嵌入式笔记 - 第五章

> *接下来的几章着重介绍嵌入式系统开发流程的几个步骤，这一章主要介绍“交叉编译”及“BootLoader”的概念。*

嵌入式系统开发流程

嵌入式处理器的选型

选择存储设备，选择音频设备，选择显示设备

外围电路设计

选择内核版本

建立交叉开发平台

移植交叉编译工具链

移植Bootloader

移植系统内核

制作文件系统

编写设备驱动

开发应用程序

交叉调试

交叉编译

交叉编译概念

交叉编译是嵌入式开发过程中的一项重要技术，它的主要特征是某机器中执行的程序代码不是在本机编译生成，而是由另一台机器编译生成，一般把前者称为目标机，后者称为主机。

采用交叉编译的主要原因在于，多数嵌入式目标系统不能提供足够的资源供编译过程使用，因而只好将编译工程转移到高性能的主机中进行。
因为目的平台上不允许或不能够安装我们所需要的编译器，而我们又需要这个编译器的某些特征；有时又是因为目的平台还没有建立，连操作系统都没有，根本谈不上运行什么编译器。

主机与目标机连接方式

JTAG(Joint Test Access Group) 并行电缆

JTAG是一种国际标准测试协议（IEEE 1149.1兼容），主要用于芯片内部测试。现在多数的高级器件都支持JTAG协议,如DSP、FPGA器件等。
标准的JTAG接口是4线：TMS、TCK、TDI、TDO,分别为模式选择、时钟、数据输入和数据输出线。

JTAG在嵌入式系统开发过程中的作用

（1）硬件基本功能的测试（2）软件下载
（3）软件调试 （4）Flash烧写
当第一次写bootloader到flash 或只读存储器时，或者由于bootloader崩溃不能引导时，都需要使用JTAG电缆从主机PC上下载bootloader，并将bootloader写入flash。
JFlash 程序用于从主机PC上下载bootloader，并写入flash。

串行电缆

主要用于在host PC和嵌入式系统之间进行通讯。当主机需要向嵌入式系统发送命令时，当嵌入式系统需要向主机发送命令输出或者显示其状态时，使用串行电缆。
它的速度较慢，主要用于少量数据的传输。如果没有快速通讯设备时，也可以用来传输大量数据，如OS内核，文件系统等。

> *超级终端和**Minicom***
> *如何把开发板上的信息显示给开发人员？*
> *最常用的就是通过串口线输出到宿主机的显示器上。*
> *通过串口通信软件对串口进行配置，主要配置参数是波特率、数据位、停止位等。*
> *常用串口通信软件Windows下超级终端:Linux下minicom。*

以太网交叉电缆

串口电缆由于速度慢，主要用于握手，传送少量数据。若需要传送大量数据，通常采用以太网。

NFS

NFS 就是 Network FileSystem 的缩写，最早之前是由 Sun 所发展出来的。他最大的功能就是可以通过网络，让不同的机器、不同的操作系统、可以彼此分享文件。它可以让CLIENT计算机通过网络将远端的NFS SERVER共享出来的文件MOUNT到自己的系统中，在CLIENT看来使用NFS的远端文件就象是在使用本地文件一样。

NFS的启动和停止

为使NFS服务器正常工作，需启动portmap和nfs这两个服务，并且portmap一定要先于nfs启动。
在停止NFS服务时，顺序相反，一定要先停止nfs再停止portmap。

```
service portmap start
service nfs start
```

```
service nfs stop
service portmap stop
```

配置NFS服务器

NFS共享目录被列举在/etc/exports文件中，并且共享目录的访问权限、允许访问的主机等参数也在该文件中被定义。出于安全考虑，防止意外输出任何资源，该文件默认配置为空，即没有任何共享目录输出。

设定格式如下：

<输出目录> [客户端1 选项（访问权限，用户映射，其他）] [客户端2 选项（访问权限，用户映射，其他）]

输出目录为NFS目录，客户端为要连接到NFS的客户端，访问权限指客户端的操作权限，用户映射可以限制用户权限客户端常用的指定方式。

客户端

指定ip地址的主机 192.168.0.200

指定子网中的所有主机 192.168.0.0/24

指定域名的主机 a.liusuping.com

指定域中的所有主机 *.liusuping.com

所有主机 *

访问权限

设置输出目录只读 ro

设置输出目录读写 rw

用户映射

all_squash 将远程访问的所有普通用户及所属组都映射为匿名用户或用户组（nfsnobody）；

no_all_squash

与all_squash取反（默认设置）；

root_squash 将root用户及所属组都映射为匿名用户或用户组（默认设置）；

no_root_squash 与rootsquash取反；

anonuid=xxx 将远程访问的所有用户都映射为匿名用户，并指定该用户为本地用户（UID=xxx）；

anongid=xxx 将远程访问的所有用户组都映射为匿名用户组账户，并指定该匿名用户组账户为本地用户组账户（GID=xxx）；

其他

secure 限制客户端只能从小于1024的tcp/ip端口连接nfs服务器（默认设置）；

insecure 允许客户端从大于1024的tcp/ip端口连接服务器；

sync 将数据同步写入内存缓冲区与磁盘中，效率低，但可以保证数据的一致性；

async 将数据先保存在内存缓冲区中，必要时才写入磁盘；

wdelay 检查是否有相关的写操作，如果有则将这些写操作 一起执行，这样可以提高效率（默认设置）；

no_wdelay 若有写操作则立即执行，应与sync配合使用；

subtree 若输出目录是一个子目录，则nfs服务器将检查其父目录的权限(默认设置)；

no_subtree 即使输出目录是一个子目录,nfs服务器也不检查其父目录的权限,这样可以提高效率;

NFS服务配置实例

可以编辑/etc/exports为：

```
/tmp *(rw, no_root_squash)
/home/linux *.the9.com (rw, all_squash, anonuid=40, anongid=40)
```

> *在/etc/exports文件中，特别要注意“空格”的使用。*
>
> `/home client(rw)`
>
> *客户端client对/home目录具有读、写权限*
>
> `/home client (rw)`
>
> *client对/home目录只具有读权限（这是系统对所有客户端的默认值）。而除client之外的其他客户端对/home目录具有读、写权限。*

配置NFS客户端

在嵌入式目标系统的Linux Shell下，执行如下命令来进行NFS 共享目录挂载：

```
mount –t nfs 192.168.0.124:/home/work /mnt/nfs –o nolock
cd /mnt/nfs
ls
```

此时，嵌入式目标系统端所显示的内容即为Linux 服务器的输出目录的内容，即Linux 服务器的输出目录/home/work 通过NFS 映射到了嵌入式目标系统的/mnt/nfs 目录。

嵌入式系统软件结构与BootLoader

嵌入式系统软件分层

嵌入式系统从软件的角度可分4层：

1.引导加载程序。包括固化在固件(firmware)中的 boot 代码(可选)和 Boot Loader两大部分。

2.Linux 内核。特定于嵌入式板子的定制内核以及内核的启动参数。

3.文件系统。包括根文件系统和建立于 Flash 内存设备之上文件系统。

4.用户应用程序。特定于用户的应用程序。有时在用户应用程序和内核层之间可能还会包括一个嵌入式图形用户界面。

嵌入式Linux的组成

嵌入式Linux由以下三个部分组成为：

- Bootloader（引导加载器）
- 内核
- 根文件系统

嵌入式Linux的构建

在嵌入式Linux的构建中，Bootloader和Linux内核一般都有相对成熟的代码。主要的工作有两步：
第一步是根据本系统硬件平台的状况进行移植;
第二步是采取交叉编译对源代码进行编译，形成运行时需要的映象（Image）文件。

BootLoader

BootLoader位于最底层，是系统上电后运行的第一个程序，类似于PC机的Bios，主要负责整个硬件的初始化和软件启动的准备工作。
BootLoader就是在操作系统内核或用户应用程序运行之前运行的一段小程序。通过这段小程序，我们可以初始化硬件设备、建立内存空间的映射图，从而将系统的软硬件环境带到一个合适的状态，以便为最终调用操作系统内核或用户应用程序准备好正确的环境。

普通计算机引导过程
① 加电----打开电源开关，给主板和内部风扇供电。
② 启动引导程序----CPU开始执行存储在ROM BIOS中的指令。

③ 开机自检----计算机对系统的主要部件进行诊断测试。

④ 加载操作系统----计算机将操作系统文件从磁盘读到RAM中。

⑤ 检查配置文件，定制操作系统的运行环境----读取配置文件，根据用户的设置对操作系统进行定制。

⑥ 准备读取命令和数据----计算机等待用户输入命令和数据。

BootLoader的主要功能

初始化系统在启动阶段必需的硬件设备

准备后续软件系统运行所需的软件环境,例如复制一部分代码到RAM中.

向内核传递启动参数;

[可选]配置系统各种参数

[可选]支持各种协议来下载bootloader,内核,文件系统

[可选]在线烧写系统firmware,如启动参数,bootloader, 内核,文件系统等

[可选]支持在线调试

引导内核启动。

Bootloader安装媒介与执行

系统加电或复位后，所有的CPU通常都从CPU制造商预先安排的地址上取指令。比如，S3C44B0在复位的都从地址0x00000000取它的第一条指令。而嵌入式系统通常都有某种类型的固态存储设备（比如：ROM、EEPROM或FLASH等）被安排这个起始地址上，因此在系统加电后，CPU将首先执行BootLoader程序。也就是说对于基于S3C44B0的这套系统，BootLoader是从0地址开始存放的，而这块起始地址需要采用可引导的固态存储设备如FLASH。

控制**BootLoader**的设备或机制

串口通讯是最简单也是最廉价的一种双机通讯设备，所以往往在BootLoader中主机

和目标机之间都通过串口建立连接，BootLoader程序在执行时通常会通过串口来进行 I/O，比如：输出打印信息到串口，从串口读取用户控制字符等。

BootLoader的启动过程

多阶段的BootLoader能提供更为复杂的功能，以及更好的可移植性。从固态存储设备上启动的 BootLoader大多都是2阶段的启动过程，也即启动过程可以分为**stage 1**和**stage 2**两部分。

Stage 1

(1) 硬件设备初始化。

(2) 为加载Stage2程序准备RAM空间。

(3) 拷贝Stage2程序到RAM空间。

(4) 设置好堆栈。

(5) 跳转到Stage2的C程序入口点。

Stage 2

(1) 初始化本阶段用到的硬件设备，如RS-232。

(2) 检测系统内存映射。

(3) 将操作系统内核映像和根文件系统映像从Flash Memory读到RAM空间中。

(4) 为操作系统内核设置启动参数。

(5) 调用操作系统内核。

BootLoader的操作模式

启动加载模式(Boot loading)：这种模式也称为“自主”（Autonomous）模式，也即BootLoader从目标机上的某个固态存储设备上将操作系统加载到RAM中运行，整个过程并没有用户的介入。这种模式是BootLoader的正常工作模式。

下载模式(Downloading)：在这种模式下 目标机上的BootLoader将通过串口连接或网络连接等通信手段从主机下载文件，比如：下载应用程序、数据文件、内核

映像等．从主机下载的文件通常首先被BootLoader保存到目标机的RAM中然后再被BootLoader写到目标机上的固态存储设备中。Boot Loader 的下载模式通常在第一次安装内核与根文件系统时被使用；此外，以后的系统更新也会使用 Boot Loader 的下载模式。工作于下载模式下的 Boot Loader 通常都会向终端用户提供一个简单的命令行接口。

像Blob或U-Boot等这样功能强大的Bootloader通常同时支持这两种工作模式，而且允许用户在这两种工作模式之间进行切换。

常见的**BootLoader**

U-Boot

U-Boot是在PPC-Boot基础上进化而来的一个开放源码的BootROM程序，采用了高度模块化的编程方式。

U-Boot作为BootLoader，具备多种引导内核启动的方式。常用的go和bootm命令可以直接引导内核映像启动。U-Boot与内核的关系主要是内核启动过程中参数的传递。

U-Boot是“**Monitor**”(与只有引导功能的“BootLoader”相对)。除了Bootloader的系统引导功能，它还有用户命令接口，提供了一些复杂的调试、读写内存、烧写Flash、配置环境变量等功能。