专题3-嵌入式Linux内核制作

一、Linux内核简介

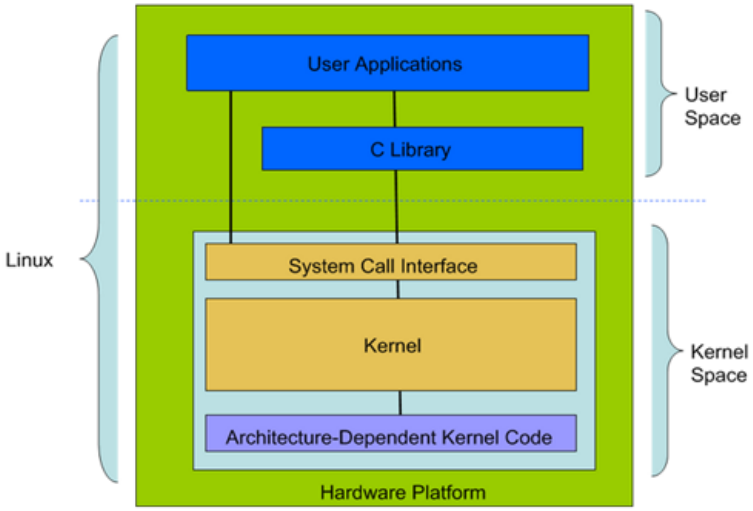
1.1、系统架构

可参考这网址的内容http://blog.csdn.net/hguisu/article/details/6122513

1.2.1、Linux系统架构

Linux体系结构，从大的方面可以分为用户空间（User Space）和内核空间（Kernel Space）。

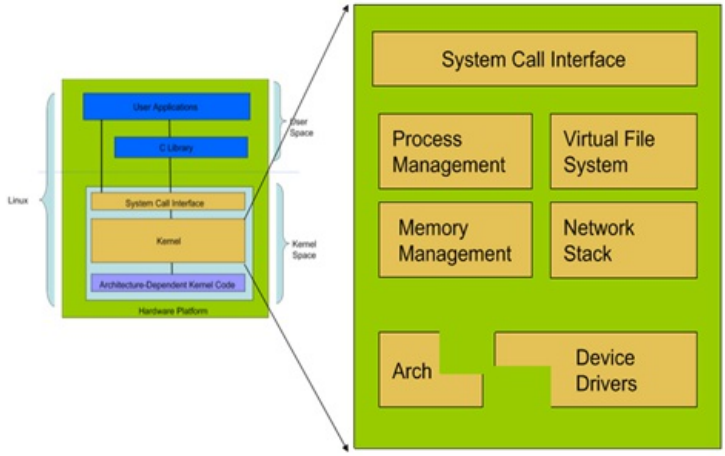
内核空间和用户空间是程序执行的两种不同状态，通过系统调用和硬件中断能够完成从用户空间到内核空间的转移。

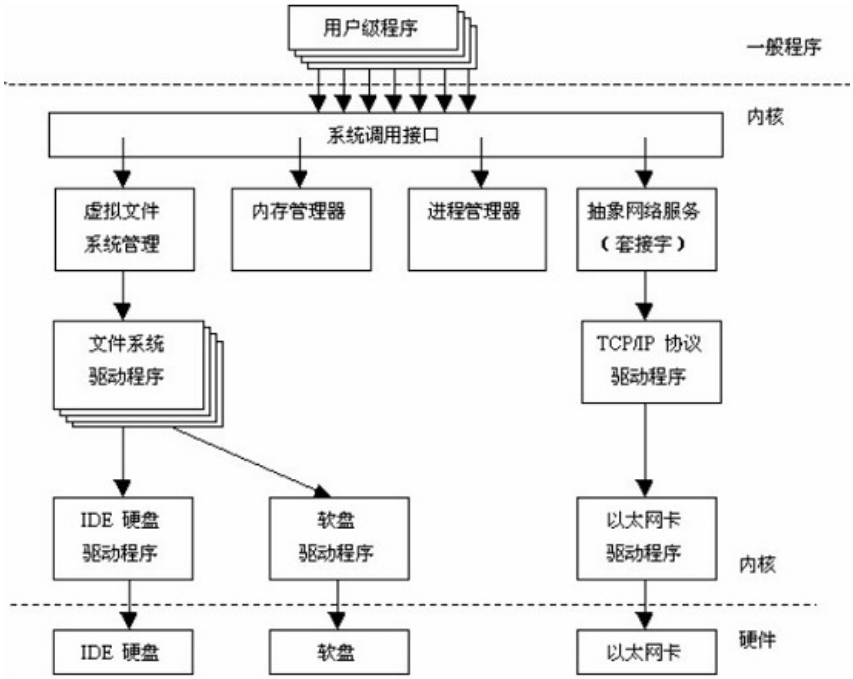


用户空间中包含了C库，用户的应用程序。在某些体系结构图中还包含了shell，当然shell脚本也是Linux体系中不可缺少的一部分。

内核空间包括硬件平台、平台依赖代码、内核、系统调用接口。

1.2.2、Linux内核架构



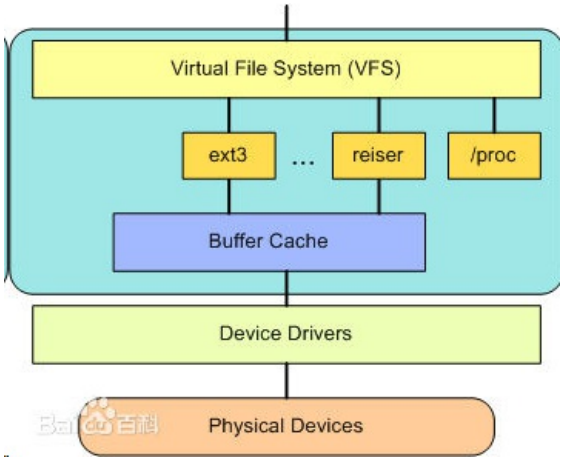


SCI层（System Call Interface），这一层是给应用用户空间提供一套标准的系统调用函数来访问Linux。前面分析Linux体系结构的时候，介绍过任何一类现代操作系统都不会允许上层应用直接访问底层，在Linux中，内核提供了一套标准接口，上层应用就可以通过这一套标准接口来访问底层。

PM（Procees Management），这一部分包括具体创建创建进程（fork、exec）,停止进程（kill、exit）,并控制他们之间的通信（signal等）。还包括进程调度，控制活动进程如何共享CPU。这一部分是Linux已经做好的，在写驱动的时候，只需要调用对应的函数即可实现这些功能，例如创建进程、进程通信等等。

MM（Memory Management），内存管理的主要作用是控制多个进程安全的共享内存区域。

VFS（Virtual File Systems），虚拟文件系统，隐藏各种文件系统的具体细节，为文件操作提供统一的接口。在Linux中“一切皆文件”，这些文件就是通过VFS来实现的。Linux提供了一个大的通用模型，使这个模型包含了所有文件系统功能的集合。如下图所示，是一个虚拟文件系统的结构图。



Device Drivers设备驱动，这一部分就是需要学习和掌握的。Linux内核中有大量的代码在设备驱动程序部分，用于控制特定的硬件设备。

Linux驱动一般分为网络设备、块设备、字符设备、杂项设备，需要我们编写的只有字符设备，杂项设备是不容易归类的一种驱动，杂项设备和字符设备有很多重合的地方。

网络协议栈，Linux内核中提供了丰富的网络协议实现。

1.2、Linux内核源代码

1.2.1、如何下载内核源代码

www.kernel.org

1.2.2、目录结构

Linux内核源代码采用树形结构进行组织，非常合理的把功能相关的文件都放在同一个子目录下，使得程序更具有可读性。

1.2.3、代码管理

二、Linux内核配置与编译

2.1、配置内核

2.1.1、为什么要配置内核

1. 硬件的需求
2. 软件的需求

选出需要的，去掉不要的！

2.1.2、如何配置内核

make config

make menuconfig

表示编译进<M>表示编译成模块，需要加载的时候手动加载（节省内存），<*>表示静态编译，也就是编译到内核中，<>表示不编译

2.1.3、内核配置结果

内核配置通常在一个已有的配置文件基础 上，通过修改得到新的配置文件，Linux内核提供了一系列可供参考的内核配置文 件，位于Arch/$cpu/configs

2.2、编译内核

2.2.1、编译内核

make zImage

make bzImage

*区别：在X86平台，zImage只能用于小于512K的内核

*如需获取详细编译信息，可使用：

make zImage V=1

make bzImage V=1

** 编译好的内核位于arch/<cpu>/boot/目录下 **

2.2.2、编译内核模块

make modules

make modules_install //并不是安装，只是把所有的.ko文件移动到/lib/modules下

编译内核模块

将编译好的内核模块，从内核源代码目录复制至/lib/modules下**，为打包做好准备

2.2.3、制作ramdisk（打包文件系统）

方法：mkinitrd initrd-$version $version

例：

mkinitrd initrd-2.6.39 2.6.39

*$version 可以通过查询/lib/modules 下的目录得到

2.3、安装内核（PC机上）

（1）、cp arch/x86/boot/bzImage /boot/vmlinuz-$version

（2）、cp initrd-$version /boot/

（3）、修改/etc/grub.conf

修改前：

```
14 title Red Hat Enterprise Linux (2.6.32-279.el6.i686)
15         root (hd0,0)
16         kernel /vmlinuz-2.6.32-279.el6.i686 ro root=/dev/mapper/VolGroup-lv_root rd_NO_LUKS
LANG=en_US.UTF-8 rd_NO_MD rd_LVM_LV=VolGroup/lv_swap SYSFONT=latarcyrheb     -sun16
crashkernel=128M rd_LVM_LV=VolGroup/lv_root  KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM rhgb quiet
17         initrd /initramfs-2.6.32-279.el6.i686.img
```

修改后：

```
18 title Red Hat Enterprise Linux (2.6.39)
19        root (hd0,0)
20        kernel /vmlinuz-2.6.39 ro root=/dev/mapper/VolGroup-lv_root rd_NO_LUKS LANG=en_US.UTF-8 rd_NO_MD
rd_LVM_LV=VolGroup/lv_swap SYSFONT=latarcyrheb-sun16 crashk     ernel=128M rd_LVM_LV=VolGroup/lv_root
KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM rhgb quiet
21        initrd /initrd-2.6.39
```

在系统启动倒计时时按下回车键，选择新内核启动。

启动之后可用 uname -r 查看运行的内核版本

系统结构	存储位置
用户空间	硬盘/Flash
文件系统	
内核	内存

2.4、清理内核

```
make clean    //清理.o，.ko文件
make distclean    //清理.o，.ko文件，配置文件
```

三、嵌入式Linux内核制作

制作嵌入式平台使用的Linux内核，方法和制作 PC平台的Linux内核基本一致。

3.1、清除原有配置与中间文件

x86: make distclean
arm: make distclean

3.2、配置内核

x86: make menuconfig
arm: make menuconfig ARCH=arm

3.3、编译内核

x86: make bzImage
arm: make uImage ARCH=arm CROSS_COMPILE=arm-linux-