《操作系统》综合课程设计指导

实时嵌入式
Linux 系统关键技术开发

北京邮电大学计算机学院

2016 年 7 月

目 录

1. 引言

综合课程设计是培养学生专业实践能力的一项重要手段。北京邮电大学学院以培养计算机通信人才为目标，《操作系统》作为专业基础课，其综合课程设计应体现出面向通信领域的特色。

实时嵌入式系统广泛应用于通信领域，实时嵌入式操作系统是通信领域中的核心基础软件。OS 操作系统知识领域为

- OS1 操作系统概述
- OS2 操作系统原理
- OS3 并发性
- OS4 调度和分派
- OS5 内存管理
- OS6 设备管理
- OS7 安全与保护
- OS8 文件系统
- **OS9 实时和嵌入式系统**
- OS10 容错（选修）
- OS11 系统性能评价（选修）
- OS12 脚本（选修）

本课程设计选取**面向通信领域的“实时嵌入式 Linux 系统关键技术开发”**作为与《操作系统》课程教学相配套的综合课程设计，培养学生对实时嵌入式操作系统的分析、设计和实现技能，掌握非核心知识单元“OS9 实时和嵌入式系统”的相关知识点。

虽然与 Windows 等常规操作系统相比，实时嵌入式操作系统在功能和结构上相对简单。但在开放源代码 Linux 基础上开发实时嵌入式操作系统，需要在对 Linux 系统结构深入分析基础上，对 Linux 的各项资源管理功能作一系列改进，工作量仍然非常大。

考虑到综合课程设计学时数和学生参与人数的限制，以及综合课程设计的目的并非开发一个完备实用的实时嵌入式操作系统，因此只选取开发实时嵌入式 Linux 系统时的部分关键技术作为《操作系统综合课程设计》的实验内容。

2. 实时嵌入式系统与实时嵌入式操作系统

- 关于实时嵌入式系统和实时嵌入式操作系统的概念、原理核开发等具体内容可参见课件“《操作系统》- 教程 - bupt-叶文- Lcture 12 Embedded and Real-time Linux”。

  本实验采用基于 SkyEye 的模拟 *ARM* 嵌入式硬件平台，操作系统为 ucLinux 嵌入式操作系统。

- rpm 软件包管理器下载、安装、使用

  实验需要用到一系列开源软件和工具，这些软件和工具是以源码方式发布，或以 rpm 软件包方式发布的。对以 rpm 软件包方式发布的软件，需要使用 rpm 软件包管理器，才能正确安装该软件。

  在 Red Hat 的发行版本光盘中带有 rpm 软件包管理器软件，也可从网上下载此软件。具体可参见“附录 1——RPM 软件包管理器的下载与升级”

3. 嵌入式系统软件开发

3.1. 嵌入式系统开发平台

通常嵌入式软件开发工具包括：编辑器、编译器、汇编器、链接器、调试器、工程管理及库函数等各种软件工具，很多厂商提供了将这些工具综合在一起的集成开发环境(IDE)。下面就常见的各种支持 ARM 和 uClinux 软件开发工具做概括性的介绍。

在 uClinux 平台下,目前最流行的开发工具是 GNU 开发套件。

GNU 开发套件作为通用的 Linux 开放套件，包括一系列的开发调试工具撇。主要组件包括：

(1) gcc ： 编译器，可以做成交叉编译的形式，即在宿主机上开发编译目标器件上可运行的二进制文件。

(2) binutils：一些辅助工具，包括 objdump(可以反编译二进制文件)，as(汇编编译器)，ld(链接器)等等。

(3) gdb： 调试器，可使用多种交叉调试方式，gdb-bdm(后台调试工具)，gdbserver(使用以太网络的调试工具)。

(4) 交叉编译器运行在某种处理器上，却可以编译另一种处理器的指令。支持一种新的处理器，必须具备一些编译，汇编工具，使用这些工具可以形成可运行于这种处理器的二进制文件。对于内核使用的编译工具同应用程序使用的有所不同。

3.2 嵌入式系统开发环境的建立

由于嵌入式系统是一个受资源限制的系统，故而直接在嵌入式系统硬件上进行编程显然是不合理的。在嵌入式系统的开发过程中，一般采用的方法是先在通用 PC 上编程，然后通过交叉编译连接，将程序做成目标平台上可以运行的二进制代码格式。最后下载到嵌入式系统。

- **交叉编译环境**

一个高级语言往往需要在不同的目标机上实现，这就产生了一个问题：如何把已在某机器上实现的一个高级语言的编译程序能否移植到另一个目标机上。通常把某个机器(称为宿主机)上已有的软件移植到另一台机器(称为目标机)上的过程称为移植。在移植过程中也常会用到交叉编译的技术。所谓交叉编译是指把一个源语言在宿主机上经过编译产生目标机的汇编语言或机器语言。

交叉编译所产生目标机的汇编语言或机器语言在宿主机上是不能运行，只能在目标机上运行，因此，程序调试比较麻烦。

- **交叉编译器的安装**

在嵌入式系统中，由一个源文件变成最终可执行的二进制文件，要经过三个过程，即编译，链接和重新定位，通过编译或者汇编工具，将源代码变成目标文件，由于目标文件往往不止一个，所以还需要链接工具将它们链接成另外一个目标文件，可以称其为“可重定位程序”。结果定址工具，将“可重定位程序”变成最终可执行文件。

一般的嵌入式系统应用程序的开发，通常采用的是主从模式，通过串口或者网口，使目标机和宿主机相连接。通常来说，编译器，连接器和定址器都是在宿主机上（一般是 pc 机，对于嵌入式开发而言，还都是运行 Linux 操作系统的 pc）运行的，而最终经过编译－链接－重新定位所得到的二进制可执行文件却都是在目标机上运行的，所以我们把这种编译过程称为“交叉编译”。

要在 Linux 下建立 ARM 的交叉编译环境一般来说比较麻烦。下面我们通过一个简单的方法来安装交叉编译器。

首先，下载 arm-elf-tools-xxxxxxxx.sh（xxxxxxxx 是版本号），这是一个 shell 脚本文件，执行它即可完成交叉编译器的安装。然后执行：

```
#./arm-elf-tools-xxxxxxxx.sh
```

这样，交叉编译环境即配置完成，下面就可以在桌面版的的 Linux 上编译 ARM 处

理器的目标代码了。

3.3 嵌入式系统软件开发步骤

嵌入式软件的编写和开发调试的主要流程为:编写－交叉编译－交叉连接－重定位下载－调试，如图 Figure 3.1。

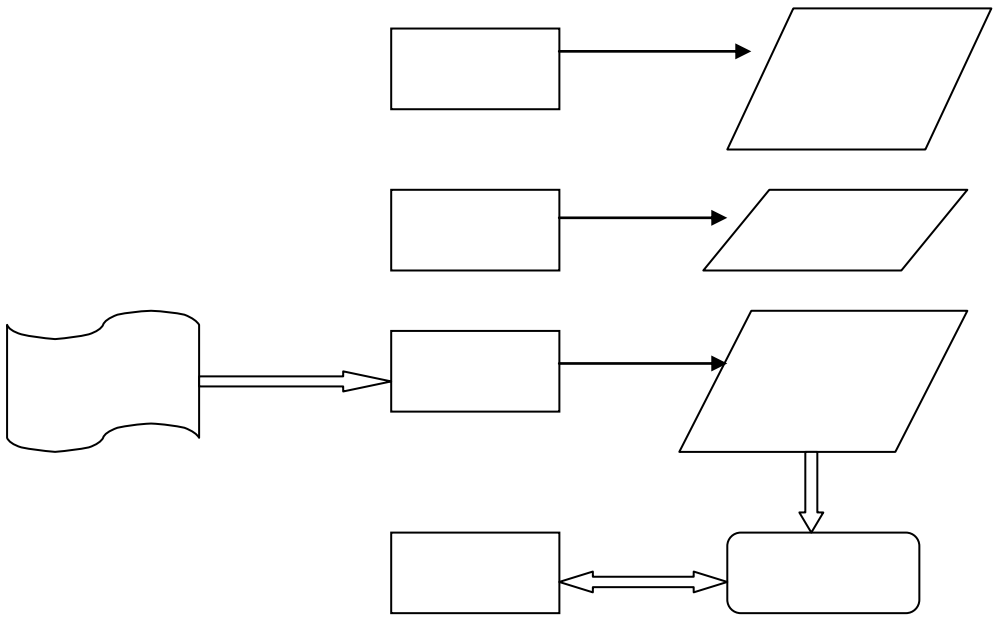


Figure 3.1 嵌入式开发流程图

3.4 嵌入式系统软件开发工具

- **Make**

make 是 GUN 的项目管理工具。make 自动分析决定在所有的项目文件中，哪些需要重新编译，并且生成编译所要使用命令。make 围绕着项目的 Makefile 文件展开工作。

Makefile 的主要工作是指示 make 如何编译和链接程序。此外 make 还通过 Makefile 来处理不同的用户输入命令如 make clean 等。

当 make 重建整个项目时，只有项目中自上次编译以来改变过的文件被重新编译；如果某个头文件发生了变化，每个包含了这个头文件的源文件将会被重编译。编译过程中会产生目标二进制文件。最终，如果有任何源文件在此次 make 中被重新编译，所有

的目标文件，包括本次 make 更新的和以前 make 未改变的，将会被重新链接生成新的目标项目文件。Makefile 主体由一系列的“规则”组成。具体的语法在这里不再叙述。

- **GDB**

gdb 是 GNU 的功能强大的调试器。在开发过程中常用的命令选项及含义如图 Figure 3.2 所示：

选项	含义
run	在 gdb 开始执行某个程序
set	设置程序入口参数
break	设置断点
watch	设置观察点
info	察看状态
delete	断点、观察点删除
continue	继续执行程序
step	单步一行源代码（进入子函数调用）
next	单步一行源代码（不进入子函数调用）
finish	完成当前栈帧
until	执行到指定位置
frame	检查当前调用栈情况
list	显示源文件内容
disassemble	反编译指定地址内容
file	指定调试程序

Figure 3.2 GDB 命令选项

- **gcc 和 ld**

选项	含义
-c	仅进行编译，不链接
-o	指定文件输出名
-pipe	使用管道而不是临时文件进行通信
-fio-builtin	不使用编译器内部函数
-W	警告控制
-g	编译时加入调试信息
-o	指定编译时的优化级别
-i	指定头文件
-nostdinc	不在系统标准的头文件目录进行头文件搜索
-l	指定链接时的库文件
-nostdlib	链接时不使用默认的启动文件和库文件
-static	静态链接
-Xlinker	向链接器增加选项
-I	指定头文件搜索目录
-L	指定库文件搜索目录
-mtune	指定 arm 目标处理器名称

-mapcs-frame	产生符合 arm 函数调用规范的栈帧

Figure 3.3 GCC 命令选项

gcc 是 GNU 的编译器，多数情况下 gcc 会在内部调常用的命令选项及含义如图 Figure 3.3 所示。

4. uclinux/ARM 实验环境与 SkyEye 软件平台

实时嵌入式 Linux 内核的运行平台为嵌入式 CPU 系统（如 ARM 系列），受实验条件限制无法搭建一个真正的嵌入式 CPU 硬件平台，因此采用现有的一些嵌入式 CPU 软件模拟器替代。

本课程设计利用 uclinux/ARM 作为实验平台，其中的 uclinux/ARM 嵌入式 CPU 及其外围 I/O 设备（如网卡等）用 SkyEye 硬件模拟平台来代替。

SkyEye 是一个开源软件（OpenSource Software）项目，它的目的是在通用的 Linux 和 Windows 平台上实现一个纯软件集成开发环境，模拟常见的嵌入式计算机系统。可在 SkyEye 上运行 Linux/uClinux 以及 uC/OS-II 等多种嵌入式操作系统和各种测试软件，并可以对它们进行源码级的分析和测试。

作为一个指令级硬件模拟平台，SkyEye 可以模拟多种嵌入式开发板，可支持多种 CPU 指令集，在 SkyEye 上运行的操作系统意识不到它是在一个虚拟的环境中运行，而且开发人员可以通过 SkyEye 调试操作系统和系统软件。

- **参考书籍**

  陈渝，李明，杨晔，《源码开放的嵌入式系统软件分析与实践—基于 SkyEye 和 ARM 开发平台》，北京航空航天大学出版社，2004 年。

- 项目网址：

a. http：//www.skyeye.org/

b. http：//www.skyeye.org/index_cn.html

c. 中国 Linux 公社论坛 首页 -> Skyeye 项目专栏：
http: //www.linuxfans.org/nuke/modules.php?name=Forums&file=viewtopic&t=99661

d. SkyEye FTP 主站点：
http：//gro.clinux.org/project/showfiles.php?group_id=327&release_id=487

e.协同开发站点：

gro.clinux.org 提供 SkyEye SourceForge 站点,主要用于分布式协同开发

- 功能介绍：

a. SkyEye 目前支持的模拟硬件：

CPU 核系列： ARM7TDMI，ARM720T, ARM9，StrongARM，Xscale

CPU 系列： Atmel AT91/X40， Cirrus CIRRUS LOGIC EP7312, StrongARM SA1100/SA1110, Intel XScale PXA 250/255，CS89712，samsung 4510B，samsung 44B0(还不全)

b. 外设系列：

Timer, UART, ne2k 网络芯片，LCD，Touch Screen 等的模拟

c. 目前在 SkyEye 上运行的操作系统和系统软件包括：

uC/OSII-2.5.x with network support

uClinux（基于 Linux2.4.x）with network support

ARM Linux 2.4.x /2.6.x

lwIP on uC/OS II

- SkyEye 的安装和使用

SkyEye 运行于 Linux 操作系统下，以在 RedHat9.0 环境的 SkyEye 安装为例：

（1）下载 SkyEye 的软件包 skyeye-x.x.x.src.tar.bz2(x.x.x 代表 SkyEye 的版本号)。

（2）解压源码包（例如解到“/embed”目录下），执行如下命令：

```
# cd /embed
# tar –xjvf skyeye-x.x.x.src.tar.bz2
```

（3）进入解压 SkyEye 目录，配置 SkyEye

```
# cd skyeye
# ./configure –target=arm-elf –prefix=/usr/local
```

（4）编译和安装

```
# make
# make install
```

正确执行完上面的指令且命令正常结束后，系统中的/usr/local/bin/skyeye 执行程序就是安装好的 SkyEye 软件。

5. 实验内容

5.1 Linux 启动过程优化

实验目的：

Linux 系统从启动到登录 shell 界面需要花费较长时间，在普通微机上的启动过程需要十几秒或更长。如果要启动 X 界面，那花费的时间就更多了。启动时间过长对嵌入式系统而言，如信息家电产品（机顶盒），是无法接受的。

Linux 系统的启动由内核态下的启动和用户态下的启动组成.。利用本实验中采取的多种方法，可以简化 Linux 系统启动过程，提高 Linux 系统启动速度，适应嵌入式系统快速启动和实时应用的需要。

实验环境：

硬件：Intel i586 PC机

软件：Red Hat Linux7.1 (Linux内核版本为2.4.2) 或更高版本

说明：可以根据需要，选择其它Linux版本

实验内容与步骤：

a. 分析原代码，理解启动流程（参见 图 5.1 ）

b. 在内核态下，利用 do_gettimeofday()函数测试时钟初始化完成后，系统启动过程中几个主要部分的启动时间，如 start_kernel()中的控制台 console_init()、inti()函数中的文件系统初始化和文件系统加载，寻找出初始化时间较长的内核部分。

c. 要求：至少选取启动过程中的 5 处较大的代码模块，测量其执行时间，保证测出的执行时间可观测

d. 内核态进程启动过程优化-IDE 检测修改

- 修改 linux/include/asm-I386/ide.h 中宏定义#define MAX_HWIFS 10，减少内核启动时检测的端口数目，提高启动速度。
- 编写测试函数，测试实验前后 IDE 接口检测的时间

e. 用户态进程启动过程优化——开机画面字符隐藏技术

用户态进程启动过程优化可以参考图 5.1，通过配置/etc/rc.d/rc.sysinit /etc/rc.d/rc 两个 Shell 脚本中的相关选项来实现。

Linux 启动时，显示器上会出现一行行字符用于提示当前系统启动进度。与启动的时间一样，这些字符信息对于家电、PDA、智能手机等嵌入式系统的用户而言是不必要的。可以利用重定向技术，实现开机字符信息的隐藏。

方法：

- 开机过程中，在内核态进程启动过程中，通过配置 GRUB 参数在 menu.lst 或者/etc/grub 中加入 console=/dev/tty2 CONSOLE=/dev/tty2 便可隐藏启动信息。
- 如果系统使用 lilo 来启动内核，则在/etc/lilo.conf 文件中为内核添加启动参数，即在内核项里添加如下一行

append = “console=/dev/tty2 CONSOLE=/dev/tty2 ”;

默认情况下开机信息在 tty1 上显示，而此参数的作用是将 tty1 的字符信息转到 tty2 上，故而隐藏了开机字符信息。如果用户需要察看字符信息，可通过 Alt+F2 切换到 tty2 察看。

f. 用户态进程启动过程优化—— 开机画面隐藏/调出

Linux 系统必备的内核进程启动完毕后，可以把开机画面调出来，原理如下。

Linux 系统通过参考/etc/sysconfig/init 中的 BOOTUP 和 GRAPHICAL 来决定是不是采用开机画面。可以通过控制这部分的相关代码实现开机画面的隐藏和调出。在/etc/init.d/rc.sysinit 中相关代码如下

```
# Start the graphical boot, if necessary; /usr may not be mounted yet, so we
# may have to do this again after mounting
RHGB_STARTED=0
mount -n /dev/pts
if    strstr "$cmdline" rhgb && ! strstr "$cmdline" early-login && [ "$BOOTUP" =
"color" -a "$GRAPHICAL" = "yes" -a -x /usr/bin/rhgb ]; then
LC_MESSAGES= /usr/bin/rhgb
RHGB_STARTED=1
```

上述代码中，通过 BOOTUP=color 和 GRAPHICAL = yes 来控制系统的行为 RHGB_STARTED 为 RHGB 状态参数。

在条件满足的情况下，将 rhgb 传给 LC_MESSAGES 即可。

g. 用户态进程启动过程优化—— 开机画面更换

Linux系统利用底层图形基础设施Frame Buffer显示图片。Frame Buffer是一种驱动程序接口，它将显示设备抽象为帧缓冲区，因此Frame Buffer可以看作显存的一

个映像，映射到地址空间。

启动画面是指系统每次进入桌面环境时出现的小幅画面，它表明系统正在加载程序。在GNOME环境下修改/替换启动画面的原理为用一幅大小适中、文件格式为PNG的图片，假定图片名称为modified.png，用下述命令替换掉**/usr/share/pixmaps/splash/gnome-splash.png**：

#cp modified.png /usr/share/pix-maps/splash/gnome-splash.png

h. 经过上述内核态和用户态优化后，测试系统总的启动时间

i. 要求：

建议用小组成员的相片作为替换的开机画面

实验指导：

1. 关于显示用户定义的画面

Linux过底层图形基础设施Frame Buffer来显示图片。 Frame Buffer是出现在内核版本2.2以后的一种驱动程序接口。这种接口将显示设备抽象为帧缓冲区。用户可以将它看成是显存的一个映像，将其映射到进程地址空间之后就可以直接进行读写操作，其中写撽作可以立即反应在屏幕上。该驱动程序的设备文件一般是/dev/fb0、/dev/fb1等。内核启动初始化Frame Buffer之后，利用Frame Buffer显示图像。

硬件的基本要求为：需要一块至少可以显示640×480像素和256色的VSA兼容图形显示卡（现在的显卡一般都满足这个要求）。

由于图片要在一开机就显示的，所以肯定在Linux Kernel的启动部分有一段代码将用户准备的图片数I通过Frame Buffer显示在屏幕上。这界牵涉到一个问题就是用户设计好的图片如何转换成一定格式的数据，并编译链接到kernel映像里去。Kernel里面用来显示图片的那段代码必须知道需要显示图片数据的格式。至于如何将图片数据编译到kernel映像里面去，是将图片数据以数组的方式放到一个头文件drivers/video/linux_logo.h中；而该头文件被头文件include/asm/linux_logo.h包括。在C源文件drivers/video/fbcon.c中引用。从而，在编译内核的时候可以将图片数据链接到kernel映像中去。

实现用户开机画面的替换需要解决2个问题：

（1） 如何将用户设计好的图片转换成linux_logo.h文件？

（2） 如何修改内核代码来根据linux_logo.h里面的图片数据以便显示图片？

目前支持两种图片数据格式：.pcf和.tif，图片颜色为256色，大小是640×480像素。

当使用 Red Hat7.1 时，开机启动时在屏幕的左上角有一个 80×80 像素的小企鹅图片。这说明内该是能显示图片的代码的。这个小企鹅图片的数据就存放在 linux_logo.h

文件里面。说明内核中已经存在图片显示代码。利用内核原有的显示图片的代码，稍作改动就可以显示 640×480 像素的图片。内核显示图片的原理是利用 Frame Budffer 来实现的。

下面需要解决将pcx和tif格式的图片转换成linux_logo.h所使用的图片格式的数据。

由于内核所便用的图片格式是按照图片标准的，所以可以使用通用图片格式转换工具。可以使用Cajus<C.Pollmeier@gmx.net>为他的lpp项目所写的两个图片格式转换工具fblogo和boot_logo。其中fblogo用于转换tif格式的图片，boot_logo用于转换pcx格式的图片。这2个程序都很方便，只要后面的参数是要住转换的图片的路径和名字就可以了。

下面需要解决将pcx和tif格式的图片转换成linux_logo.h所使用的图片格式的数据。

由于内核所便用的图片格式是按照图片标准的，所以可以使用通用图片格式转换工具。可以使用Cajus<C.Pollmeier@gmx.net>为他的lpp项目所写的两个图片格式转换工具fblogo和boot_logo。其中fblogo用于转换tif格式的图片，boot_logo用于转换pcx格式的图片。这2个程序都很方便，只要后面的参数是要住转换的图片的路径和名字就可以了。

现在假设已经将一张需要使用的图片转换成了linux_logo.h头文件，那就用这个linux_logo.h替换文件driver/video/linux_logo.h。在替换之前，由于由于修改的内核代码还朋要对转换后的linux_logo.h做一些改动，为此写了一个shell脚本，自动对linux_logo.h做修改，并将修改后的linux_logo.h文件安装到源代码目录中去替换头文件driver/video/linux_logo.h。

最后就是最关键的对源代码进行修改。其实，对源代码的修改非常少，只涉及一个文件drivers/video/fbcon.c中的一个宏和一个函数。

原来的宏定义为：

```
#define LOGO_H    80
#define LOGO_W    80
```

将其修改为

```
#define LOGO_H    640
#define LOGO_W    480
```

修改函数：

```
static    int    int    fbconshow_logo(void ) {
  ….
for ( x=0 ; x < smp_num_cpus*(LOGO_W + 8)&&)
//  将上面这一行代码用下面代码替换掉
// for (x=0; x < (LOGO_W + 8) $$
```

…
，完成对内恢的修改。

内核使用Frame Buffer显示图片，所以在内核启动的时需要给内核传速Frame Buffe的图形模式参数。图片是640×480像素，256色，则VGA的参数应该是：vga=0x301。所以应在lilo.conf里面加一行

vga=0x301

最后执行命令lilo使对/etc/lilo.conf的修改生效。

2． Linux 的内核态和用户态的启动过程

Linux kernel2.4.2 内核态和用户态启动过程参见图 5.1，具体解释参见“附录 2——Linux 的内核态和用户态的启动过程”

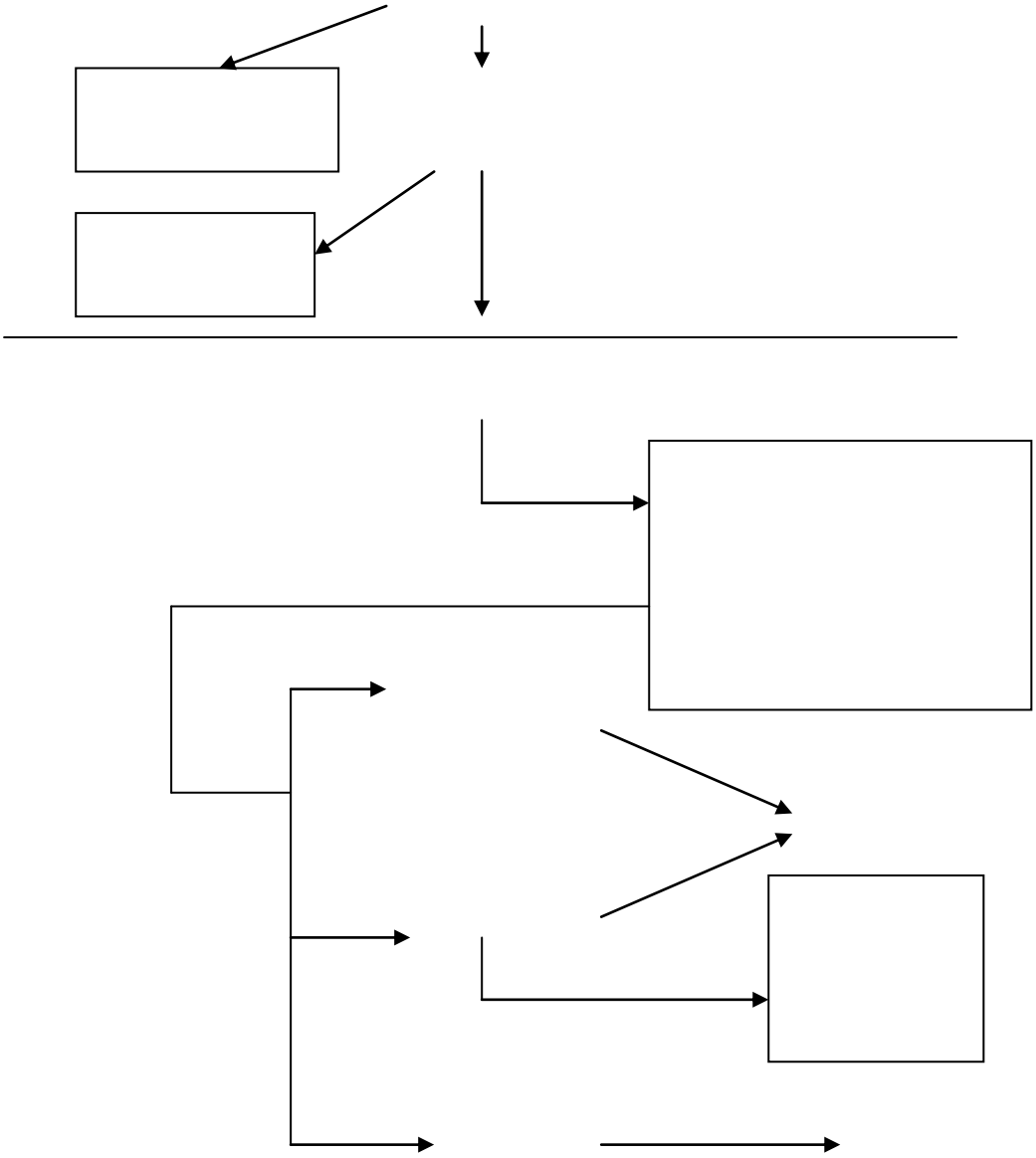


图 5.1 Red Hat Linux7.1 (Linux 内核版本为 2.4.2)启动过程

5.2 Linux 系统内核/系统配置小型化

实验目的：

1 个 Linux 系统发行版本可多达上百兆，功能众多，支持众多硬件设施。但在一个实际 Linux 应用系统中，并非用到 Linux 系统发行版本所提供的全部功能。所以，可以针对实际计算机系统的硬件配置及其应用需求，对 Linux 系统内核/系统进行合理的选择和裁减，得到一个更接近实际需要的、无冗余、启动和运行更为高效的 Linux 系统。

例如，嵌入式系统存储容量有限、支持的硬件外设类型也有限，因此可通过内核/系统配置与裁减，选择嵌入式应用系统所必需的一些内核/系统功能（如设备驱动程序）进行配置。由此得到一个满足系统功能、体积更小的、可放入嵌入式系统的容量较小的 Flash 中的 Linux 系统内核。

本实验要求学生针对 PC 机上的 Linux 系统发行版本，利用 make menuconfig 方法，在对硬件深入了解的基础上，选择相应的设备驱动程序和与主机 CPU 相关的 Linux 部分，进行 Linux 系统内核/系统配置，掌握 Linux 系统内核/系统的配置方法。

实验环境：

硬件：Intel i586 PC机

软件：Red Hat Linux7.1 (Linux内核版本为2.4.2) 或更高版本

实验内容与步骤：

1．对实验所用 PC 机硬件深入了解，了解 Linux 系统发行版本中的所包含的模块

2．按照参考文献[2]“6.1 Linux 内核配置与编译”、参考文献[1]“6.4 系统小型化”的内容，利用 make menuconfig 方法，将不必要的内核功能去掉，只留下最基本的一些功能模块，如 CPU 为 i586、IDE 设备驱动、TCP/IP 协议栈、ne2000 网卡驱动、Minix 文件系统等，完成内核配置工作。

3．比较配置前后 Linux 内核/系统的体积大小、功能差异，指明配置过程中各配置选项的作用

实验指导

1. Linux 的内核配置

创建一个能运行于目标系统之上的 Linux 内核的第一步是配置 Linux 内核，可以采用多种配置内核方法，配置过程中涉及许多配置选项选择，内核都会在配置完成之后生成一个.config 的文件，还会产生大量的符号连接和文件，在.cofig 文件中保存了在 menuconfig 中的选择定义的相应变量，在 Linux 内核目录下的 Makefile 文件中将会包含这个文件，

配置选项规定了在 Linux 内核中包含的选项。针对不同目标系统，选项菜单会有所不同，但有些选项则是任何嵌入式系统所必选的，如下是一些需要选择的主要选项：

- Code maturity level options
- Loadable module support
- General setup
- Memory technology devices
- Block devices
- Networking options
- ATA/IDE/MFM/RLL support
- SCSI support
- Network device support
- Input core support
- Character devices
- Filesystems
- Console drivers
- Sound
- Kernel hacking

当然一些系统也有它们自己的特殊的配置选项，如下即ARM的特殊选项：

- Acorn-specific block devices
- Synchronous serial interfaces
- Multimedia capabilities port drivers

需要注意的是，如果一个选项在配置菜单中是可选的，并不意味着目标系统就支持这个功能，而是配置菜单可以让用户尝试更多没有在目标系统上尝试过的内核功能。

目前的Linux内核主要支持4种主要的配置方法：

Make config、Make oldconfig、Make menuconfig、Make xconfig

在本实验中采用的是**Make menuconfig**的方法。

2．Linux 网络相关的内核编译配置

参见“附录 3 — Linux 网络系统及其配置”。

5.3 ARM/ucLinux- SkyEye 的安装、配置与集成

实验目的：

安装 SkyEye 嵌入式硬件模拟器，配置嵌入式操作系统 ucLinux，建立基于 SkyEye 的 ARM/ucLinux 嵌入式系统平台。

实验环境：

硬件：Intel i586 PC机

软件：Red Hat Linux7.1 (Linux内核版本为2.4.2) 或更高版本，SkyEye，ucLinux

或：Windows操作系统，Cygwin，SkyEye，ucLinux

注：

- SkyEye和ucLinux可从“光盘-《源码开放的嵌入式系统软件分析与实践——基于SkyEye和ARM开发平台》”中获取。
- windows下的Linux模拟软件Cygwin可以从其网站http：//www.cygwin.com上下载并安装最新版本。

实验内容与步骤：

参照参考文献[2]“3.4 安装使用 SkyEye”和“6.1.2 ucLinux 配置”、参考文献[2]所带光盘“光盘-《源码开放的嵌入式系统软件分析与实践——基于 SkyEye 和 ARM 开发平台》”中的 read 文件、“附录 4——SkyEye-ucLinux 安装说明”等内容，按照如下步骤进行：

1．编译安装 skyeye

2．安装 arm-elf-tools 交叉编译器，用于编译 ucLinux、SkyEye

3．安装 uclinux

4．配置 ucLinux，如选择选择模拟平台、配置 uclinux 网络支持

5．其中，网络部分的内核编译配置可参考“附录 3 — Linux 网络系统及其配置”。

6．利用参考文献[2] “3.4 安装使用 SkyEye”中所给例子，用 SkyEye 运行和调试 ucLinux 内核

5.4 –1 Linux Shell 小型化

实验目的：

Shell 是一种 Linux 系统与用户的交互方式，是最基本也是最重要的一个应用程序。很多嵌入式系统也离不开 Shell。Red Hat Linux 所用的 shell 是 BASH。BASH 功能强大，但高达 500 多 K，如果把所需要的库以及应用程序包括起来大小会有 3M 到 4M，用在嵌入式系统并不是很合适。

体积更小的 shell 主要有 BusyBox 和 ash，BusyBox 目前在嵌入式 Linux 中使用较多。BusyBox 将大量的 UNIX 工具集成到一个可执行文件中，实现了 UNIX 环境下所使用的文件操作、文本编辑、压缩数据等工具。目前 BusyBox 提供了 107 个命令的功能。在 i386 体系结构下，可执行文件的大小只有几百 KB。

本实验要求学生针对 PC 机上的 Linux 系统发行版本，理解 Linux 根文件系统组织结构，通过 Linux 系统内核/系统的配置，将应用程序 BusyBox 放入文件系统，利用 BusyBox 替换原有的 shell，掌握 Linux 系统 shell 小型化方法。

实验环境：

硬件：Intel i586 PC机

软件：Red Hat Linux7.1 (Linux内核版本为2.4.2) 或更高版本

实验内容与步骤：

a) 理解 Linux 根文件系统组织结构，按照参考文献[1]“6.4.3 小型化 shell”的内容，用 BusyBox 替换原有的 shell

b) 在新的 shell 下运行各类 shell 命令，验证替换过程的正确性

BusyBox 软件的获取：

- **BusyBox** 网站 http：//www.busybox.net

实验指导

作为一个目前在嵌入式 Linux 中使用较多的 shell，Busybox 包括了大量 Linux 下的工具。BusyBox 最早是由 Bruce Perens 于 1996 年为 Debian Linux 的安装盘所写，当时的目标是制作一张可以启动并运行的一个完整系统的修复盘。由于修复盘需要提供大量的命令工具，同时又受单张盘容量的限制，因此就诞生了一个最简单的 BusyBox。

BusyBox 将大量的 UNIX 工具集成到一个可执行文件中，可以小的代价替换了我们平时在 UNIX 环境下所使用的文件操作、文本编辑、压缩数据等工具。目前 BusyBox 提供了 107 个命令的功能。在 i386 体系结构下，它可执行文件的大小只有几百 KB，而这几百 KB 的大小就替代了原来 Linux 下 3MB-4MB 大小的工具。BusyBox 的具体大小还要看它在编译时采用动态连接还是静态连接。如使用 BusyBox 0.48 版本，在动态连接的时候它的大小是 263900 字节，当然这还要考虑 1.2MB 左右标准 C 库的大小。如果是静态连接，那它的大小是 781576 字节。BusyBox 就为嵌入式系统提供了一个比较完整的而且体积较小的 POSIX 运行环境，但这些命令的参数选项就要比原来完整的 GNU 命令少。

BusyBox 在编写时不仅考虑到代码体积的优化，同时还可以按照模块的方式加载所需命令。这就使用户在将 BusyBox 用于自己的嵌入式系统时可以很容易地按照自己的需求来增加或者删除命令。用户只要通过修改 BusyBox 源代码目录下的 config.h 文件除去自己不需要的工具以及工具中不需要的参数，这类似于使用 make config 命令来配置内核。

目前，BusyBox 由 Eric Anderson 维护，几乎是所有商用嵌入式 Linux 的一部分，在许多产品上都可以看到它。

在嵌入式 Linux 系统中使用 uClibc 和 BusyBox 的组合，可以有效地减小系统的规模。用 uClinux 提供的 M68K 体系的工具来建立一个静态链接 uClibc 库的 BusyBox，实现的过程大致分为两大块：

(1)建立 M68K 的交叉编译环境，具体的过程可以参看下一节的建立 ARM 的交叉

编译环境的介绍，惟一不同的是要用 uClibc 来代替 glibc。

(2）用 M68K 的交叉编译环境来编译 BusyBox。在这里选用 BusyBox-0.48，它的源代码包可以从 http：//www.busybox.net/downloads 下载。接着所作的就是修改一下源代码包中的 Makefile 文件，指定编译的工具是 M68K 的而不是机器上原来的 gcc 等。

5.4 –2 ARM/ucLinux- SkyEye 下 BusyBox 集成——Shell 小型化

实验目的：

理解 Linux 根文件系统组织结构，将应用程序 BusyBox 放入 Linux 文件系统，在嵌入式 ARM/ucLinux- SkyEye 环境下，集成小型化 shell BusyBox。

实验环境：

硬件：Intel i586 PC机

软件：Red Hat Linux7.1 (Linux内核版本为2.4.2) 或更高版本，SkyEye，ucLinux

或：Windows操作系统，Cygwin，SkyEye，ucLinux

实验内容与步骤：

1. 参照按照参考文献[2]“6.3.3 嵌入式应用软件 BusyBox”、参考文献[1]“6.4.3 小型化 shell”的内容，用 BusyBox 替换原有的 shell
2. 在新的 shell 下运行各类 shell 命令，验证替换过程的正确性

注：实验 5.4 –1 和 5.4 –2 可任选一个

5.5 Linux 环境下的 Ramdisk 技术及其实验

实验目的：

ramdisk 技术利用内存模拟硬盘空间，将一个内存块作为 1 个盘分区使用，可以提高访问速度。多某些特定的访问频率高的文件，将其存储于 ramdisk 上，可以提高访问性能。

在嵌入式系统中，利用 ramdisk 技术，可以实现应用程序预加载，即初始化完成后立即加载，而不是等到需要使用时才从硬盘读取， 从而提高系统速度：将一些系统启动后经常使用的程序，如浏览器、Xwindows，利用 ramdisk 将整个应用程序的可执行文件及其库复制到内存，然后修改可执行程序加载的搜索路径，用 ramdisk 所在的目录代替原来的可执行文件的目录路径。

本实验通过 Linux 环境下 ramdisk 的配置使用，了解 ramdisk 的原理和实现机制，验证其对改善系统运行速度的有效性。

实验环境：

硬件：Intel i586 PC机

软件：Red Hat Linux7.1 (Linux内核版本为2.4.2) 或更高版本

实验内容与步骤：

按照参考文献[1]“6.3 ramdisk 技术”相关内容，

1. 察看内核选项 CONFIG_BLK_DEV_RAM，确认 Linux 内核支持 ramdisk 选项
2. 创建/格式化 1 个 ramdisk 设备，并安装到文件系统的相应目录结构下
3. 运行命令“df –k /dev/ram0”，查看可使用的 ramdisk 空间大小
4. 重新修改 ramdisk 空间大小
5. 将 Linux 系统内常用的目录，如/temp 目录安装为 ramdisk，通过打开一个大文件，并对文件进行读、写操作，观察验证系统运行速度是否加快。**可以利用系统自身的时间测量函数**。

   要求：文件>100M
6. 编写一个文件访问程序，对比将此文件分别存储于外设硬盘、ramdisk 分区时，程序的执行速度。

   要求：程序反复多次读、写文件，以便对比不同方式下的访问时间差异。

实验指导

ramdisk 是作为一个盘分区使用的一个内存块，它将内存模拟为硬盘空间，从而可以像访问硬盘空间一样在其上保存文件。

使用 ramdisk 可以提高那些频繁访问磁盘文件的执行速度，若事先知道特定的文件将被高频率访问，通过将文件存放在内存里就可以提高数据访问速度。

在实时嵌入式系统中，可以采用应用程序预加载的方法，加快启动过程。某些系统启动后要运行的应用程序，如 X-windows、浏览器等，可以在内核初始化完成后立刻加载，而不是等到需要运行时才从硬盘读取。这个加载就是通过使用 ramdisk 将整个应用程序的可执行文件及其相关库复制到内存中，然后修改可执行程序加载的搜索路径，用 ramdisk 所在的目录代替原来的可执行文件所在的日录路径。当然，这样做是以可用物内存来换取系统启动速度。

从 Red Hat 6.0 开始，默认安装本身就有对 ramdisk 的支持。因此，ramdisk 使用非常简单，所要做的就是格式化一个 ramdisk 设备，然后安装（mount）到相应目录结构下即可。通过命令 ls -al /dev/ram 可以查看系统可利用的 ramdisk 设备的数目。在设置以后，这些 ramdisk 才发挥作用。

创建一个 ramdisk 的过程为

(1) 为 ramdisk 创建一个目录/挂载点，如 mkdir /mnt/ramdisk0

(2) 格式化/创建一个文件系统, 如 mke2fs /dev/ram0

(3) 将这个 ramdisk 安装(mount)到 /mnt/ramdisk0 目录下，

， 然后就可以将该目录作为一个分区使用。

若格式化时失败，则说明系统内核没有包括对 ramdisk 的支持，则需要重新编译内核，支持 ramdisk 的内核选项是：CONFIG_BLK_DEV_RAM。可以通过在

Block devices → RAM disk support

中作出选择来对 ramdisk 的支持编译成模块形式或者直接编译到内核中。

默认的 ramdisk 的大小为 4MB=4096K bytes。在创建 ramdisk 文件系统时可以得到 ramdisk 大小的信息.

安装完 ramdisk 分区之后, 运行命令 df –k /dev/ram0 查看可以真正使用的 ramdisk 空间大小, 因为创建文件系统时会占用一些空间.

Ramdisk 编译进内核后, 可以在 grub 中使用设置大小语句设置 ramdisk 大小, 例如

```
ramdisk_size=25000   //*设置 ramdisk 大小为 25M
```

需要注意的是，当系统重新启动以后，ramdisk中的数据会丢失。如果ramdisk中的数据被修改了, 必须将其备份到别的目录下。可以启动一个cron任务来完成这个工

作，如每10分钟检查一次看是否有数据被修改，如有则对数据进行备份。

除了将ramdisk支持选项编译进内核之外，还可以通过将ramdiak的支持编译为模块的方式，在需要时将其加载。模块方式可以在加载时动态地决定ramdisk的大小，方法为向/etc/conf.modules中添加下列语句：

options rd rd_size = 1000

或者在使用模块加载命令时以参数形式传入：

insmod rd rd_size = 1000

作为ramdisk的1个应用，可以将/temp目录安装为ramdisk，若系统中有很多程序使用/temp目录的话，则可以很明显地加快系统速度，而且每次系统重新启动时，这些暂时数据都会丢失.

5.6 Linux 环境下嵌入式数据库 mSQL 的集成

实验目的：

实时嵌入式应用常常需要数据库的支持。虽然很多情况下可以用文件方式实现部分数据库功能，但是当应用程序需要执行一些比较复杂的数据操作时，文件方式就无能为力了，更为合适的方式是采用适合实时嵌入式硬件和操作系统平台的嵌入式数据库系统。

要求学生通过本实验了解嵌入式数据库的基本特点。在 Linux 环境下安装配置嵌入式数据库 mSQL，建立简单的数据库，利用 SQL 语言和 mSQL API 实现简单的数据库访问功能。

实验环境：

硬件：Intel i586 PC机

软件：Red Hat Linux7.1 (Linux内核版本为2.4.2) 或更高版本

实验内容与步骤：

按照参考文献 [3]“第 14 章 嵌入式数据库”相关内容，

1. 从 mSQL 网站 www.hughes.com.au 下载一份以源代码方式发布的 mSQL 软件包，在 Linux 环境下正确安装
2. 用 mSQL 自带工具检查软件安装，并面向具体领域，建立数据库，如 GSM/GPRS 移动通信网络配置数据库
3. 利用 SQL 语句访问数据库系统，验证所安装系统的正确性。
4. 参考示范程序，利用 mSQL API，编写实现数据库访问功能的 C 语言程序。

实验指导

参见 “附录 5 嵌入式数据库 mSQL”.

6. 课程设计与实验报告要求

- 学生以小组为单位，**每组 2 人，每人至少参与 3 个实验，**共同完成上述工作。
- 实验完成后提交课程实验报告文档，并验收程序代码，进行上机演示。
- 课程实验报告应包括：题目，实验目的、实验内容、实验设计原理、实验步骤、实验结果及分析和**人员任务分配等**；
- 文档应格式规范，内容完整。包括封面、目录、正文内容、案例图片、附录等，应**附有实验运行截图。**
- 课程设计完成后，各组提交设计文档、程序，并通过程序演示进行课程验收。
- 按时提交文档，并进行验收。验收前，文档发至：yewen@bupt.edu.cn

  验收时间、地点：2015 年 7 月 16 号（周四，第 20 周），

  下午：14:00-18:00, 教 3-217

注意：验收时，全组成员必须到场！！

- **重点注意事项：**
  - ✧ 不要照抄指导书中内容
  - ✧ 将所有设计文档合并在一个 Word 文件中，图片插入文档中合适位置；所有程序代码以附录形式单独放在一个文件中
  - ✧ 提交文档的名称统一采取如下形式：

班级_姓名 1_姓名 2_操作系统综合课程设计

✧ 文档第一页如下，各人填写好学生信息、课程设计内容

北京邮电大学课程设计报告

课程设计名称		学 院		指导教师	
班 级	班内序号	学 号	学生姓名	成绩	
课程设计内容	简要介绍课程设计的主要内容，包括课程设计教学目的、基本内容、实验方法和团队分工等				
学生课程设计报告(附页)					

课程设计成绩评定	遵照实践教学大纲并根据以下四方面综合评定成绩： 1、课程设计目的任务明确，选题符合教学要求，份量及难易程度 2、团队分工是否恰当与合理 3、综合运用所学知识，提高分析问题、解决问题及实践动手能力的效果 4、是否认真、独立完成属于自己的课程设计内容，课程设计报告是否思路清晰、文字通顺、书写规范 评语： 成绩： 指导教师签名： 年　　月　　日

注：评语要体现每个学生的工作情况，可以加页

7. 参考文献与参考网址

1. 李善平，刘文峰，王焕龙等，《Linux 与嵌入式系统》，清华大学出版社，2003。
2. 陈渝，李明，杨晔，《源码开放的嵌入式系统软件分析与实践——基于 SkyEye 和 ARM 开发平台》，北京航空航天大学出版社，2004。
3. 刘峥嵘，张智超，《嵌入式 Linux 应用开发详解》，机械工业出版社，2004。
4. 深圳市英蓓特信息技术有限公司，Embest S3CEV40 uClinux，实验指导手册。

5．徐虹，何嘉，张钟澎，操作系统实验指导-基于 Linux 内核，清华大学出版社，北京，2004 年 11 月。

参考网址：

a) http：//www.linuxfans.org/
b) http：//www.uclinux.org/

8. 附录

附录 8.1. RPM 软件包管理器的下载与升级

1. 简介

RPM，Redhat 软件包管理器，是一种开放打包系统，任何人都可以使用。RPM 在 Red Hat Linux，以及其它 Linux 和 UNIX 系统上运行。Red Hat, Inc. 鼓励其它销售商在他们自己的产品上使用 RPM 技术。RPM 按照 GPL 条款被发行。

Red Hat Linux 系统上的所有软件都被分成可被安装、升级、或删除的 RPM 软件包。使用图形化和命令行工具可以管理 Red Hat Linux 系统上的 RPM 软件包。

RPM 软件包的获取：

- Red Hat Linux 发行版本光盘
- http：//www.rpm.org/
- Red Hat 勘误网页： http：//www.redhat.com/apps/support/errata/
- Red Hat FTP 镜像网站： http：//www.redhat.com/download/mirror.html

对于终端用户来说，RPM 简化了系统更新。安装、删除安装、升级 RPM 软件包可以使用简短的命令就可完成。RPM 维护一个已安装软件包和关于它们的文件的数据库。拥护可以在系统上使用功能强大的查询和校验，并提供了图形化界面。

对于开发者来说，可利用 RPM 将开发的软件编码和程序打包，然后提供给终端用户，减轻了发行软件新版本所带来的维护负担。

2. RPM 的设计目标与功能

- 可升级性

使用 RPM，用户不必全盘重装就可以在系统上升级个别组件。当用户得到一个基于 RPM 的操作系统的新发行版本（如 Red Hat Linux），用户不必重新安装系统（基于其它打包系统的操作系统需要重装）。RPM 允许智能化、自动化地就地升级用户的系统。软件包中的配置文件在升级中被保留，因此用户不会丢失定制的设置。

- 强大的查询功能

RPM 提供了强大的查询功能。用户可以在整个数据库中搜索软件包或某些特定文件。用户还可以轻易地了解到哪个文件属于哪个软件包，软件包来自哪里。RPM 软件包的文件包括在被压缩的归档中，其中有定制的二进制档头，该档头内包含关于软件包及其内容的信息，允许用户快速简捷地查询个体软件包。

- 系统校验

如果用户担心用户可能删除了某软件包上的一个重要文件，只需校验该软件包即可。任何异常情况都会向用户通知。到时，用户可以在必要时重装该软件包。用户修改过的配置文件在重装中会被保留。

- 纯净源码

允许使用与软件的原创者所发行源码一致的“纯净”软件源码。使用 RPM，用户会有纯净源码、使用过的补丁、以及完整的建构指令。这是一个重要的优越性。首先，如果程序的新版本被推出,用户不必从头开始编译。用户可以看一看补丁来判定用户 可能 需要做什么。使用这种技术，所有内编译的默认值，以及为正确建构软件而进行的任何改变都一目了然。

保持源码纯净对开发者来说是重要的，它也给终端用户带来高质量的软件。

附录 8.2　Linux 的内核态和用户态下的启动过程

以在 PC 机上运行 Red Hat Linux7.1（Linux 内核版本为 2.4.2）为例，说明系统从内核启动到字符登录界面出现这一过程。

CPU 加点执行完 BIOS，系统控制权将交给硬盘第一个扇区后，就开始由 Linux 来控制系统了。对 i386 体系结构的系统而言，系统引导扇区的代码是由 Linux 内核中的文件 arch/i386/boot/bootsect.S 所生成的。它先将引导扇区的代码复制到内存地址 0x90000 中，然后将 setup 部分（linux/arch/i386/boot/setup.S ）复制到内存地址 0x90200 中，最后将实际的内核映像复制到 0x100000 (0x100000 是使用 bzImage 时的位置，bzImage 对应的地址是 0x1000)。bootsect..S 完成加载动作后，就直接跳转到地址 0x90200，这里正是 setup.S 的程序入口。

setup.S 的主要功能就是将系统参数（由 BIOS 返回的包括内存、磁盘等参数信息）复制到 0x90000-Ox901FF 内存地址中，这个地方正是 bootsect. S 刚才存放的地方，这时它将被系统参数覆盖。以后这些参数将由保护模式下的代码来读取。除此之外，setup.S 还调用 video.S 中的代码，检测和设置显示器和显示模式。最后，setup. S 将系统转换到保护模式，并跳转到地址 0x100000 的内核引导代码。整个这一部分的启动时间只有几个毫秒，对于整个系统的启动时间来说可以忽略不计，也不需要对这部分代码作任何的改动。

参见“图 5.1 Red Hat Linux7.1 (Linux 内核版本为 2.4.2)启动过程”，Linux 内核启动后执行的第一个函数为 start_kernel()(linux/init/main.c), 完成以下初始化工作

- printkk(linux_banner)，显示Liux内核的版本信息。
- setup_arch(&command_line)，做与体系结构相关的初始化工作。
- parse_options(command_line), 解释系统参数
- trap_init(), 设置系统异常的入口点
- init_IRQ()，初始化系统中断服务
- sched_init(), 系统调度器的初始化
- time_init()，时钟、定时器初始化
- softirq_init(), 系统软中断的初始化
- console_init, 控制台初始化
- kmem_cache_init，内核cache的初始化
- calibrate_dalay(), 校准时钟
- mem_init(), 内存初始化
- kmem_cache_size_init()，创建及设置通用cache
- fork_init(mempages), 建立uid_cache，并根据系统内存大小来确定最大进程数目
- buffer_init(mempages), 块设备缓冲区的初始化，初始化一系列的cache
- check_bug()，检查体系结构漏洞

- kernel_thread(init, NULL, CLONES_FS|CLONES_FILES|CLONES_SIGNAL)，创建第一个核心进程，启动init进程
- cpu_idle(), 运行idle进程

下面的初始化工作由init()函数来完成。Init()首先锁定内核，然后调用do_basic_setup()来完成外部设备以及驱动程序的初始化。外设的初始化要根据内核的配置来决定．一般需要做下面的初始化工作

- PCI总线初始化
- 网络初始化
- 一系列其它设备的初始化
- start_context_thread()创建事件管理核心进程keventd
- 通过do_initcalls函数来启动任何使用_initcall标识的函数
- 文件系统初始化
- 加载文件系统

在do_basic()调用完成之后， init()会释放初始化函数所用的内存，并且打开/dev/console设备重新定向控制台，让系统调用execve来执行程序init()。

到这里为止，Linux内核的初始化工作已经完成。下面开始用户态进程的初始化。Init()程序存放在目录/sbin下，运行时需要读/etc/inittab文件来决定它具体的工作内容。在inittab中比较重要的几条是：

- id ： 3： initdefault, 决定系统登录的方式，通常出现的数值是3或者5，3表示多用户模式, 5表示X图形登录方式。
- si：： sysinit：/ etc/rc.d/rc.sysinit,执行/ etc/rc.d/rc.sysinit的脚本, rc.sysinit中最常见的动作就是激活交换分区，检杳磁盘，加载硬件模块，这些动作无论在哪个运行级别中都需要优先执行
- 13： 3： wait： /etc/rc.d/rc 3, 脚本/etc/rc.d/rc的执行必须在rc.sysinit脚本执行完之后再进行。根据前而所指定的优先级3，以3作为命令行参数，然后再执行/etc/rc.d/rc3.d目录下面的所有脚本。
- 1： 2345： respawn：/sbin/mingeety tty1，在rc返回后，init将得到控制权．并启动mingetty输出登录界而及提示，接受用户名的输人并以该用户名作login参数，加载login程序

到此为止，作为宇符界面的 Linux 启动已经完成。

附录 8.3 Linux 网络系统及其配置

3.1. Linux网络层结构

从整体角度考虑，Linux网络系统基本可以分为硬件层／数据链路层、IP层、INET Socket层、BSD Socket层和应用层五个部分。其中在Linux内核中包括了前四个部分，在应用层和BSD Socket层之间的应用程序接口以4.4 BSD为模板。INET Socket层实现比 IP协议层次高，实现对IP分组排序、控制网络系统效率等功能。IP层是在TCP/IP网络协议栈中心的互联网层实现。硬件层在TCP/IP协议栈本身中就和数据链路层区分不明确，暂时称这个包括了硬件驱动和硬件发送组织工作的层次为硬件层。

3.2. Linux内核编译选项的配置

开放的Linux内核允许配置和编译自己需要的内核映像，用于满足有特殊需要的硬件环境。在Linux内核开发小组人员每次发布的新内核中，包括了内核配置程序的代码。可以使用3种方式配置内核选项。在linux目录下运行make config, make menuconfig或者make xconfig可以分别运行不同界面的配置程序。在每个内核源代码arch目录下的config.in文件就是内核配置程序代码需要获得的源数据；在linux/script/目录下存放了几个不同操作方式的内核配置程序。其中，linux/script/menuconfig是用终端dialog方式配置内核的程序；linux/script/configure是用终端命令行方式配置内核的程序。另外如果使用xconfig，会利用scripts目录下的tcl/tk程序创建kconfig.tk文件用作运行xconfig配置方式的脚本，在Window环境下运行窗口配置界面。

选择编译选项可能有多种选择。对于一些以Linux模块（Module）实现的程序，可以选择编译为 Module或者编译在内核之中。那些没有实现为Module的程序，就只能选择是否编译进内核了。一般而言，驱动程序、文件系统等代码在Linux中都是实现为Module的，也就是说，可以在配置时根据硬件板的情况选择需要的文件系统和驱动程序代码，去除不需要的代码，从而减小最终image的大小。如果需要特别小的内核，但是设备驱动又不能舍弃，否则无法驱动必要设备，那么可以使用Module文件作为驱动程序代码。

在内核编译选项选择完毕之后，保存设置，就可以开始编译内核了。这时候的内核编译配置存放在隐藏文件linux/.config中。在这个文件中的宏一般都是以CONFIG_为前缀，表示各个内核中可选的配置，如果某个宏的值是y，则表示这个宏代表的选

项将要被编译进入内核；如果某个宏的值是n，则表示这个选项不用编译；如果某个宏的值是m，则表示这个选项将要被编译成为Module。这些宏在内核中也会出现，.config是参与预编译过程的。其实，可以直接编辑．config中的选项，免去配置过程的麻烦。但是 .config文件的生成经过了依赖性检测，因此自己手动编辑文件无法检测依赖性，如果没有经验，最好不要选择手动编辑 .config文件的方法。

3.3．网络相关的内核编译选项配置

一般情况下，如果需要嵌入式Linux系统支持TCP/IP网络协议栈，只需要标记内核支持TCP/IP协议栈即可。首先是“General Setup " ——〉 " Networking Support”支持(CONFIG_NET)，然后是“Networking Options " ——〉" TCP/IP networking”支持(CONFIG_INET)。这样编译出来的内核已经可以支持TCP/IP协议栈了，包括TCP协议、UDP协议和IP协议。但是，付出的空间代价是将内核映像的大小增加了100KB左右。

对应于网络的配置，在 .config文件中的宏对应的就是内核中某一个选项开关。对于网络而言，这样的选项开关也非常多。下面是几个常用的选项。

- CONFIG_NET

整个Linux是否提供网络支持，最终都归结到这个宏来控制。即使将Linux运行在一个没有网络连接的环境，让Linux有网络支持也是让Linux大多数应用程序可以正常运行的关键。因为Linux中大多数程序都是建立在网络基础之上的，没有了网络支持，程序将无法运行，或出现这样那样的异常。因此一般情况下都要将这个开关打开。

- CONFIG_INET

在Internet和大部分以太网媒质上使用的协议。大部分在Internet网络上流行的网络程序都是基于TCP/IP协议的。即使在没有网络连接的情况下，这些程序还是需要运行，因为它们可以基于loopback网络设备完成网络连接。只有打开这个选项才能支持TCP/IP协议栈。选择了这个选项之后，TCP, UDP, IP, ICMP, ARP等协议自动获得了支持。这个选项会将内核映象增加114KB左右；如果对TCP/IP协议栈做精简，可以使内核映像增加的数量比这个数字小很多。

- CONFIG_NET_ETHERNET

支持以太网协议（IEEE 802.3或者ISO 8802-2）的选项。这个支持可以实现对10Base-2, 10 Base-T, 10Base-F, 100Base-TX, 100Base-T4, 100Base-FX, 1000M以太网的支持。只有选择了该选项之后才能选择对以太网卡的支持。

- CONFIG_PACKET

包协议用于那些在网络设备之间不通过内核中所有其他协议传输数据的程序，也

就是说，使用“原始数据包”(raw packet)传输数据。例如，用来监测TCP/IP协议栈数据的程序tcpdump，它只使用packet协议，而不使用TCP/IP协议栈。检测网络数据的应用程序一般都会和tcpdump一样使用这个协议。

- CONFIG_NETFILTER

在使用Linux作为路由器或者网关的情况下需要将网络数据包做过滤和混合。这个选项控制了整个过程的框架结构，又称Packet Filter。防火墙的功能也在这里实现，可以根据网络数据包的来源、类型和目的地等信息判断是否应该接受或转发数据包。如果作为网关，连接在局域网和Internet之间，局域网中使用内部IP地址，由网关来做内部IP地址到外部IP地址之间的转换，可以使得多台机器以同一个外部IP地址连接在Internet上。Packet Filter负责完成这个“IP伪装”（又称Network Address Translation NAT）的功能。

- CONFIG_UNIX

选择Unix domain socket的选项。在UNIX中，sockets是标准的建立网络连接和访问网络的机制。同时，Unix domain socket也可以作为一种进程间通信时使用的机制。根据传统，很多X Window程序和syslog程序都使用Unix domain sockets建立网络和进程之间的连接，因此如果系统中需要支持这些程序，便需要打开这个选项。

- CONFIG_IP_MULTICAST

用于IP上多播的代码。

- CONFIG_IP_PNP

自动配置IP地址的上层选项。如果选择了该选项，那么至少需要提供下面三个选项之一来确定自动配置IP地址的方式；如果选了多个，就会逐个尝试自动配置的方式：

(1) DHCP (CONFIG_IP_PNP_DHCP)：通过DHCP服务器分配IP地址，会在局域网内部自动寻找DHCP服务器。

(2) BOOTP (CONFIG_IP_PNP_BOOM：通过BOOTP服务器获得IP地址。

(3) RARP (CONFIG_IP_PNP_RARP)：通过RARP服务器获得IP地址。

(4) CONFIG_NET_IPIP

IP隧道（tunnel）的选项配置。隧道可以将一个协议的数据封装在另外一个协议的数据包中，然后通过一个可以理解该协议的隧道传输此数据包。IP隧道是用IP协议封装IP协议的隧道，可以通过隧道将两个物理上不在同一区域的局域网连接成同一个局域网。

有关网络配置的选项都可以在linux/Documentation/Configure.help文件中找到帮助。

附录 8.4 SkyEye-ucLinux 安装说明

以下安装过程是在 Red Hat 9 系统下编译通过的，最好用 2.4 以上的内核＋gnome 图形界面编译。

Step1. 将下载的 4 个数据包（skyeye,arm-elf-tools,uClinux-dist,8019AS）copy 至安装目录(这里假设为/embed)，进入目录 embed：cd /embed

Step2. 安装 skyeye：

2.1 解压 skyeye：tar -jxvf skyeye-0.8.6.tar.bz2

2.2 进入 skyeye： cd skyeye

2.3 预处理： ./configure --target=arm-elf --prefix=/usr/local

2.4 编译安装： make;make install

Step3. 安装 arm-elf-tools 交叉编译器 (用于编译 ucLinuxSkyEye) ：

3.1 如果 arm-elf-tools-20030314.sh 没有执行权限，则增加执行权限；

3.2 安装： /arm-elf-tools-20030314.sh

Step4.安装 uclinux：

4.1 解压： tar -zxvf uClinux-dist-20030522.tar.gz

4.2 将 8019AS 中文件夹 1 中的全部文件拷贝到 uClinuxdist/linux2.4.x/drivers/net 覆盖同名文件。

4.3 将 8019AS 中文件夹 2 中的全部文件拷贝到 uClinuxdist/vendors/GDB/ARMulator 覆盖同名文件；

/*此处是将用户自己开发的 8019AS 网卡驱动程序集成进 Linux 系统中，谨供参考，对于用户开发的其它应用程序可按类似方法集成进系统；如果不需要集成用户应用程序，可省略这 2 步*/

4.4 进入解压出的目录： cd uClinux-dist

4.5 执行：./make menuconfig

4.6 选择模拟平台： Target Platform Selection-->Vendor/Product->GDB/ARMulator，需要选择 Customize Kernel Settings；

4.7 退出，保存；

4.8 继续配置 uclinux 以支持网络：

Network device support->Ethernet (10 or 100Mbit)：' SkyEye ne2k ethernet support(for ARMUlator)'；

4.9 退出，保存；

4.10 编译生成：make dep;make

Step5. 配置：

5.1 在工作目录(/embed/uClinux-dist)下建立专门用于基于 AT91X40 开发板的 SkyEye 硬件配置文件 skyeye.conf：

cpu： arm7tdmi

mach： at91

mem_bank： map=M, type=RW, addr=0x00000000, size=0x00004000

mem_bank： map=M, type=RW, addr=0x01000000, size=0x00400000

mem_bank： map=M, type=R, addr=0x01400000, size=0x00400000, file=./boot.rom

mem_bank： map=M, type=RW, addr=0x02000000, size=0x00400000

mem_bank： map=M, type=RW, addr=0x02400000, size=0x00008000

mem_bank： map=M, type=RW, addr=0x04000000, size=0x00400000

mem_bank： map=I, type=RW, addr=0xf0000000, size=0x10000000

net： state=on, mac=0：4：3：2：1：f, ethmod=tuntap, hostip=10.0.0.1

5.2 建立文件系统的链接，需要在目录/embed/uClinux-dist 目录下执行如下命令：

#ln -s images/romfs.img boot.rom

Step6. 运行：

6.1 在/embed/uClinux-dist 目录下：/usr/local/bin/skyeye linux-2.4.x/linux

6.2 在 skyeye 环境下：

tar sim

...

load

...

run

经过上述操作，即可完成系统安装。

附录 8.5 嵌入式数据库 mSQL

6.1.嵌入式数据库的特点

由于应用环境的特殊限制，嵌入式数据库相对普通数据库而言有其自身的特点

- 支持常用嵌入式系统（如 Linux ，Windows CE ，Palm OS 等）和通信协议。内核小，占用内存小。
- 提供数据库功能的自由限制，能够根据具体应用或行业特点定制系统功能。
- 方便的查询功能，支持 SQL 查询语句。
- 完善的数据管理功能，支持 SQL 标准的子集，提供数据库及数据表的管理等功能。
- 操作简单方便，提供简明的 API 接口，可在高级语言中调用。

6.2.mSQL 简介

MiniSQL(mSQL)由澳大利亚的 David J Hughes 开发，是嵌入式数据库家族中的佼佼者。

mSQL 是一种小型的关系数据库管理系统。它自身结构小巧紧凑，占用系统资源小。但是它的功能十分的强大，足以胜任大型数据库集的索引、查询任务。但是不足的是，mSQL 对某些 SQL 的功能不支持。

6.3. mSQL 的安装及配置

6.3.1. mSQL 的安装

mSQL 以两种形式发布：一种是 RPM 软件包形式；另外一种是 tar 压缩的源代码方式。

- RPM 软件包的安装很简单，命令如下：
  rpm –ivh xxxxxx.rpm(文件名)
  RPM 软件管理器简化了系统更新的步骤，一个简单的命令就能完成所有的文件安装，
- 以源代码方式发布的 mSQL 的步骤如下：
  首先，用 tar 程序解开压缩包：
  tar –zvxf xxxxxxx.tar.gz (文件名)

该命令会在当日安目录下建立一个名为 xxxxxxx 的文件夹，它用于存放所有的发布文件，包括源代码目录 src 和文档目录 doc。接下来的步骤要使用 setup 程序来设置后面的编译选项。

./setup

该命令会将一些编译选项保存在 src/site.mm 文件中，如果用户需要改变 mSQL 程

序的安装路径以及 C 编译器的类型，可以修改该文件中的对应内容。程序的默认安装路径为/usr/local/msql3.

接下来，开始编译 mSQL 的源程序，命令如下：

make all

如果编译完成而且正确，则可以开始安装 mSQL，命令如下：

make install

至此，mSQL 即被正确安装在系统中。

6.3.2. mSQL 的系统配置

mSQL 的系统配置文件名为 msql.conf,位于安装目录下（默认是/usr/local/msql3）。另外，所有标准 mSQL 应用程序及公用程序都可以通过在执行时加上－f 参数来指定一个非标准的配置文件，以强制改变原有的默认参数值。这时当应用程序没有找到配置文件（或虽找到但有部分参数未设定）。就会自动使用默认值。

1．配置文件格式

配置文件由若干个段（section）组成，可包含空白行和注释，在注释之前应有“＃”字符。每个段落都有一个段落标题，用方括号括起作为段落名称。

段落中参数值的设定方法：在参数名称之后跟上一个等号和相应的参数值，等号之后的参数值即为新值，但是每一行只能由一个参数设定项。如果配置文件中某些参数值未获设定，则 mSQL 执行时会使用默认值。

2．配置文件参数说明

（1）general 段。

mSQL 运行时使用的一些通用参数一般在配置文件的 general 段设定，下面介绍部分参数的意义：

- Inst_Dir：mSQL 的安装目录，默认时/usr/local/msql3。
- DB_Dir：用户建立的数据库文件保存路径，默认时％I/msqldb。％I 代表上面的 Inst_Dir；即 DB_Dir 的默认时/usr/local/msql3/msqldb。
- Msql_User：mSQL 服务器当前的用户，默认是 daemon。若由别的用户激活服务器，则系统的用户号 UID 会发生改变。
- Admin_User： mSQL 的特权用户，默认是 root。特权用户可以执行特权操作，如数据库的建立和关闭等。
- id_File： mSQL 服务器进程号 PID 的保存文件完整路径，默认是％I/msql3.pid。
- UNIX_Port： mSQL 服务器 UNIX 套接字文件完整路径，默认是％I/msql3.sock。本级客户通过这个套接字文件与服务器联接。
- TCP_Port： mSQL 服务器的 TCP 服务端口，默认是 1114。基于 TCP/IP 网络 的客

户端通过这个端口和服务器连接。

（2）System 段

mSQL 的系统参数都在配置文件的 System 段中设定，下面介绍各个参数的意义。

- Msynch_Timer：定义 mSQL 服务器自动使用内存数据与硬盘数据的时间间隔同步，以秒为单位，默认是 30。如果该值设定为 0，则服务器不自动使用内存与硬盘数据同步。
- Host_Lookup： 决定是否需要主机的 IP 地址。默认是 True，表示不符合主机名称的连接请求将被拒绝。
- Read_Only：设置服务器工作模式为只读，拒绝任何修改数据库系统的操作。默认是 False。
- Remote_Access：允许基于 TCP/IP 网络的远端用户访问 mSQL 服务器，默认是 False。
- Local_Access：允许本机用户应用程序访问 mSQL 服务器，默认是 True。
- Num_Children：设置 mSQL 多进程服务器可以同时处理的任务数，默认是 2。

3．配置文件范例

配置 mSQL 数据库管理系统所做的工作就是修改其配置文件 mslq.conf。在实际应用中经常要改动的是 Msql_User 和 Admin_User 两个参数。Msql_User 用来设定运行数据库服务器的用户。而 Admin_User 用于设置能对数据库系统执行特权操作的用户。因此，如果设置 Msql_User＝dbman，Admin_User＝admin，则表示由 dbman 用户运行服务器程序，由 admin 用户执行特权操作。

配置完成后，以 root 登陆，创建 dbman 用户：

Useradd –g sqlusers dbman

接下来，将/usr/local/msql3 目录下的文件及目录的拥有者改成 dbman，执行以下命令：

Chown –R dbman

再进入/usr/local/msql3/bin 目录，执行：

./msql3d &

这样就以后台方式启动了 mSQL 服务器系统，从而开始进行具体的数据库创建、查询等操作。

以下是一个配置文件的范例：

6.4. mSQL 的工具程序

mSQL 中包含的使用及管理管理数据库的工具程序共有 6 个，分别用于完成数据库

管理、服务器程序监视、数据库结构检索、数据转储、数据导出及导入等操作。

- 服务器管理程序 msqladmin

  Msqladmin 用于对 mSQL 服务器执行管理操作，如建立新数据库、关闭服务器、数据库复制等。

- mSQL 的交互程序 msql

  Msql 为用户提供一个与 mSQL 服务器进行对话的界面，用户可以通过它向 mSQL 服务器发送标准的 SQL 命令。msql 通常是用来建立数据表或是向服务器传送 SQL 查询命令，以测试数据库内容是否正确，在用户应用程序中并不应用。

- 数据库结构浏览程序 relshow

  relshow 用于显示 mSQL 中数据库的结构

- 数据库的重建程序 msqldump

  msqldump 用于产生一个包含标准 SQL 命令的的 ASCII 文本文件，通过这个文件可以重建数据库。这个 ASCII 文本文件包含了重建一个数据库所需的 CREATE TABLE 、CREATE INDEX 以及 CREATER SEQUENCE 命令，以确保重新建立的数据库中的资料与原数据库中相同。

- 数据导出程序 msqlexport

  msqlexport 程序以纯文本方式导出指定数据表中的数据，输出的文本数据可以用于其他的程序中。

  利用 msqlexport 程序可以对输出的文本数据格式进行灵活设置。用户自己可以自定义字段分隔符、数据中的分隔符替换字符、每项数据是否要引用符号括起来以及用什么符号作为引用符号。

- 数据导入程序---msqlimport

  Msqlimport 程序的作用是与 msqlexport 相反，是将一个纯文本文件内容导入到 mSQL 数据表。这时文本文件中的每行作为数据表的一个记录

6.5. mSQL 的 C API 函数

嵌入式系统中的应用程序通过调用 mSQL 的 API 函数来执行对特定数据库的操作，任何 C 语言程序可以利用 API 函数与 mSQL 的数据库引擎进行通信。

mSQL 的 API 函数库名称为 libsql.a，一般位于 mSQL 安装路径下的 lib 目录中，库中的函数在 msql.h 中定义。用户在编写程序时，应包含该头文件，该文件一般位于 mSQL 安装路径下的 inclube 目录，如/usr/local/msql3/include 中。另外，在对 C 程序进行编译链接的时候，应加上链接参数。

下面对 mSQL 的 API 函数作简单的分类介绍：

6.5.1.查询类函数

（1）msqlConnect()函数

该函数定义为：

用于建立与 mSQL 服务器的连接。参数 host 为服务器所在的主机的名称或 IP 地址，当参数 host 设为 NULL 时，表示将连接本机上运行的 mSQL 服务器。若函数调用成功，则返回一个代表连接描述符的整形值，作为连接句柄工其他 API 函数调用。在使用过进程服务器的程序中，多次调用该函数将产生多个不同的连接句柄。

（2）msqlSelectDB（）函数

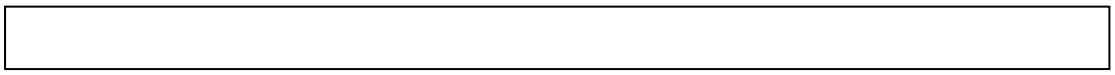
该函数定义为：

在对一个数据库进行查询等操作之前，必须先调用 msqlSelectDB()函数选中他。参数 sock 为调用 msqlSelctSB()函数返回的连接句柄，db 为要选择的数据库名称。当应用程序成功选中某个数据库后，就可以对该数据库进行各种操作。

（3）msqlQuery（）函数

该函数定义为：

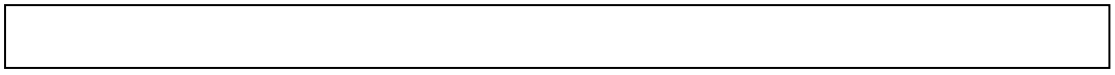
PDATE 等 SQL 命令。参数 sock 为调用 msqlConnect（）函数返回的连接句柄，q 为标准的 SQL 语句。如：



执行该函数后，从服务器返回的数据保存在内存中，应用程序需紧接着调用下面的 msqlStoreResult（）函数以获得查询结果，否则一旦应用程序提交了其他的查询请求，本次查询的结果将丢失。

（4）msqlStoreResult（）函数

该函数定义为：



msqlStoreResult（）函数用于保存查询结果，该结果保存在一个 m_result 结构中，应用程序通过一个指向该结构的指针来获取查询结果。

（5）msqlFreeResult（）函数

该函数定义为：

该函数与 msqlStoreResult（）函数向对应，当应用程序已经取代了查询结果或不需要该结果数据的时候，应该调用这个函数来释放内存。

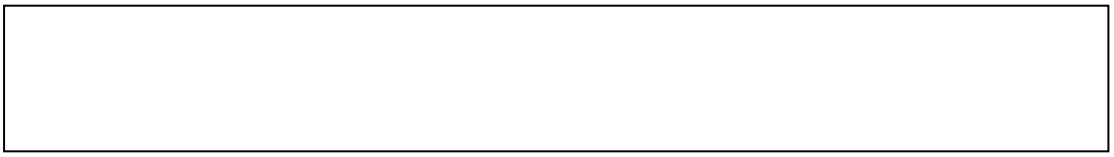
（6）msqlFetchRow（）函数

该函数定义如下：

用于提取通过 msqlStoreResult（）函数保存的查询结果的某个记录数据，该函数将数据保存在一个 m_row 数组中。

（7）msqlDataSeek（）函数

该函数定义为：



用于移动服务器相应用户查询所返回的记录集的游标。参数 handle 指向一个保存查询结果的 m_result 结构体，offset 为当前记录位置，其实之为 0。应用程序通过调用 msqlDataSeek（）函数移动记录集游标后，再调用 msqlFetchRow（）函数来提取相应记录。

（8）msqlFetchField（）函数

该函数定义为：

用于获取某个查询字段的信息，包括字段名、表名、字段数据类型、字段长度以及文字属性，返回结果将保存在一个 m_field 结构中。

（9）msqlFieldSeek（）函数

该函数定义为：

与 msqlDataSeek（）函数用法相似，该函数用于移动查询字段的游标，参数 handle 指向一个保存查询结果的 m_result 结构体。Offset 为当前字段符号，起始值时 0。应用程序通过调用 msqlFieldSeek（）函数移动字段游标后，再调用 msqlFetchField()函数来提供相应字段信息。

（10）msqlClose（）函数

该函数定义为：

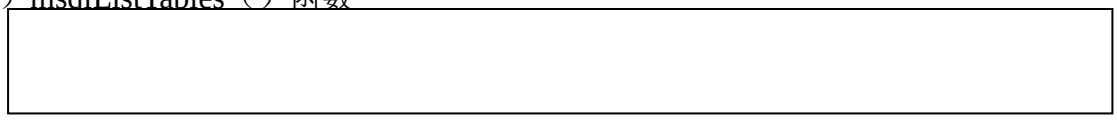
当应用程序不再使用某个已打开的数据库时，应及时调用 msqlClose（）函数关闭连接，以释放系统资源。参数 sock 为用 msqlConnect（）函数打开的连接句柄。

6.5.2.纲要类函数

（1）msqlListDBs（）函数。该函数定义为：

用于获取当前连接的 msql 服务器上的数据库列表。

（2）msqlListTables（）函数



用于查询当前选中的数据库中的数据表。

（3）msqlListIndex()函数

该函数定义为：

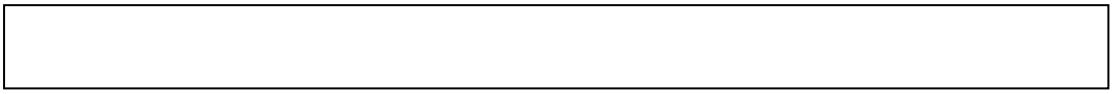
用于获得数据表索引，返回结果保存于 m_result 结构中。

（4）日期时间类函数

mSQL 的 API 函数中还包含了丰富的日期时间转换函数，以满足用户所需的格式。如 msqlTimeUnixTime（）和 msqlUnixTimeToData（）。

（5）其他类型函数

mSQL 中还有一些其他类型的函数，如配置文件加载函数 msqlLocalConfigFile().其定义为：



用于加载一个非默认的配置文件，参数 file 为要加载的配置文件名。