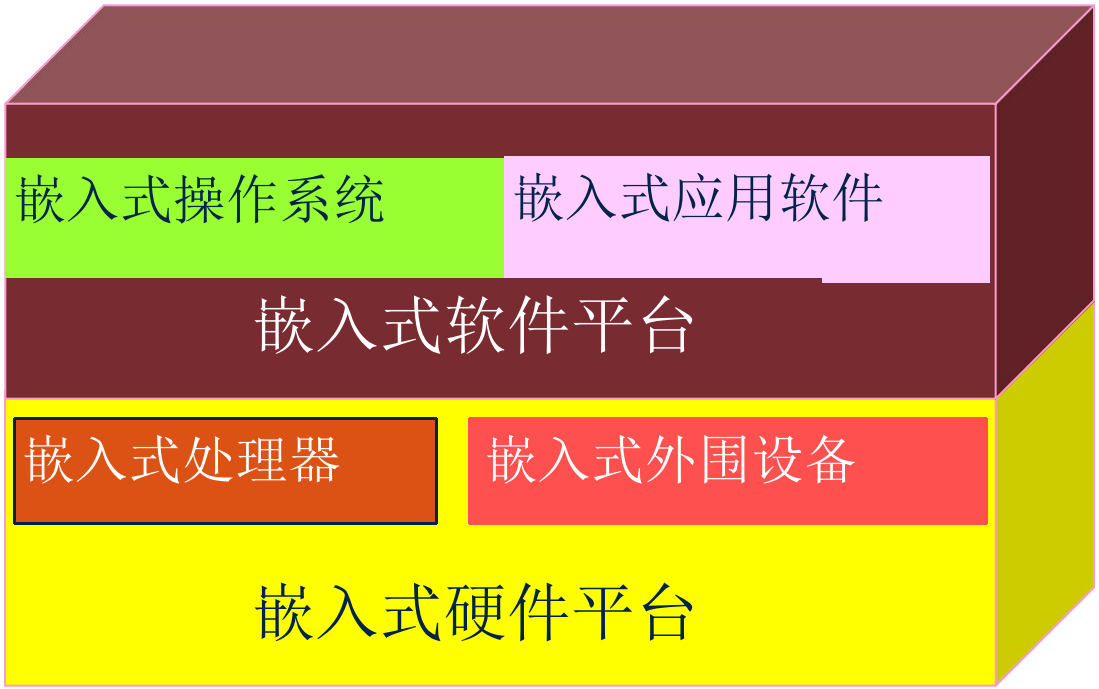
嵌入式系统

总复习

嵌入式系统概述

- 什么是嵌入式系统（定义）？
  - 以应用为中心
  - 可剪裁软硬件
  - 专用计算机系统

嵌入式系统组成



- 硬件核心是什么？
- 常见的外围设备有哪些？

嵌入式计算机系统同通用型计算机系统相比

- **应用特性**
- **设计方法**
  - 量体裁衣、去除冗余（专用性）
- **开发能力**
  - 嵌入式系统本身不具备自主开发能力
- **其他**
  - 升级换代和具体产品同步进行
  - 软件固化

嵌入式处理器相关知识

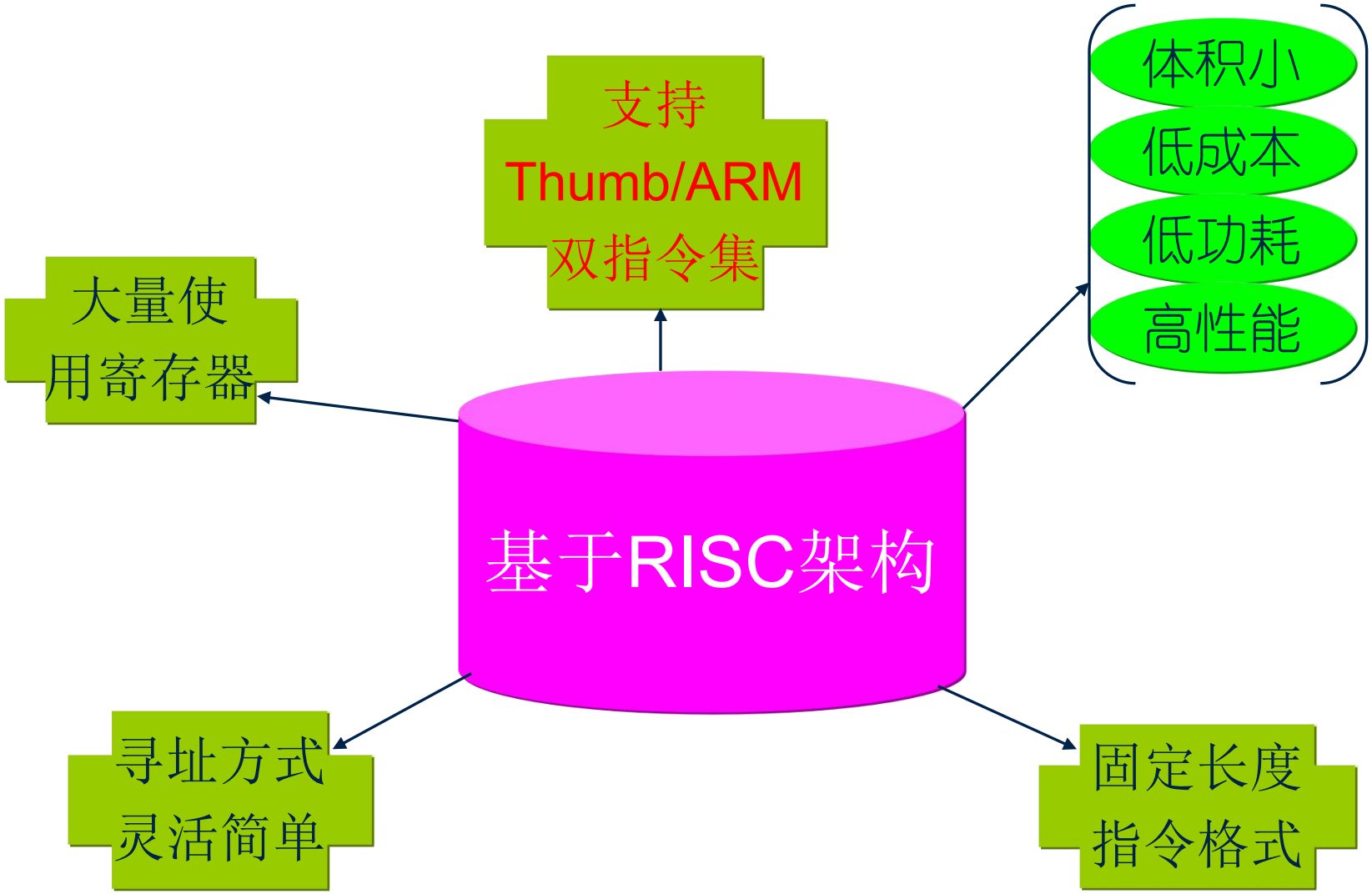
- **嵌入式处理器结构（区别）**
  - 冯诺依曼结构（普林斯顿结构）
  - 哈佛结构
- **关键寄存器**
  - CPSR
  - SPSR
  - PC
- **指令集，微架构，芯片产品**
  - 关系
  - RISC-V特点（相对ARM的比较优势）

嵌入式处理器

- 嵌入式处理器分类
  - EMPU，MCU，EDSP，SOC
- **EMPU种类（了解）**
  - MIPS、Power PC、SH处理器、ARM
- 区别**ARMv7**和**ARM7**

什么是**ARM**？

ARM特点？


支持
Thumb/ARM
双指令集
体积小
低成本
低功耗
高性能
大量使
用寄存器
基于RISC架构
寻址方式
灵活简单
固定长度
指令格式

ARM微处理器- Cortex

- 三个系列：Cortex-A、Cortex-R和Cortex-M
- Cortex-A面向高性能应用，它具有长达13级的流水线，并且可以支持1-4个核
- Cortex-R面向具有高实时性要求的应用，通常应用于专用集成电路(ASIC)，仍然采用8级流水线
- Cortex-M是全球微控制器的标准，面向对能耗和价格有较高要求的用户，采用低延迟的3级流水线

ARM指令集

- **指令集特征**
  - ARM指令集属于加载/存储型指令
  - 指令的操作数都储存在寄存器中，处理结果直接放回到目的寄存器中
- **指令基本组成部分**
- **ARM指令和Thumb指令的区别**
- **寻址方式**
  - 立即寻址
  - 寄存器寻址：ADD R1, R1, R2
  - 寄存器偏移寻址
  - …

嵌入式软件-----嵌入式操作系统

- **操作系统作用？**
  - 系统资源管理
  - 硬件虚拟化
  - 提供资源

- **常见嵌入式OS？**
  - 嵌入式Linux，Windows CE，Symbian，VxWorks，QNX，Palm等
  - 鸿蒙（特点？）

嵌入式Linux特点

- **Linux组成结构（4部分）**
  - Linux内核组成结构（5部分）

- **特点（了解）**
  - 广泛的硬件支持
  - 内核高效稳定
  - 开放源码，软件丰富
  - 优秀的开发工具
    - 以gcc做编译器，以gdb, kgdb, xgdb调试
  - 完善的网络通信和文件管理机制

内核版本号特征？

嵌入式Linux基础知识要点

- **Linux Shell操作**
  - 提示符$、#
- **Linux编程**
  - make及Makefile的使用方法

Linux Shell操作

- **Linux常用命令的使用**
  - ls
  - cp
  - rm
  - mv
  - tar
  - mount
  - ln
  - chmod, chown
  - mkdir, rmdir
  - vi / vim
  - find
- **Linux文件权限管理**
  - 三段式

```
[user@localhost tt]$ ls -l
总用量 4
-rw-rw-r--. 1 user user 216  9月   8 10:36 memo.tar.gz
```

Linux编程

- 编程基础
  - C语言
  - vi使用
- 编译方法

3种模式？

gcc vs. arm-linux-gcc?

问题：
当前工作目录为/opt，且在/opt/hello/中已编写好hello.c程序，请在Linux终端（Shell）下编译并运行hello.c程序
root@opt#

```
cd hello
gcc -o hello -c hello.c
./hello
```

编译工具make及Makefile

- **基本概念**
  - 什么是make、makefile？
- **Makefile构成**
  - 分析Makefile

```
write.o : io.h  write.c
        gcc –c write.c
```

找出“目标”、“依赖”、“命令”的对应代码

Makefile使用要点

- 宏及用法
  - $(宏标识符)
  - $(CC)

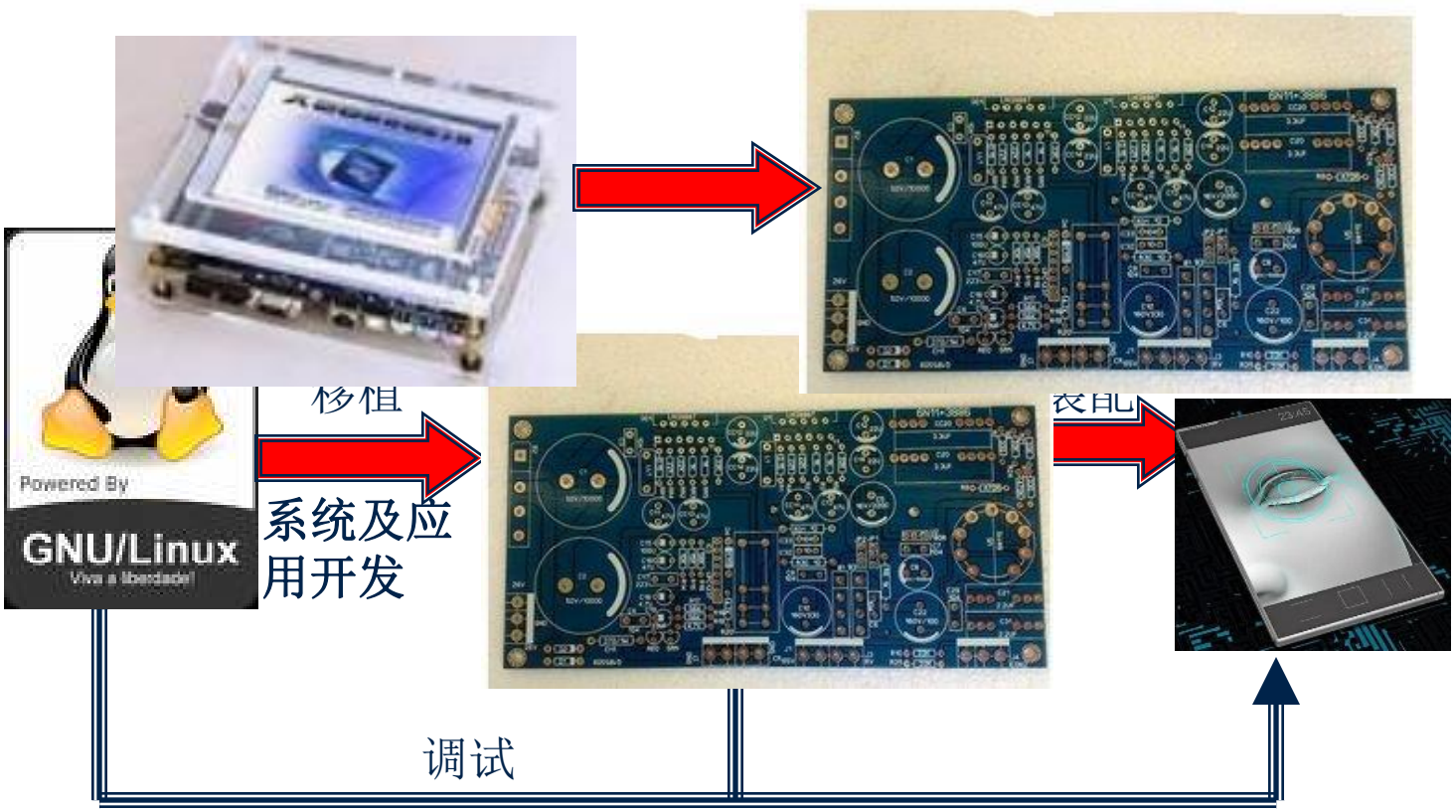
```
# makefile test for hello program
#written by Emdoor
CC=gcc
CFLAGS=
OBJS=hello.o
all: hello
hello: $(OBJS)
        $(CC) $(CFLAGS) $(OBJS) –o hello
hello.o: hello.c
        $(CC) $(CFLAGS) –c hello.c –o $(OBJS)
clean:
        rm –rf hello *.o
```

- 汇编语言程序
- 可重入问题
- C语言和汇编混合编程

嵌入式存储-Flash Memory

- **Non-Volatile内存**
  - 写平衡？
- **主要技术**
  - NAND和NOR
- **比较**
  - NAND：单元排列串行，以块和页为单位进行读写，顺序读写，随机读写慢
  - NOR：　单元排列并行，按字节进行读写，随机读写快，可以片内执行

如何进行嵌入式开发？如智能手机开发？



嵌入式Linux开发的主要步骤

- 选择硬件开发平台
- **建立嵌入式Linux开发环境**
- 系统软件开发
  - 建立引导装载程序Bootloader
  - **ARM-Linux内核**
  - **嵌入式文件系统**
  - **嵌入式设备驱动**
  - **嵌入式GUI**
- 嵌入式应用开发—Android/鸿蒙应用开发

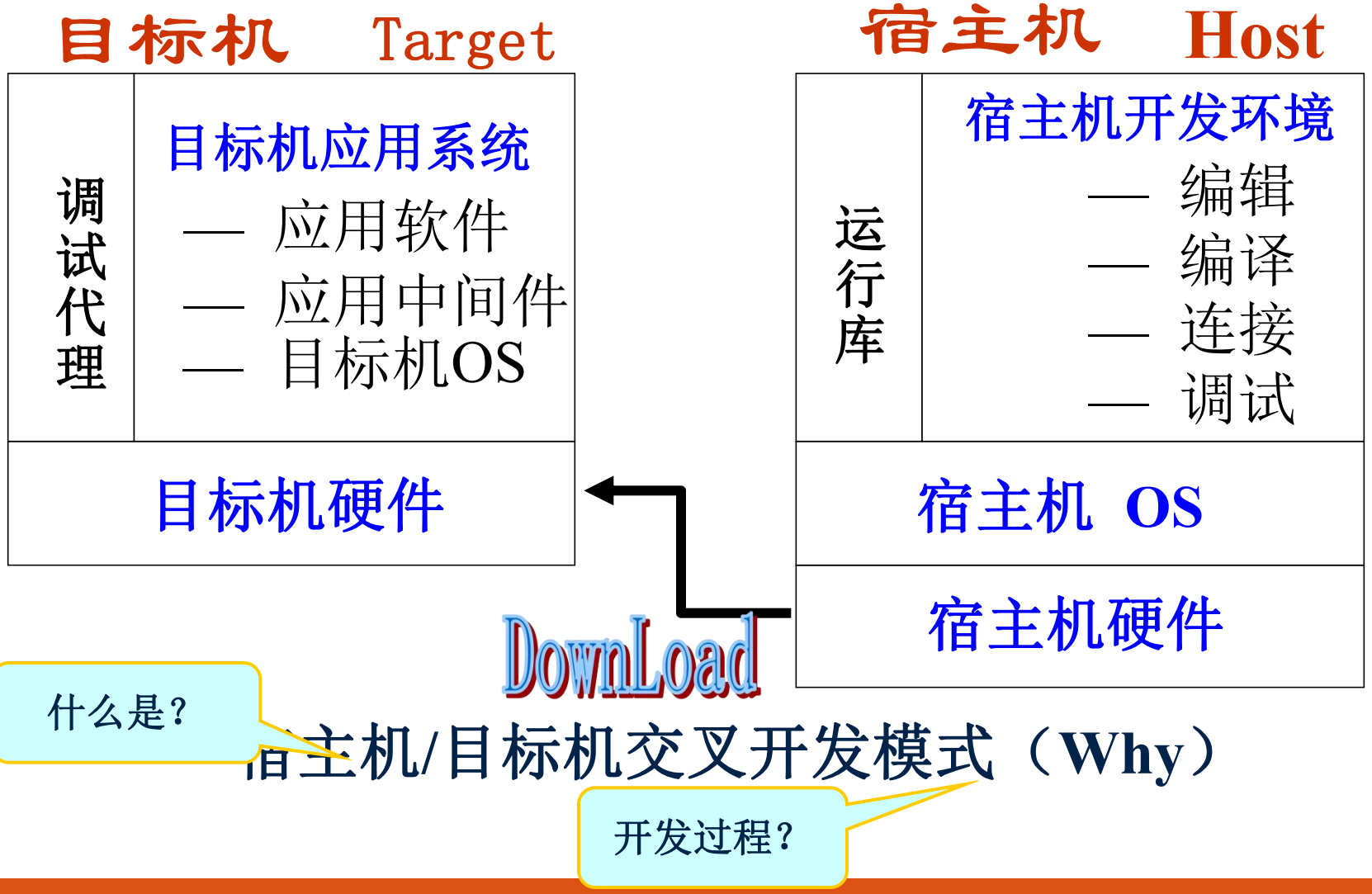
嵌入式Linux开发的主要步骤

- 选择硬件开发平台
- **建立嵌入式Linux开发环境**
- 系统软件开发
  - 建立引导装载程序Bootloader
  - **ARM-Linux内核**
  - **嵌入式文件系统**
  - **嵌入式设备驱动**
  - **嵌入式GUI**
- 嵌入式应用开发—Android/鸿蒙应用开发

嵌入式Linux开发的主要步骤

- 选择硬件开发平台
- **建立嵌入式Linux开发环境**
- 系统软件开发
  - 建立引导装载程序Bootloader
  - **ARM-Linux内核**
  - **嵌入式文件系统**
  - **嵌入式设备驱动**
  - **嵌入式GUI**
- 嵌入式应用开发—Android/鸿蒙应用开发

嵌入式系统开发模式


目标机 Target
宿主机 Host
调试代理
目标机应用系统
— 应用软件
— 应用中间件
— 目标机OS
目标机硬件
运行库
宿主机开发环境
— 编辑
— 编译
— 连接
— 调试
宿主机 OS
宿主机硬件
DownLoad
什么是？
宿主机/目标机交叉开发模式（Why）
开发过程？

嵌入式开发模式

- 硬件环境
  - 通信接口和设备
- 软件环境

宿主机环境的建立

什么是交叉编译？

- 交叉编译环境
  - **安装交叉编译工具集ToolChain**
- 建立宿主机-目标机之间的通信连接
  - **JTag口**
  - **实现串口通信**
  - **TFTP协议**

主要功能？

配置方法？

作用？

交叉编译

- 交叉编译就是在一个架构下编译另一个架构的目标文件
- 采用何种交叉编译器产生何种格式的目标文件还要取决于目标机的操作系统。

TFTP协议

- **TFTP服务的全称是Trivial File Transfer Protocol**
  - TFTP可以看成一个简化了的FTP
- **特点**
  - TFTP承载在UDP上
  - 最普遍使用的是第二版TFTP使用UDP 的67端口
- **TFTP在安装时一定要设立一个单独的目录作为TFTP服务的根目录，以减少安全隐患**
- **利用tftp下载Linux映像**

Linux映像(image)文件结构

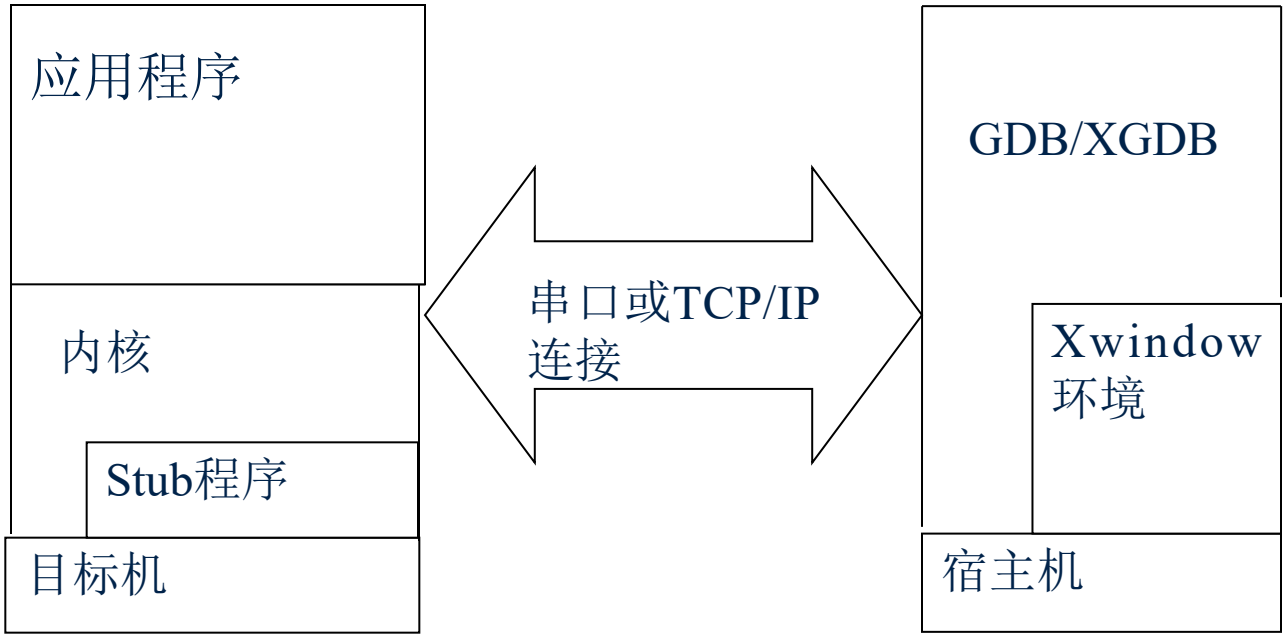
Linux下的调试--gdb调试

- 运行程序，可以给程序加上所需的任何调试条件
- 在给定的条件下让程序停止
- 检查程序停止时的运行状态
- 通过改变一些数据，可以更快地改正程序的错误

- **GDB用法？**
  - 在编译时，必须要把调试信息加到可执行文件中
  - 使用编译器（ cc/gcc/g++）的 -g 参数

搭建嵌入式系统的远程调试环境

- **搭建原理**
  - stub方案
- **远程调试原理**



嵌入式Linux开发的主要步骤

- 选择硬件开发平台
- **建立嵌入式Linux开发环境**
- 系统软件开发
  - 建立引导装载程序Bootloader
  - **ARM-Linux内核**
  - **嵌入式文件系统**
  - **嵌入式设备驱动**
  - **嵌入式GUI**
- 嵌入式应用开发—Android/鸿蒙应用开发

Bootloader

- PC机与嵌入式系统的启动过程及其差别？
- **什么是BootLoader？**
- 主要工作
  - 初始化硬件设备和建立内存空间的映射图
  - 将系统的软硬件环境带到一个合适的状态
  - 为最终调用操作系统内核准备好正确的环境

常见Bootloader？

相关概念

- **Boot Loader的操作模式**
  - 启动加载模式
  - 下载模式
    - 如何使用串口终端
- **Boot Loader与主机之间的通信设备及协议**
  - 最常见的是串口，协议xmodem/ymodem/zmodem
  - 以太网，协议tftp
- **操作系统的角度看，Boot Loader的总目标就是正确地调用内核来执行**

Boot Loader的典型结构框架

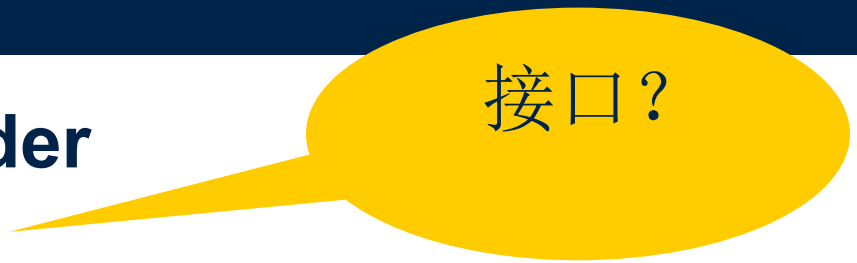
- 大多数Boot Loader都分为阶段1和阶段2两大部分
  - **阶段1的任务？**
    - **阶段1实现依赖于CPU体系结构的代码 (汇编)**
  - **阶段2的任务？**
    - **阶段2实现一些复杂的功能 （C语言）**

（了解）两阶段主要工作

- **阶段1介绍**
  - 1）硬件设备初始化。
  - 2）为加载阶段2准备RAM空间
  - 3）拷贝阶段2到RAM中
  - 4）设置堆栈指针sp
  - 5）跳转到阶段2的C入口点
- **阶段2介绍**
  - 1）初始化本阶段要使用到的硬件设备
  - 2）检测系统的内存映射
  - 3）加载内核映像和根文件系统映像
  - 4）设置内核的启动参数
  - 5）调用内核

Boot Loader的使用

- **烧写Boot Loader**
  - 烧写方法



- **Boot Loader加载或烧写内核和文件系统**
  - Boot Loader的操作模式
  - Bootloader编程： trampoline
- **Bootloader如何检测内存（RAM）可读写？**
- **Bootloader与内核的通信**

嵌入式Linux开发的主要步骤

- 选择开发平台
- **建立嵌入式Linux开发环境**
- 系统软件开发
  - 建立引导装载程序Bootloader
  - **ARM-Linux内核**
  - **嵌入式文件系统**
  - **嵌入式设备驱动**
  - **嵌入式GUI**
- 搭建远程调试环境

内存管理

- **内存管理内容**
  - 包含地址映射、内存空间的分配，有时候还包括地址访问的限制（即保护机制）
  - 如果将I/O也放在内存地址空间中，则还要包括I/O地址的映射
  - 另外，像代码段、数据段、堆栈段空间的分配等等都属于内存管理
- **影响内存管理的两个方面**

ARM-Linux内核知识要点

- **MMU**

作用？

地址映射方式种类？

- **MMU**
  - “内存管理单元”
    - 地址映射
    - 对地址访问的保护和限制
  - MMU可以做在芯片中，也可以作为协处理器

- 虚拟内存
- 内存映射模型

进程管理和调度

- 什么是进程？
- **Linux进程的创建**
  - 三个系统调用
- **Linux进程的执行**
  - fork vs. exec
- 进程的销毁—三个事件驱动

进程调度依据

- Policy
  - 调度策略，用来区分实时进程和普通进程
  - 实时进程会优先于普通进程运行
- Priority
  - 进程(包括实时和普通)的静态优先级
- Counter
  - 进程剩余的时间片，起始值就是priority的值
  - 可以看作是进程的动态优先级
- rt_priority
  - 实时进程特有的，用于实时进程间的选择

Linux模块机制

- 必要性？
- 模块的代码结构
- 相关的主要命令

与Linux模块相关的命令

- **lsmod** 把现在 **kernel** 中已经安装的**modules** 列出来
- **insmod** 把某个 **module** 安装到 **kernel** 中
- **rmmod** 把某个没在用的 **module** 从**kernel**中卸载
- **depmod** 制造 **module dependency file**，以告诉将来的 **insmod** 要去哪儿找**modules** 来安装

中断

- 一个流程
- 三个环节
  - 中断响应
  - 中断处理
  - 中断返回
- GPIO

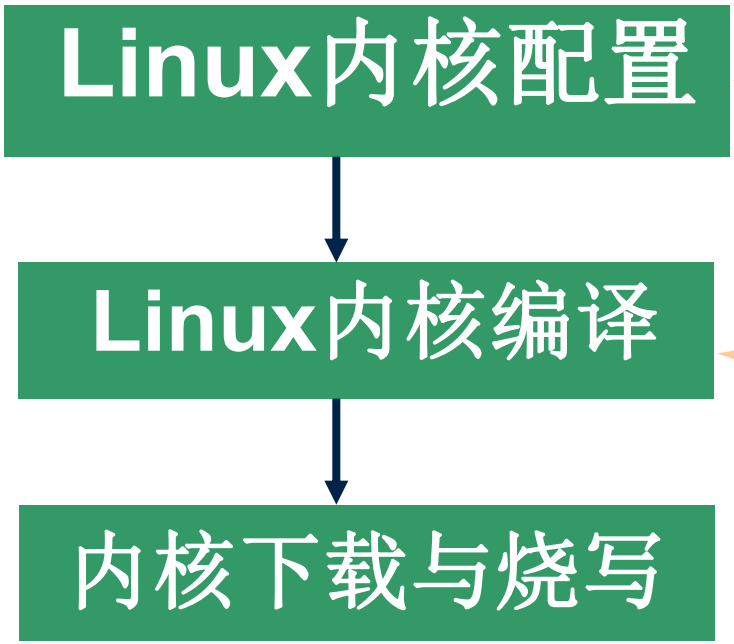
系统调用

- 实现方式
  - X86
  - arm

启动和初始化

- 过程
- **/etc/inittab**
  - id:runlevel:action:process

嵌入式Linux内核编译



区别make config、
make menuconfig、
make xconfig？

区别make,
make zImage,
make bzImage？

嵌入式Linux开发的主要步骤

- 选择开发平台
- **建立嵌入式Linux开发环境**
- 系统软件开发
  - 建立引导装载程序Bootloader
  - **ARM-Linux内核**
  - **嵌入式文件系统**
  - **嵌入式设备驱动**
  - **嵌入式GUI**
- 嵌入式应用开发—Android/鸿蒙应用开发

嵌入式文件系统知识点

- 控制设备驱动的途径有哪些？
- Linux文件系统组织结构？
- Linux文件系统与Windows文件系统区别？

常用文件系统

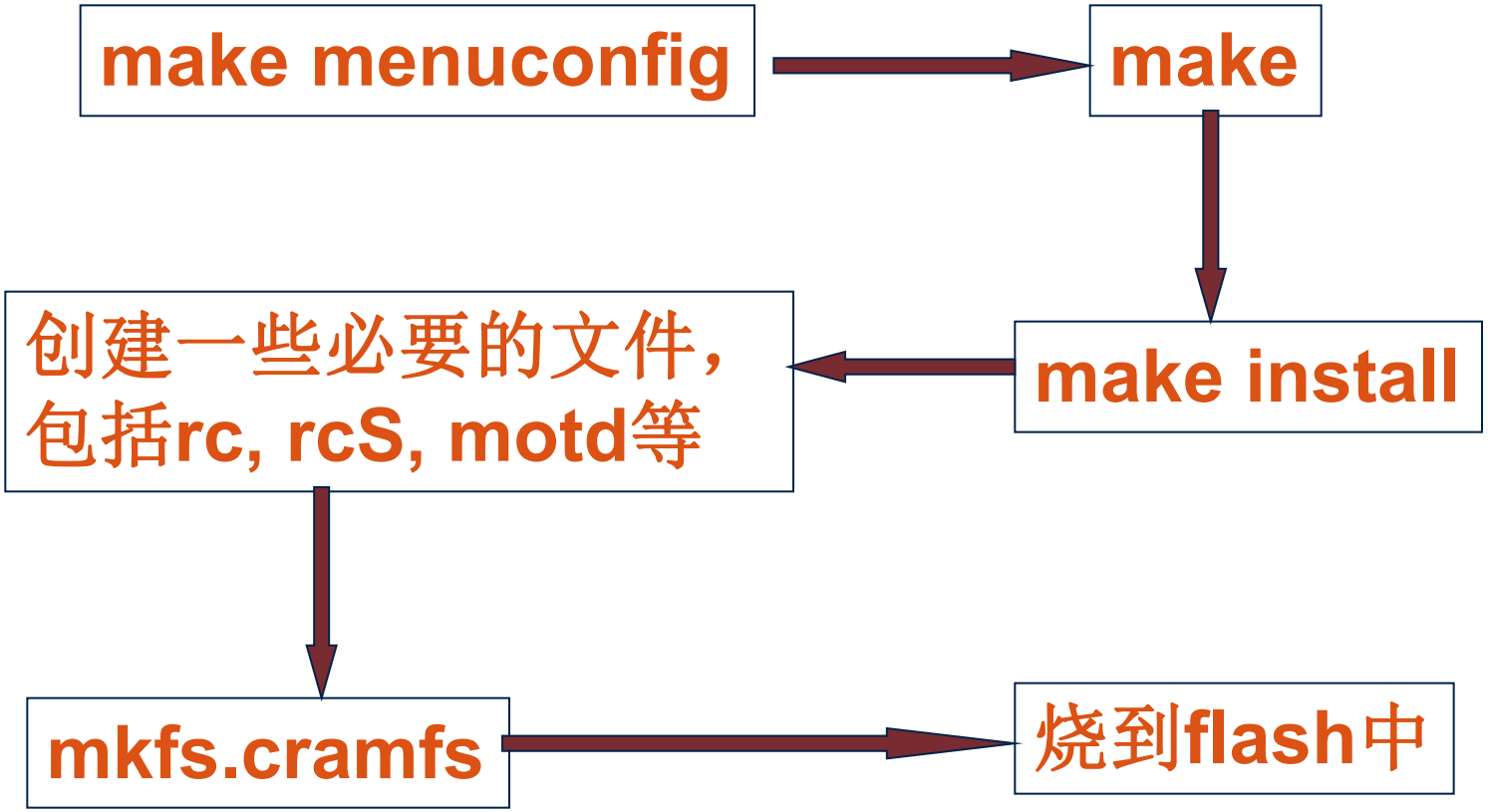
- 通用文件系统
  - **ext2**
- 常用嵌入式文件系统
  - **NAND Flash → YAFFS2**
  - **NOR Flash → JFFS2**
  - **RAM → RAMFS**
  - **Network → NFS**

能否做嵌入式文件系统?

根文件系统

- 什么是根文件系统？
- 如何建立根文件系统？

cramfs嵌入式根文件系统的构造



嵌入式Linux开发的主要步骤

- 选择开发平台
- **建立嵌入式Linux开发环境**
- 系统软件开发
  - 建立引导装载程序Bootloader
  - **ARM-Linux内核**
  - **嵌入式文件系统**
  - **嵌入式设备驱动**
  - **嵌入式GUI**
- 搭建远程调试环境

嵌入式Linux设备驱动

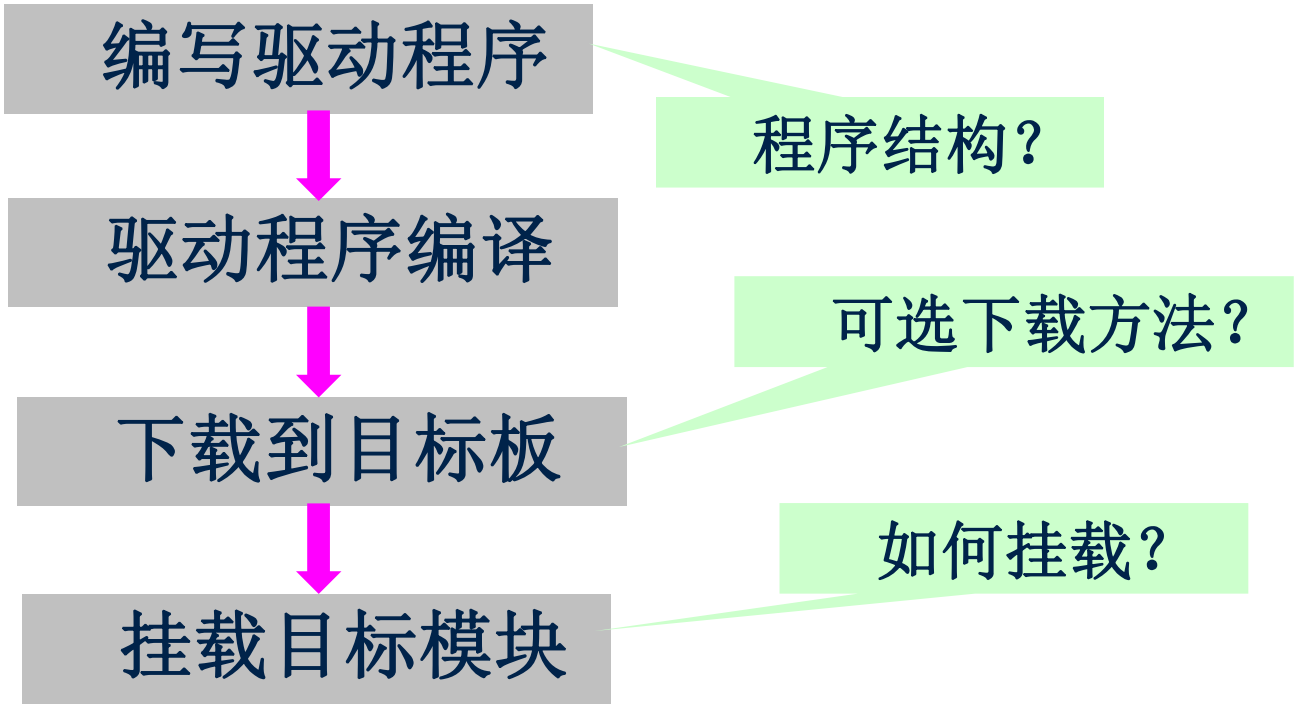
- 基本概念
  - 作用
  - 设备的分类
  - 设备文件
  - 设备号
- Linux的设备驱动程序功能
- **sysfs文件系统**

- **设备驱动的功能**
  - 对设备的初始化和释放
  - 把数据从内核传送到硬件和从硬件读取数据
  - 读取应用程序传送给设备文件的数据和回送应用程序请求的数据
  - 检测和处理设备出现的错误
- **Linux设备的分类**
  - 字符设备
  - 块设备
  - 网络设备

- **Linux设备文件**
  - Linux抽象了对硬件的处理，所有的硬件设备都可以作为普通文件一样来看待
  - 可以使用和操作文件相同的、标准的系统调用接口来完成打开、关闭、读写和I/O控制操作
  - 对用户来说，设备文件与普通文件并无区别
- **主设备号**
  - 标识该设备的种类，也标识了该设备所使用的驱动程序
- **次设备号**
  - 标识使用同一设备驱动程序的不同硬件设备
- **[root@wzchent]# mknod /dev/lp0 c 6 0**

- **Linux设备驱动程序的代码结构大致可以分为如下几个部分**
  - 驱动程序的注册与注销
  - 设备的打开与释放
  - 设备的读写操作
  - 设备的控制操作
  - 设备的中断和轮询处理

驱动程序的编写及加载



同步机制

- **2.6内核同步**
  - 同步锁、信号量、原子操作和完成事件
- **设备操作—同步 or 异步**

DMA（Direct Memory Access）

- 直接内存存取
- 解决快速数据访问
- DMA控制器可以不需要处理器的干预，在设备和系统内存高速传输数据

字符型设备驱动

- 数据结构
  - struct file_operations
  - struct file
  - struct inode
- 串行总线接口
  - SPI
  - I2C：I2C驱动架构有三个组成部分

块设备驱动

- 数据结构
  - gendisk
  - request_queue
  - request
- 主要工作
  - 初始化
  - 请求处理

网络驱动

- 数据结构
  - net_device
  - sk_buffer
- 主要工作
  - 初始化
  - 打开接口

嵌入式Linux开发的主要步骤

- 选择开发平台
- **建立嵌入式Linux开发环境**
- 系统软件开发
  - 建立引导装载程序Bootloader
  - **ARM-Linux内核**
  - **嵌入式文件系统**
  - **嵌入式设备驱动**
  - **嵌入式GUI**
- 嵌入式应用开发—Android/鸿蒙应用开发

嵌入式GUI

- 基本概念
- 分类
  - 主流嵌入式GUI
- 体系结构--分层设计
  - 初始化
  - 打开接口

嵌入式GUI

- 基本概念
- 分类
  - 主流嵌入式GUI
- 体系结构--分层设计
  - 初始化
  - 打开接口

嵌入式Linux开发的主要步骤

- 选择开发平台
- **建立嵌入式Linux开发环境**
- 系统软件开发
  - 建立引导装载程序Bootloader
  - **ARM-Linux内核**
  - **嵌入式文件系统**
  - **嵌入式设备驱动**
  - **嵌入式GUI**
- **嵌入式应用开发—Android/鸿蒙应用开发**

Android应用程序构成

- 一个Android应用程序
  - 活动（Activity）
  - 意图（Intent）
  - 服务（Service）
  - 内容提供器（Content Provider）
- 生命周期
- 开发环境

鸿蒙应用开发

- 基本开发流程
- **Ability**
  - **FA**
  - **PA**
- 技术特性

THANKS!