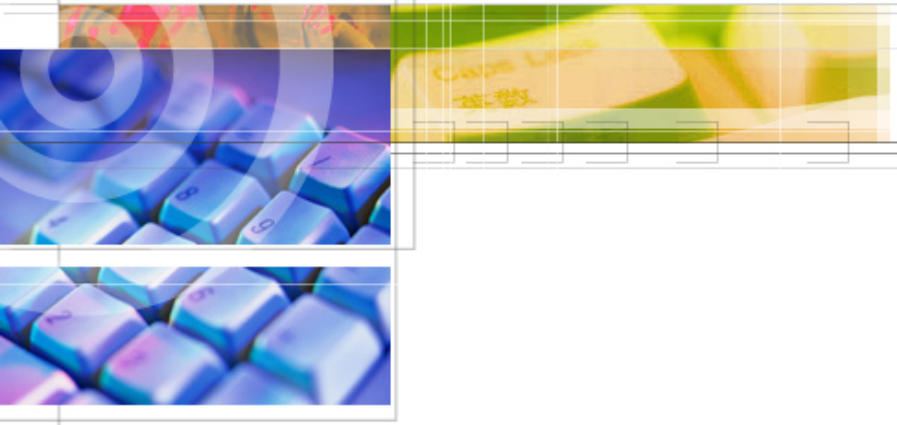


第四讲 嵌入式Linux系统移植

华中科技大学电信学院
鄢舒
E-mail: yan0shu@gmail.com

内容提纲(1/4)

- 1. Bootloader移植
- 2. 嵌入式Linux系统移植
- 3. Web Server应用程序移植
- 4. 实验内容与要求

1.1 嵌入式Linux系统软件分层

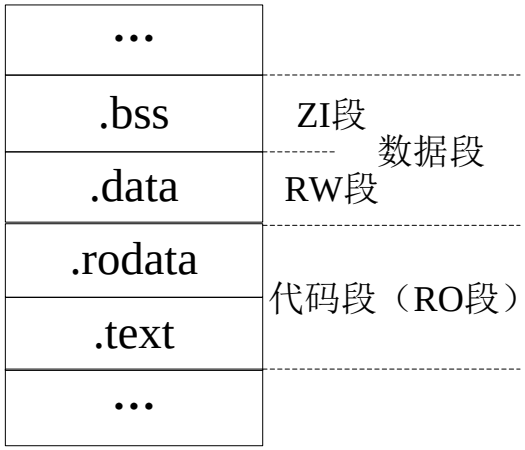
- 作为嵌入式系统软件的最底层，Bootloader是上电后启动运行的第一个程序，它类似于PC机上的BIOS程序功能，主要负责整个硬件系统的初始化和软件系统启动的准备工作。

1.2 可执行文件的组成

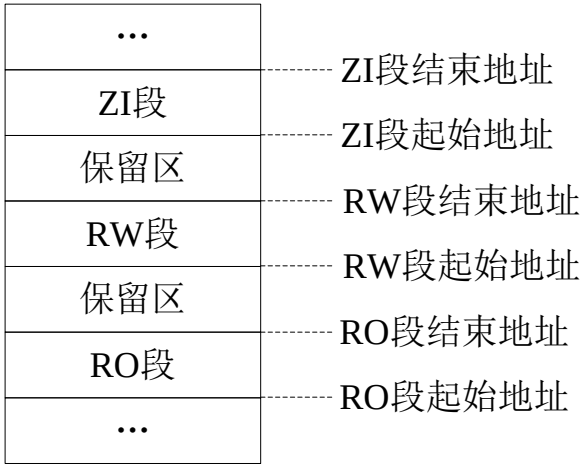
- 可执行文件一般由两部分组成：代码段和数据段
  - 代码段进一步分为：可执行代码段（.text）和只读数据段（.rodata）
  - 数据段进一步分为：初始化数据段（.data）和未初始化数据段（.bss）

1.3 可执行文件的运行态

- 可执行文件先是以存放态的形式存储在磁盘或flash芯片上，执行时通过装载过程搬入到RAM中运行，从而变成运行态（以ARM为例）。

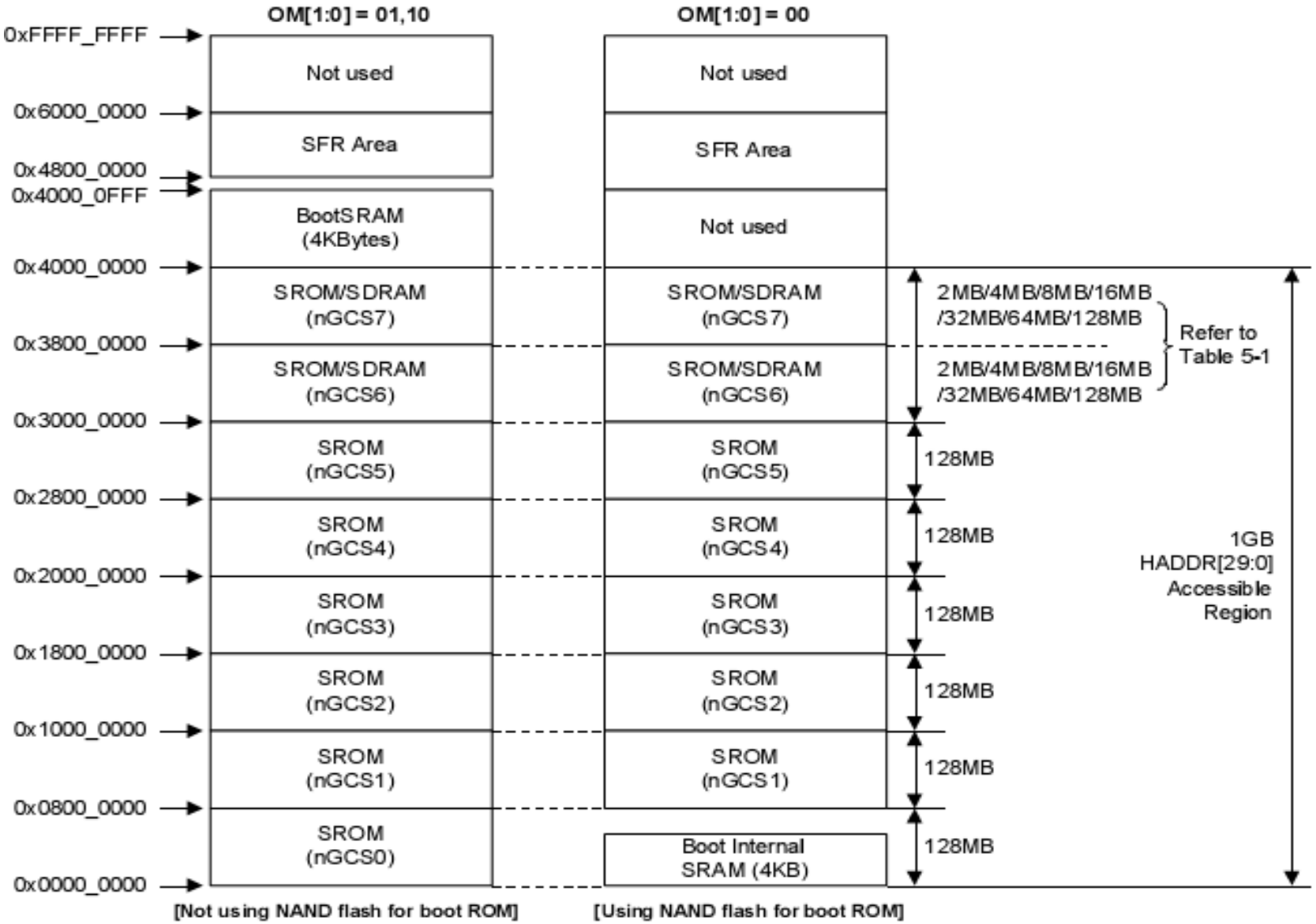


在存储器中的存放态



在RAM中的运行态

1.4 S3C2410X地址空间分配图



1.5 代码段地址指定

```
MYLOADER 0x30000000      ;MYLOADER: 为可执行文件的名称, 可自定义
                         ;0x3000000: 起始地址
{
  RO 0x30000000          ;RO 只读代码段的名称
                         ;0x30000000: 只读代码段的起始地址
  {
    init.o (Init, +First) ;Init代码段为可执行文件的第一部分。
    * (+RO)              ;所有其它的代码段和只读数据段放在该部分
  }
  RW +0                  ;RW: RW段的名称。+0: 表示RW段紧接着RO段
  {
    * (+RW)              ;所有RW段放在该部分
  }
  ZI +0                  ;ZI: ZI段的名称。+0: 表示ZI段紧接着RW段
  {
    *(+ZI)               ;所有ZI段放在该部分
  }
}
```

1.6 Bootloader的汇编部分

1. 建立中断向量异常表
2. 显式地切换到SVC且32位指令模式
3. 关闭内部看门狗
4. 禁止所有的中断
5. 配置系统时钟频率和总线频率
6. 设置内存区的控制寄存器
7. 初始化中断
8. 安装中断向量表
9. 把可执行文件的各个段搬到运行态的各个位置
10. 跳到C代码部分执行

1.7 Bootloader的C部分

1. 初始化MMU
2. 初始化外部端口
3. 中断处理程序表初始化
4. 串口初始化
5. 其它设备初始化（可选）
6. 主程序循环或跳到操作系统入口

1.8 U-Boot简介

- U-Boot是德国DENX小组的开发用于多种嵌入式CPU的bootloader程序, U-Boot不仅仅支持嵌入式Linux系统的引导，当前，它还支持NetBSD, VxWorks, QNX, RTEMS, ARTOS, LynxOS嵌入式操作系统。U-Boot除了支持PowerPC系列的处理器外，还能支持MIPS、 x86、ARM、NIOS、XScale等诸多常用系列的处理器。
  - ✓ 官方网站 http://www.denx.de和http://sourceforge.net/projects/u-boot
  - ✓ 开发套件ELDK（Embedded Linux Development Kit）

1.9 U-Boot的特点

1. 开放源码；
2. 支持多种嵌入式操作系统内核，如Linux、NetBSD, VxWorks, QNX, RTEMS, ARTOS, LynxOS；
3. 支持多个处理器系列，如PowerPC、ARM、x86、MIPS、XScale；
4. 较高的可靠性和稳定性；
5. 高度灵活的功能设置，适合U-Boot调试、操作系统不同引导要求、产品发布等；
6. 丰富的设备驱动源码，如串口、以太网、SDRAM、FLASH、LCD、NVRAM、EEPROM、RTC、键盘等；
7. 较为丰富的开发调试文档与强大的网络技术支持。

1.10 U-Boot主要目录结构

- board：目标板相关文件，主要包含SDRAM、FLASH驱动；
- common：独立于处理器体系结构的通用代码，如内存大小探测与故障检测；
- cpu：与处理器相关的文件。如mpc8xx子目录下含串口、网口、LCD驱动及中断初始化等文件；
- driver：通用设备驱动，如CFI FLASH驱动（目前对INTEL FLASH支持较好）
- doc： U-Boot的说明文档；
- include：U-Boot头文件；尤其configs子目录下与目标板相关的配置头文件是移植过程中经常要修改的文件；
- lib_xxx：处理器体系相关的文件，如lib_ppc, lib_arm目录分别包含与PowerPC、ARM体系结构相关的文件；
- net：与网络功能相关的文件目录，如bootp,nfs,tftp；
- post：上电自检文件目录。尚有待于进一步完善；
- rtc：RTC驱动程序；
- tools：用于创建U-Boot S-RECORD和BIN镜像文件的工具；

1.11 U-Boot可支持的主要功能

- 系统引导： 支持NFS挂载、RAMDISK（压缩或非压缩）形式的根文件系统；支持NFS挂载、从FLASH中引导压缩或非压缩系统内核；
- 基本辅助功能： 强大的操作系统接口功能；可灵活设置、传递多个关键参数给操作系统，适合系统在不同开发阶段的调试要求与产品发布，尤对Linux支持最为强劲；支持目标板环境参数多种存储方式，如FLASH、NVRAM、EEPROM；CRC32校验，可校验FLASH中内核、RAMDISK镜像文件是否完好；
- 设备驱动： 串口、SDRAM、FLASH、以太网、LCD、NVRAM、EEPROM、键盘、USB、PCMCIA、PCI、RTC等驱动支持；
- 上电自检功能 SDRAM、FLASH大小自动检测；SDRAM故障检测；CPU型号；
- 特殊功能： XIP内核引导。

1.12 U-Boot移植的主要步骤(1/3)

以S3C2410处理器为例，说明u-boot的主要移植步骤：

1.修改Makefile文件

```
bks2410_config : unconfig
    @./mkconfig $(@:_config=) arm arm920t bks2410 NULL s3c24x0
```

各项的意思如下:

arm: CPU的架构(ARCH)

arm920t: CPU的类型(CPU)，其对应于cpu/arm920t子目录。

bks2410: 开发板的型号(BOARD)，对应于board/bks2410目录。

NULL: 开发者/或经销商(vender)。

s3c24x0: 片上系统(SOC)。

1.12 U-Boot移植的主要步骤(2/3)

2. 建立board/bks2410目录，拷贝board/smdk2410下的文件到board/bks2410目录，将smdk2410.c更名为bks2410.c
3. cp include/configs/smdk2410.h include/configs/bks2410.h
4. 将arm-linux-gcc的目录加入到PATH环境变量中
5. 测试编译能否成功：
   make bks2410_config
   make all ARCH=arm
   生成u-boot.bin就OK了

1.12 U-Boot移植的主要步骤(3/3)

6. 依照你自己开发板的内存地址分配情况修改board/bks2410/memsetup.S文件
7. 在board/bks2410加入NAND Flash读函数，建立nand_read.c 。
8. 修改board/bks2410/Makefile
9. 修改cpu/arm920t/start.S文件
10. 修改include/configs/bks2410.h文件
11. 重新编译u-boot
    make all ARCH=arm
12. 通过jtag将u-boot烧写到flash

内容提纲(2/4)

- 1. Bootloader移植
- 2. 嵌入式Linux系统移植
- 3. Web Server应用程序移植
- 4. 实验内容与要求

2.1 Linux Kernel 2.6移植过程（1/3）

1. 下载Linux内核源代码
2. 修改Makefile文件
   - ARCH?=arm
   - CROSS_COMPILE?=arm-linux-
3. 设置flash分区
   - 建立Nand Flash分区表
   - 加入Nand Flash分区
   - 建立Nand Flash芯片支持
   - 加入Nand Flash芯片支持到Nand Flash驱动
   - 指定启动时初始化Nand Flash及分区设置

2.1 Linux Kernel 2.6移植过程（2/3）

建立Nand Flash分区表，如某个Nand Flash总共64MB, 按如下大小进行分区：

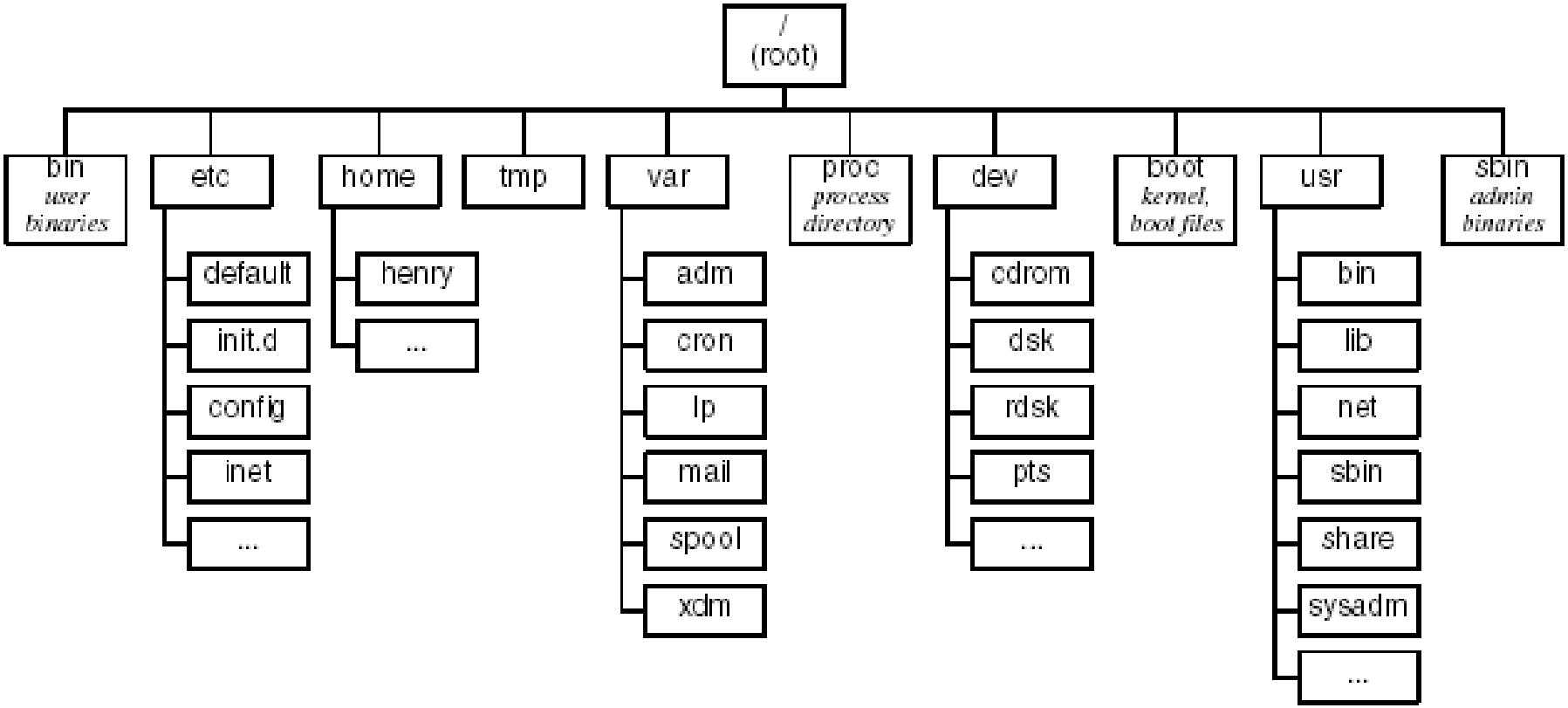
```
static struct mtd_partition partition_info[] ={
    { /* 1MB */
    name: "bootloader",
    size: 0x00100000,
    offset: 0x0,
    },{ /* 3MB */
    name: "kernel",
    size: 0x00300000,
    offset: 0x00100000,
    }, { /* 40MB */
    name: "root",
    size: 0x02800000,
    offset: 0x00400000,
    }, { /* 20MB */
    name: "user",
    size: 0x00f00000,
    offset: 0x02d00000,
   }
};
```

name: 代表分区名字 size: 代表flash分区大小 offset: 代表flash分区的起始地址

2.1 Linux Kernel 2.6移植过程（3/3）

4. 配置内核
   - #cp arch/arm/configs/smdk2410_defconfig .config
   - #make menuconfig
5. 编译内核
   - #make zImage
6. 下载zImage到目标系统
   - 下载到SDRAM中
   - 烧写到Flash中
7. 启动目标系统

2.2 Linux根文件系统的基本结构(1/2)



2.2 Linux根文件系统的基本结构(2/2)

目 录	内 容
bin	必要的用户命令（二进制文件）
boot	引导加载程序使用的静态文件
dev	设备文件和其他特殊文件
etc	系统配置文件，包括启动文件
home	用户主目录，包括供服务帐号所使用的主目录，如FTP
lib	必要的链接库，如C链接库、内核模块
mnt	安装点，用于暂时安装文件系统
opt	附加的软件套件
proc	用来提供内核与进程信息的虚拟文件系统
root	root用户的主目录
sbin	必要的系统管理员命令（二进制文件）
tmp	暂时性文件
usr	在下层包含对大多数用户都有用的大量应用程序和文件
var	监控程序和工具程序所存放的可变数据

2.3 根文件系统的移植过程（1/3）

1. 建立空根目录
   - #mkdir my_rootfs
2. 在该目录中建立Linux目录树
3. 创建linuxrc文件
   - 挂载/etc为ramfs, 并从/mnt/etc下拷贝文件到/etc目录当中
   - 重新创建/etc/mtab设备入口
   - 挂载/dev/shm为tmpfs文件系统
   - 挂载/proc为proc文件系统
   - 挂载/sys为sysfs文件系统
4. 修改目录和文件的权限

2.3 根文件系统的移植过程（2/3）

5. 移植Busybox
   - 下载Busybox源代码
   - 配置Busybox

     #make menuconfig
   - 编译并安装Busybox

     #make TARGET_ARCH=arm CROSS=arm-linux- \

     PREFIX=…/my_rootfs/ all install
6. 创建相关配置文件
   - /etc/profile文件：设置用户路径和动态库搜索路径
   - /etc/fstab文件：挂载文件系统信息

2.3 根文件系统的移植过程（3/3）

7. 建立根目录文件系统包

- 下载cramfs工具
- 制作cramfs包

  #mkcramfs my_rootfs my_rootfs.cramfs
- 下载cramfs包到目标系统
  - ✓ 下载到SDRAM中
  - ✓ 烧写到Flash中

内容提纲(3/4)

- 1. Bootloader移植
- 2. 嵌入式Linux系统移植
- 3. Web Server应用程序移植
- 4. 实验内容与要求

3.1 应用程序移植的步骤

- 准备源代码

  一般需要解压软件包，安装源代码。

- 编译源代码

  需要生成或修改Makefile文件，必要时需要修改源代码。

- 安装应用程序

  拷贝到目标板上，可能还需要配置文件和其他必要的文件。

- 运行应用程序

  直接在目标板上运行或修改系统配置文件实现系统启动时自动运行。

3.2 Web Server应用程序

- 当前用于嵌入式平台的Web Server有多种，如商业的有Blunk Microsystems的TargetWeb™，Mbedthis的AppWeb HTTP Server以及McObject的eXtremeWS™等，开源的有GoAhead Software Inc的 GoAhead，此外还有 boa、httpd、thttpd等。

3.3 boa介绍

- boa是一款轻量级的web server，作为一个单任务的http服务器，可以支持所有类UNIX系统，源代码开放、支持认证、CGI，性能高，满足GPL协议。
- 不支持ASP、SSL等高级功能，而且更新很缓慢，目前最新版本0.94.13。
- 官方网站：http://www.boa.org/

3.4 boa的移植步骤（1/3）

（1）准备源代码

到boa网站http://www.boa.org下载boa-0.94.13.tar.gz源代码。

解压软件包，安装源代码。

tar xzvf boa-0.94-13.tar.gz

（2）编译源代码

进入src目录，编译源代码。

解压后src目录下有Makefile.in文件，但没有Makefile文件，为了编译源代码，需要先生成Makefile文件，在src目录下运行configure命令即可。

#./configure

3.4 boa的移植步骤（2/3）

（3）修改Makefile文件

由于生成的Makefile文件是针对X86平台的，为了生成能够在ARM上运行的boa，需要修改Makefile文件。

将Makefile中的下面内容：

CC = gcc

CPP = gcc –E

改为（即采用交叉编译工具）：

CC = armv4l-unknown-linux-gcc

CPP = armv4l-unknown-linux-gcc –E

然后输入make命令进行编译，在src目录下就会生成boa文件。

```
# make
```

3.4 boa的移植步骤（3/3）

（4）配置boa服务器

boa启动时需要一个配置文件boa.conf，该文件的缺省目录由src/defines.h文件的SERVER_ROOT定义，或者在启动boa的时候通过参数“-c”指定。其中指定的默认目录是：/etc/boa/

（5）登录boa服务器

在PC机浏览器地址栏输入目标系统的IP地址，访问存在于目标系统中的网页。

http://192.168.0.11

3.5 GoAhead WebServer介绍

- GoAhead是GoAhead公司的Embedded Management Framework产品的一部分，这个软件包主要用于解决未来嵌入式系统开发的相关问题。
- GoAhead WebServer的主要特性有：
  - 支持ASP
  - 嵌入式的JavaScript
  - 标准的CGI执行
  - 内存中的CGI处理GoFroms
  - 扩展的API
  - 快速响应，每秒可处理超过50个请求
  - 完全和标准兼容
  - 如果不包含SSI，仅要求60K的内存；包含SSI，要求500K内存
  - web页面可以存在ROM或文件系统中
  - 支持多种操作系统，包括eCos 、LINUX 、LynxOS 、QNX 、VxWorks 、WinCE、pSOS等

3.6 GoAhead的移植

- 解压源代码包

  # tar xzvf goahead218.tar.gz

- 修改LINUX/Makefile文件

  在开头（注释之后）加入以下内容：

  CC=armv4l-unknown-linux-gcc

  AR=armv4l-unknown-linux-ar

- 在LINUX目录下编译

  #make

- 安装到目标板上

  拷贝可执行文件webs和web子目录到指定位置。

内容提纲(4/4)

- 1. Bootloader移植
- 2. 嵌入式Linux系统移植
- 3. Web Server应用程序移植
- 4. 实验内容与要求

4.1 本次实验要求

- 学习嵌入式Linux系统移植的基本方法；
- 在实验平台上构建http服务器：
  - 在实验平台上移植一种web server应用程序，实现在PC机上用浏览器打开实验平台上的自己编写的动态网页。
- 有能力可选编译移植Linux内核或Bootloader；
- 利用NFS方法调试、运行。

4.2 实验注意事项

- boa只支持CGI，而GoAhead还支持ASP，但注意需要仿照webs目录中asp例子来编写；
- boa可以通过命令行参数来设定网页目录的位置的，而GoAhead中是直接写入main.c文件中，需要修改后再重新编译；
- 如果希望实验平台上电后自动运行web server需要把可执行文件和网页都拷贝到实验平台的Flash可写分区/usr上，而且还要修改相应的配置文件。