# 目录

[目录 1](#_Toc19709)

[笔记资源依赖网站 4](#_Toc19534)

[第1季-嵌入式的童年(操作系统,编程语言,开发板基础) 6](#_Toc25017)

[第\*\*课-VMware和虚拟机系统 6](#_Toc3022)

[第\*\*课-Linux磁盘分区管理 9](#_Toc8514)

[第\*\*课-vim编辑器 15](#_Toc28098)

[第\*\*课-Linux网络配置 83](#_Toc18909)

[第\*\*课-远程登录Linux系统 86](#_Toc20245)

[第\*\*课-Windows与Linux共享 92](#_Toc26571)

[第\*\*课-TFTP与NFS服务器搭建 94](#_Toc15483)

[第\*\*课-Linux在线升级管理yum 96](#_Toc15455)

[第\*\*课-sudo用户管理 99](#_Toc29575)

[第\*\*课-Linux密码故障排除 100](#_Toc22408)

[第\*\*课-GCC程序编译 101](#_Toc17741)

[第\*\*课-开发板介绍 102](#_Toc19476)

[第\*\*课-开发板串口连接 103](#_Toc12075)

[第\*\*课-开发板系统安装准备 103](#_Toc10383)

[第\*\*课-安装Linux到开发板 104](#_Toc31132)

[第\*\*课-C语言基础与扩展 104](#_Toc7991)

[让你不再害怕指针(201612061859) 105](#_Toc16084)

[第\*\*课-C语言底层原理 109](#_Toc2719)

[C语言基础(传智播客扫地僧) 118](#_Toc19196)

[C语言提高(传智播客扫地僧) 118](#_Toc1114)

[C++基础(传智播客扫地僧) 123](#_Toc30669)

[C++进阶(传智播客扫地僧) 137](#_Toc20995)

[C和C++与数据结构(传智播客扫地僧) 137](#_Toc11299)

[C和C++与设计模式基础(传智播客扫地僧) 141](#_Toc4138)

[第\*\*课-逆向工程基础 141](#_Toc18131)

[第\*\*课-英语基础(单词量) 165](#_Toc3275)

[第2季-小应用大智慧(Linux应用编程)-上学期 336](#_Toc1657)

[第01课-课程规划与学习方法 336](#_Toc23135)

[第02课-GDB程序调试 336](#_Toc28876)

[第03课-Coredump程序故障分析 337](#_Toc14426)

[第04课-Linux应用程序地址布局 339](#_Toc15777)

[第05课-Linux编程规范 343](#_Toc25764)

[第06课-静态函数库设计 343](#_Toc11015)

[第07课-动态函数库设计 344](#_Toc17870)

[第08课-系统调用方式文件编程 345](#_Toc2836)

[第09课-库函数方式文件编程 351](#_Toc7196)

[第10课-时间编程 355](#_Toc8551)

[第11课-进程控制理论 360](#_Toc6032)

[第12课-多进程程序设计 362](#_Toc21863)

[第13课-无名管道通讯编程 366](#_Toc25499)

[第14课-有名管道通讯编程 369](#_Toc10180)

[第15课-信号通讯编程 373](#_Toc32041)

[第16课-信号量互斥编程 378](#_Toc3277)

[第17课-信号量同步编程 387](#_Toc3607)

[第二季-小应用大智慧(Linux应用编程)-下学期 396](#_Toc6798)

[第18课-共享内存通讯 396](#_Toc12508)

[第19课-消息队列编程 404](#_Toc12979)

[第20课-多线程程序设计 415](#_Toc7777)

[第21课-多线程同步 426](#_Toc25492)

[第22课-网络协议分析 428](#_Toc22776)

[第23课-Linux网络编程模型 431](#_Toc20675)

[第24课-TCP通讯程序设计 432](#_Toc11431)

[第25课-UDP通讯程序设计 447](#_Toc11677)

[第26课-网络并发服务器设计 450](#_Toc4488)

[第\*\*课-高并发epoll服务器设计 458](#_Toc851)

[第27课-守护进程设计 465](#_Toc8061)

[第28课-Shell脚本高级编程 468](#_Toc22480)

[小应用大智慧应用 483](#_Toc9231)

[第\*\*课-github的使用 494](#_Toc19615)

[第三季-裸奔吧ARM(bootloader)-上学期 496](#_Toc9274)

[专题00-课程规划与学习方法 496](#_Toc32249)

[专题01-工欲善其事-必先利其器 498](#_Toc27507)

[专题02-我从内部看ARM 507](#_Toc30603)

[专题03-汇编语言得玩转 511](#_Toc16091)

[专题04-我是bootloader设计师 519](#_Toc16332)

[专题05-核心初始化 536](#_Toc11083)

[专题06-点亮指路灯 545](#_Toc1294)

[专题07-ARM跑快了---时钟初始化 547](#_Toc11637)

[专题08-不用内存怎么行 556](#_Toc32697)

[专题09--代码搬移不可少 564](#_Toc9500)

[专题10-C语言环境初始化 570](#_Toc26500)

[专题11-瞻前顾后 577](#_Toc24645)

[第三季-裸奔吧ARM(bootloader)-下学期 578](#_Toc28423)

[专题01-世界一下变大了-MMU 578](#_Toc27878)

[专题02-通过按键玩中断 588](#_Toc30025)

[专题03-NandFlash变硬盘 629](#_Toc5327)

[专题04-串口来做控制台 685](#_Toc17603)

[专题05-快车道DMA 788](#_Toc22156)

[专题06-LCD亮起来 794](#_Toc26547)

[专题07-触摸屏显身手 803](#_Toc8102)

[专题08-网卡搭建新通道 805](#_Toc14536)

[专题09-移植tftp客户端 821](#_Toc10952)

[专题10-移植bootm命令启动内核 842](#_Toc28783)

[专题11-自我检测:移植USB驱动 853](#_Toc14234)

[第四季-内核驱动奥秘多(内核驱动)-上学期 853](#_Toc21290)

[专题01-课程规划与学习方法 853](#_Toc16629)

[专题02-U-Boot新手入门 854](#_Toc15119)

[实验02-uboot开心的移植 859](#_Toc13184)

[专题03-嵌入式Linux内核制作 893](#_Toc16989)

[专题04-嵌入式文件系统 896](#_Toc26600)

[专题05-内核模块开发 901](#_Toc4301)

[专题06-Linux内核子系统 909](#_Toc22972)

[专题07-Linux内核链表 914](#_Toc17288)

[专题08-Linux系统调用 919](#_Toc18955)

[专题09-Linux驱动开发前奏 921](#_Toc30142)

[专题10-字符设备驱动模型 925](#_Toc21532)

[第四季-内核驱动奥秘多(内核驱动)-下学期 950](#_Toc30482)

[专题01-课程规划与学习方法 950](#_Toc27828)

[专题01-按键驱动程序设计 951](#_Toc26158)

[专题02-总线设备驱动模型 981](#_Toc18636)

[专题03-串口驱动程序设计 1001](#_Toc22851)

[专题04-网卡驱动程序设计 1063](#_Toc20125)

[专题05-触摸屏驱动程序设计 1064](#_Toc19265)

[专题06-LCD驱动程序设计 1064](#_Toc3182)

[专题07-FLASH驱动程序设计 1065](#_Toc31750)

[专题08-I2C驱动程序设计 1065](#_Toc8922)

[专题09-SPI驱动程序设计 1066](#_Toc14956)

[专题10-USB驱动程序设计 1067](#_Toc3566)

[专题11-CAN总线驱动程序设计 1073](#_Toc4722)

[专题\*\*-Linux总线驱动总体框架总结 1074](#_Toc24252)

[第五季-小试牛刀-项目开发 1077](#_Toc10125)

[阶段1-岗前培训(1~3m) 1077](#_Toc21112)

[嵌入式项目：移动物体监控系统 1079](#_Toc22096)

[嵌入式项目：网络安全传输系统 1086](#_Toc16112)

[嵌入式项目：基于视频压缩的实时监控系统 1086](#_Toc25287)

[嵌入式项目：基于ARM9的广告投放终端项目 1086](#_Toc28560)

[嵌入式项目：WEB网络MP3终端项目 1087](#_Toc27641)

[嵌入式项目：GPS电子地图项目 1087](#_Toc11110)

[嵌入式项目：手机远程的智能家具终端项目 1087](#_Toc14421)

[第六季-疑难汇总 1088](#_Toc13398)

[第七季-我的运筹帷幄 1113](#_Toc2716)

# 笔记资源依赖网站

Embedded Studio:

<https://www.segger.com/>

Dr Chip’s Vim Page:

[http://www.drchip.org/astronaut/vim/index.html#SYNTAX\_TAGS](http://www.drchip.org/astronaut/vim/index.html" \l "SYNTAX_TAGS)

[51CTO.COM - 技术成就梦想 - 中国领先的IT技术网站](https://www.baidu.com/link?url=SsAgLBwXygnzGQt01kWVvZlbvCXT3PyzeGRE1FdYb87&wd=&eqid=c926050e00006c2100000004587c78f9" \t "_blank)

驱动博客园笔记：<http://www.cnblogs.com/hello2mhb/p/3341291.html>

Linux入门教程：<http://www.92csz.com/study/linux/>

package资源网站：<https://pkgs.org/download/libc.so.6(GLIBC_2.15)>

rpm包资源网站:

<http://rpmfind.net/linux/rpm2html/search.php?query=libc.so.6%28GLIBC_2.15%29%2864bit%29&submit=Search+...&system=&arch>=

Linux软件包搜索站:

<http://hpux.connect.org.uk/hppd/cgi-bin/search?term=vim&Search=Search>

威客中国<http://www.vikecn.com/>

媒体播放器：<http://smplayer.sourceforge.net/zh/downloads>

压缩工具rar:<http://www.rarsoft.com/download.htm>

压缩工具tar:<http://www.gnu.org/software/tar/>

ChinaUnix官网：<http://www.chinaunix.net/> (源码包方式下载:mirrors)

gcc编译器官网：<https://gcc.gnu.org/>

安装包资源：<https://pkgs.org/download/libstdc++.so.6>

不学网资源：<http://pan.noxue.com/>

C开源中国社区:<https://www.oschina.net/>

CSDN下载官网:<http://download.csdn.net/>

第三方登录:1989291498

开源中国昵称:seafly0616

开源中国手机:17769260939

开源中国第三方账号(QQ):1052061602

关键字:wps for linux: <http://bbs.wps.cn/forum-wpslinux-1.html>

<http://wps-community.org/download.html>

wps社区登录网址: <http://bbs.wps.cn/>

第三方登录:1989291498

音视频编解码技术资源：

<http://blog.csdn.net/leixiaohua1020/article/details/15811977/>

<http://blog.csdn.net/leixiaohua1020/>

Linux内核源代码下载：<https://www.kernel.org/>

Uboot源码网站：ftp://ftp.denx.de/pub/u-boot/

leangoo敏捷项目协作工具：<https://www.leangoo.com/>

leangoo邮箱登陆: seafly0616@qq.com

技术博客(CSDN博客频道)：<http://blog.csdn.net/lmj623565791>

登录(第三方(QQ)):1052061602

安卓CSDN博客专家：

英文强化博客：<http://blog.csdn.net/hongdeng123>

<http://blog.csdn.net/sbsujjbcy> //开源Linux技术交流群主

<http://blog.csdn.net/lmj623565791>

<http://blog.csdn.net/sinyu890807>

<http://blog.csdn.net/jdsjlzx>

<http://blog.csdn.net/harvic880925>

<http://blog.csdn.net/lfdfhl>

<http://blog.csdn.net/lpjishu>

<http://blog.csdn.net/hejjunlin>

<http://blog.csdn.net/iwanghang>

IOS博客专家:

<http://blog.csdn.net/qq_31810357>

Qt博客专家:

<http://blog.csdn.net/u011012932>

媒体解码技术博客专家:

<http://blog.csdn.net/shaqoneal>

# 第1季-嵌入式的童年(操作系统,编程语言,开发板基础)

# 第\*\*课-VMware和虚拟机系统

## 老毛桃PE系统使用经验

### 拆分分区

打开分区工具后，鼠标选中D分区，然后右键选择<拆分分区>，然后根据自己情况分出新分区E分区

### 合并分区

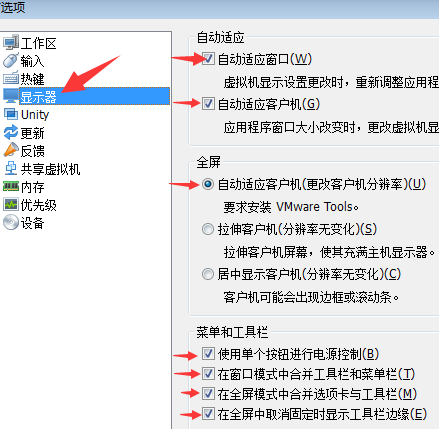
打开分区工具，鼠标选中E分区，然后右键选择<拆分分区>，然后选择合并到D分区.

## VMware的虚拟机桌面显示调整设置

01.首先选择“首选项”如下：

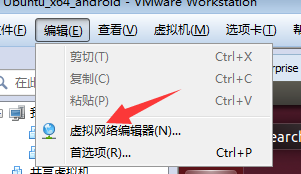


在显示器选项中按照如下设置：



## VMware有线网络和无线网络的VMware配置

首先打开“虚拟网络编辑器”如下：



如果是您的物理机是连接无线网的话，就要手动将“自动”改为你的无线网卡：



## vmware虚拟机怎么克隆

首先让虚拟机处于关机状态，然后开始下面的克隆工作：

>>“虚拟机”>>

>>“管理”>>

>>“克隆”>>

>>“创建完整克隆”>>

>>“虚拟机名称”（克隆后的虚拟机名称）>>

>>“位置”（克隆后的保存位置）>>

## 第\*\*课-自己动手写U盘启动系统

**NOTE:本课要有一定Linux启动流程基础**

第一步,Linux中插上U盘之后umount该U盘

第二步,用fdisk对U盘分区,这里必须有主分区和逻辑分区,这是前辈的经验.

第三步,将分区后的主分区和逻辑分区都格式化为ext2格式.经验告诉我要分2个以上区。

第四步,挂载U盘到/mnt/usb

第五步,拷贝宿主Linux的stage:cp /boot/grub/stag\* /mnt/usb/boot/grub

第6步,在/mnt/usb下输入grub命令进入grub命令行

grub> help

grub> root (hd1,0)

grub> setup (hd1)

grub> quit

第7步,使用busybox制作根文件系统,进入配置menuconfig

使用静态binary

不使用/usr

## 第\*\*课-移植Linux到U盘

第一步，准备一个U盘并分一个区并格式化为ext3格式

第二步，打包Linux根目录下的所有文件:**tar –czvpf root.tar.gz / --exclude=/sys/\* --exclude=/proc/\* --exclude=/tmp/\* /tmp /proc /sys /bin**

第三步，挂载U盘到/mnt:**mount /dev/sdb1 /mnt**

第四部，把打包好的压缩包复制到/mnt下并解压:**tar –xzvf root.tar.gz**

第五部，制作grub MBR:**grub –install --root-directory=/mnt/dev/sda**

第六部，修改/etc/fstab:把LABEL=/1改为/dev/sda1

第七部，修改/boot/grub.conf:root=LABEL=/1改为root=/dev/sda1

第八部，启动U盘Linux进入单用户模式shell命令行

# mount –o remount.rw /

# vim /etc/sylinux/config将SELINUX设置为disable关闭selinux

# blkid -s LABEL 查看LABEL命令

#e2abel /dev/sda1 /1设置LABEL的命令

# 第\*\*课-Linux磁盘分区管理

## 磁盘基本管理

查看已挂载分区的使用情况:df

[root@redhat6 Desktop]# df -h

Filesystem Size Used Avail Use% Mounted on

/dev/mapper/vg\_redhat6-lv\_root

16G 14G 1.5G 90% /

tmpfs 948M 264K 947M 1% /dev/shm

/dev/sda1 485M 36M 425M 8% /boot

[root@redhat6 Desktop]#

查看某挂载目录的总大小:du

[root@redhat6 Desktop]# du -sh /home/redhat6/

6.7G /home/redhat6/

[root@redhat6 Desktop]#

## 磁盘的LVM管理

### 核心理论

硬盘类型/dev/sda VS /dev/hda

HDA是使用了IDE接口的硬盘的名称; SDA是SATA接口的硬盘的名称;在最新的2.6.19内核里,所有的硬盘都叫SDA了。

硬盘接口类型

IDE(Integrated Driver electronics)

SATA(Serial ATA(Advanced Technology attachment))

SCSI(Small Computer System Interface)

LVM概述

LVM是Linux系统中对磁盘分区进行管理的一种逻辑机制，它是建立在硬盘和分区之上的，文件系统之下的一个逻辑层，在建立文件系统时屏蔽了下层的磁盘分区布局，能够在保持现有数据不变的情况下动态调整磁盘容量，从而提高磁盘管理的灵活性。

PV(Physical Volume, 物理卷)

物理卷是LVM机制的基本存储设备，通常对应为一个普通分区或整个磁盘。创建物理卷时会在分区或硬盘的头部创建一个保留区块，用于记录LVM的属性，并把存储空间分割成默认大小为4MB的基本单元PE，从而构成物理卷。物理卷一般直接使用设备文件名称，如/dev/sdb1, /dev/sdb2等.

用于转换成物理卷的普通分区，建议先使用fdisk工具将分区类型的ID标记号改为8e。

若是整块硬盘，先将所有磁盘空间划分为一个主分区后再做相应调整。

VG(Volume Group, 卷组)

由一个或多个物理卷组成一个整体，管理一个或多个物理卷的一种架构。在卷组中可以动态添加或移除物理卷。许多物理卷可以分别组成不同的卷组，卷组名称由用户自行定义。

LV(Logical Volume, 逻辑卷)

逻辑卷是建立在卷组之上的，与物理卷没有直接关系。对于逻辑卷来说，每一个卷组就是一个整体，从这个整体中切出一小块空间，作为用户创建文件系统的基础，这一小块空间就称为逻辑卷。使用mkfs等工具在逻辑卷上创建文件系统以后，即可挂载到Linux系统中的目录下使用。

建立LVM分区管理机制的流程

首先，将普通分区或整个硬盘创建为物理卷;接着将物理卷上比较分散的各个物理卷的存储空间组成一个逻辑整体,即卷组;最后,基于卷组这个整体,分割出不同的数据存储空间,形成逻辑卷。而逻辑卷才是最终用户可以格式化并挂载使用的存储单位。

LVM原理图

沙漏模型,最下面是物理卷,中间漏点是卷组,最上层是逻辑卷。

### LVM管理命令

|  |  |  |  |
| --- | --- | --- | --- |
| 功能 | 物理卷管理 | 卷组管理 | 逻辑卷管理 |
| scan 扫描 | pvscan | vgscan | lvscan |
| create 建立 | pvcreate | vgcreate | lvcreate |
| display | pvdisplay | vgdisplay | lvdisplay |
| remove | pvremove | vgremove | lvremove |
| extend | pvextend | vgextend | lvextend |
| reduce | pvreduce | vgreduce | lvreduce |

### 主要命令的语法

# pvcreate 设备名

# vgcreate 卷组名 物理卷名1 物理卷名2 ...

# lvcreate -L 大小 -n 逻辑卷名 卷组名

# lvextend -L +大小 /dev/卷组名/逻辑卷名

# resize2fs /dev/vgxxx/lvxxx //修改逻辑卷大小后通过resize2fs重定义(刷新)逻辑卷大小显示

注意事项-LVM管理顺序

对于创建物理卷、创建卷组、创建逻辑卷我们是有严格顺序的，同样对于删除操作我们也有严格顺序:

\*\* 结束正在使用的逻辑卷文件挂载点进程:fuser –k /media/disk1

\*\* 卸载逻辑卷: umount /media/disk1或 umount /dev/vgxxx/lvxxx

\*\* 删除逻辑卷: lvremove命令

\*\* 删除卷组: vgremove命令

\*\* 删除物理卷: pvremove命令

注意事项-查看swap分区是否激活(是否显示容量)

[root@redhat6 Desktop]# free -m

total used free shared buffers cached

Mem: 1894 646 1248 0 154 241

-/+ buffers/cache: 250 1644

Swap: 4063 0 4063

[root@redhat6 Desktop]#

[root@redhat6 Desktop]# mkswap /dev/sdaX//如果swap分区未激活,则通过此命令激活swap分区

[root@redhat6 Desktop]# vim /etc/fstab //然后根据新UUID修改fstab中的swap分区的UUID编码

[root@redhat6 Desktop]# 最后重启生效

### LVM使用实例

公司准备在Internet中搭建邮件服务器。但由于用户数量众多，邮件存储需要大量的空间，考虑到动态扩容的需要，计划增加两块SCSI硬盘并构建LVM逻辑卷(挂载到/mail目录下)专门用于存放邮件数据。

根据上述案例环境和需求，具体操作步骤如下:

\*\* 关闭服务器，打开机箱，正确挂接2块SCSI新硬盘。

\*\* 开启服务器主机，并执行fdisk –l命令进行检查,确认已识别新增的硬盘(sdb, sdc)

\*\* 每个硬盘所有空间划分为一个独立的主分区,并将分区类型更改为8e。分好后使用fdisk -l查看结果。

\*\* 将/dev/sdb1, /dev/sdc1分区转换为物理卷。

\*\* 将上述两个物理卷整合,创建名为mail\_store的卷组。

\*\* 在mail\_store卷组中创建一个名为mail的逻辑卷,容量大小设置为3G。

\*\* 使用mkfs命令在mail逻辑卷中创建ext3文件系统,并挂载到/mail目录下。

\*\* 动态扩展mail逻辑卷的容量(+500MB),并更新系统识别的文件系统大小。

### 光驱(外存盘)系统管理

### 从光盘制作光盘镜像文件(制作iso文件)

# cp /dev/cdrom /home/redhat6/disk.iso 或

# dd if=/dev/cdrom of=/home/redhat6/disk.iso

### 将文件和目录制作成光盘文件

# mkisofs –r –J –V mydisk –o /home/redhat6/mydisk.iso /home/redhat6/mydir

注:将mydir目录制作成mydisk.iso文件,光盘卷标为mydisk。

### 查看移动盘文件系统类型

[root at pldyrouter /]# fdisk -l  
　　/dev/sdc1 1 510 4096543+ 7 HPFS/NTFS   
　　/dev/sdc2 511 4864 34973505 f Win95 Ext'd (LBA)   
　　/dev/sdc5 511 4864 34973473+ b Win95 FAT32

### 挂载iso镜像文件

# mount –t iso9660 -o loop /home/redhat6/mydisk.iso /media/isodir

### 挂载一般的U盘

# mount –t msdos /dev/sdb5 /media/udisk 或

# mount –t vfat /dev/sdb5 /media/udisk

### 支持汉字文件名的挂载方式

# mount -t vfat -o iocharset=cp936 /dev/sdc5 /media/udisk 或

# mount –t vfat –o iocharset=utf8 /dev/sdc5 /media/udisk

### Linux磁盘坏道检测

坏道的定义

它是一块磁盘或者闪存上不能再被读写的部分，一般是由于磁盘表面特定的物理损坏或闪存晶体管失效导致的。

随着坏道的继续积累，它们会对你的磁盘或闪存容量产生令人不快或破坏性的影响，甚至可能会导致硬件失效。

同时还需要注意的是坏道的存在警示你应该考虑更换新磁盘了，或简单地将坏块标记为不可用。

使用系统自带的磁盘检查工具

1. sudo fdisk –l sda
2. sudo badblocks -v /dev/sda5 >badsectors.txt
3. 如果你发现任何坏道，紧接着走下一步
4. ---------------------------针对ext3/ext3/ext4文件系统--------------------------

$ sudo e2fsck -l badsectores.txt /dev/sda5 #-l告诉命令将指定山区号码加入坏道列表

----------------------------针对其他文件系统-------------------------------------

$ sudo fsck -l badsectors.txt /dev/sda5

利用smartmontools工具扫描坏道(第三方软件)

这个方法对带有S.M.A.R.T(自我监控分析报告技术)系统的现代磁盘(ATA/SATA和SCSI/SAS硬盘以及固态硬盘)更加可靠和高效。S.M.A.R.T系统能够帮助检测,报告,以及可能影响它们的健康状况,这样你就可以找出任何可能出现的硬件失效。

$ man smartctl 或 smartctl -h

$ sudo smartctl -H /dev/sda 或 sudo smartctl --health /dev/sda

# 第\*\*课-vim编辑器

## 启动vim时的操作

vim 启动vim

vim file 用vim打开file这个文件

vim -R file 只读方式打开文件

进入vim后

:help {command}

学会经常使用ESC键，ESC进入命令模式,ESC刷新操作。

:edit newfile 在vim中打开一个文件

## vim编辑时的操作

进入编辑模式

插入字符模式:i

新插入一行,在光标所在行上面一行新增一行:O

新插入一行,在光标所在行下面一行新增一行:o

光标处附加内容:a (append)

常用的就如下几种进入编辑模式的方式：

一般直接输入i就进入，输入之后就从该光标前插入。

也可以输入a，输入之后就从该光标后插入。

也可以输入大写A，输入之后就从该行末尾插入。

也可以输入o，输入之后就在该行下新增一行并把光标移到新增行开头插入。

也可以输入大写的O，输入之后就在该行上方新增一行并把光标移到新增行开头插入。

具体什么效果还要自己去跟着上面做试验去体验一把印象更深刻。

选择处理内容

块选择:<Ctrl-v> {vim方向键}

选择一行中某个字符或字符串:<Ctrl-v> {vim方向键}

选择某几行:大写V {vim上下键}

全选该文件内容:V G

复制选中内容:yw

剪切选中内容:c

删除选中内容:x或d

删除该行：dd

删除该词：dw （执行之前先定位到词首字符）

粘贴选中内容:p

粘贴选中内容3次:3p

合并当前行和下一行内容:J

左右缩进: < 或 >

自动缩进: =

撤销操作：u （撤销操作）

恢复撤销：Ctrl-r （反撤销操作）

nx 连续向后删除n个字符

d$ 删除光标到行尾位置内容

d0 删除光标到行首位置内容

5dd 连续向下删除5行内容

d1G 删除光标到第一行所有数据

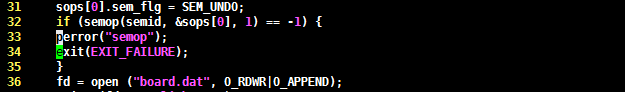
dG 删除光标到最后行的所有数据

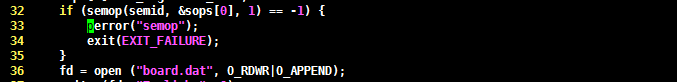
0...^...$ 分别定位到行首，行首非空上，行尾。

Ctrl-b 向前翻页

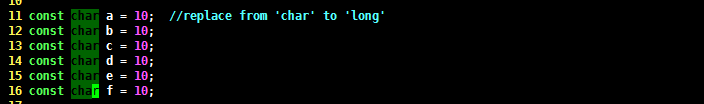
Ctrl-f 向后翻页

批量自动缩进：首先Ctrl-v选中要缩进的列块，然后直接按=即可批量缩进，如下图：

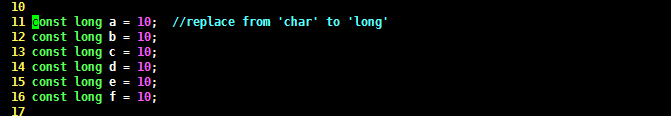




批量替换列块信息：首先Ctrl-v选中需要替换的列，如我们要替换的char列块，如图：



紧接着按c，然后输入你要替换的新字符串，然后按ESC，即可替换成功，如下图：



## vim定位命令

光标定位操作

vim中的方向键:h(L-左), j(D-下), k(U-上), l(R-右) 推荐使用字母来移动光标

一行内容中光标定位操作:0.....^.......$:分别表示定位到行首,非空,行尾。

定位到某行:命令模式下直接输入行号即可.例如,:12表示定位到第12行。

定位到第1行: :1

定位到第6行: :6

定位到文件末尾: G

到下一个单词开头: w 或 W

到下一个单词结尾: e 或 E

到前一个单词开头：b

定位到当前编辑区页首：H

定位到当前编辑区页中：M

匹配括号移动: %

匹配光标当前所在的单词: # 和 \* 分别是向上一个匹配和向下一个匹配

<start position><command><end position>

从当前位置拷贝到本单词的最后一个字符: ye

拷贝两个foo之间的字符串: y2/foo

字母改变大小写: ~

**h：左移一个字符  
j：下一行  
k：上一行  
l：右移一个字符  
$：移动到行尾  
0：移动到行首  
H：移至屏幕上方  
M：移至屏幕中间  
L：移至屏幕下方**

**：set  nu ：设置显示文本的行数  
：set  nonu：取消显示文本的行数  
gg：移到文件的行首  
G：移到最后一行  
：n：移到文件的第n行（与nG的功能相同）**

文件缓冲区定位：

比如我们的test.c在缓冲区标签为[283:test.c]

我们要跳转到该标签可以：

:b 283<CR>即可，或者

:b test.c<Tab键盘>

再或者如果是多个同名但不同目录的标签文件，可以这样：

:LUBufs<CR> 然后输入你要找的标签名并按Ctrl-n或Ctrl-p选择，回车打开。

撤销/反撤销

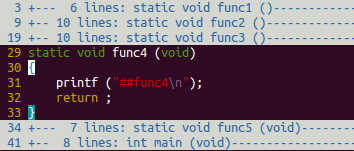
u 表示撤销(undo)

<Ctrl-r>表示反撤销(redo)

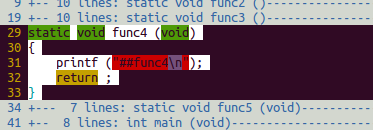
## 代码折叠相关操作

创建折叠的一般需求：

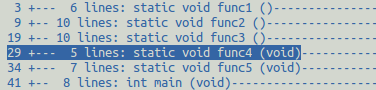
一般代码折叠用在函数上，其他结构体什么的尽量不要用折叠，因为这个折叠是有一个隐含的前提格式要求的，函数的折叠必须要让{}单独占用两行。如下：



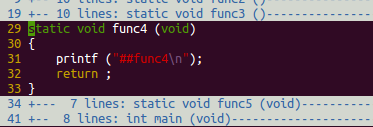
上面的{}我们都是单独让它占用一行，然后我们来创建折叠来折叠它，创建折叠方式有如下几种，这里本人总结的比较好用的有这两种，第一种就是选中整个函数代码块，然后输入zf命令折叠它，如下：先选中整个函数代码块：



然后输入zf折叠它，如下：

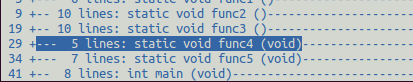


另一种创建折叠的方式下面也提到了，我们先在这里告诉大家，就是直接通过一个命令来折叠这个函数，如下，先确定函数所在行数，如下的函数代码块在29~33行，所以我们可以直接执行如下命令折叠这块内容：





执行后效果如下：



剩下的就是一些折叠的打开和关闭操作以及删除等，下面黄色标签的为本人认为比较常用的折叠操作。具体操作细节可以参考下列标签来进行尝试体会。

折叠命令：

zf 创建折叠，可以按照前面的方式进行折叠，也可以选中代码后进行折叠。

zF 在当前行创建折叠。当一开始就计划要折叠所写代码的时候，可以用该命令创建一对折叠符号，然后再往里面填写内容。

:5,10fo 在vim中运行该命令会在折叠 5-10 行中的代码。

zd 删除光标下的折叠。

zD 删除光标下的折叠，以及嵌套的折叠。

zE 删除窗口内的所有折叠。仅当 manual 和 marker 折叠方法下有效。

打开和关闭折叠：

zo 打开光标下的折叠。

zO 打开光标下的折叠，以及嵌套的折叠。

zc 关闭光标下的折叠。

zC 关闭光标下的折叠，以及嵌套的折叠。

za 当光标在关闭折叠上时，打开之。在打开折叠上时，关闭之。

zA 和za类似，不过对当前折叠和其嵌套折叠都有效。

zv 打开当前光标所在折叠，仅打开足够的折叠使光标所在的行不被折叠。

zr和zm 一层一层打开折叠和一层一层关闭折叠，这两个命令会递减和递增一个叫foldlevel的变量。如果你发现zm>和zr不灵了，那有可能是你连续按的zr或zm次数多了，只要多按几次让foldlevel回到正常状态即可。执行以下zR和zM也>可直接让foldlevel回到正常状态。

zR和zM 打开所有折叠，设置foldlevel为最高级别。关闭所有折叠，设置foldlevel为0。

在折叠间移动：

[z 到当前打开折叠的开始。如果已在开始处，移到包含这个折叠的折叠开始处。

]z 到当前打开折叠的结束。如果已在结束处，移到包含这个折叠的折叠结束处。

zj 把光标移动到下一个折叠的开始处。

zk 把光标移动到前一个折叠的结束处。

## vim常用窗口管理

显示行号：set number

取消行号：set nonumber

如何消除搜索后的关键字高亮,键入:noh或者**:nohlsearch**

如何取消自动缩进**:set noautoindent**

编辑区(上下)分割窗口: <Ctrl-w> s

编辑区(左右)分割窗口: <Ctrl-w> v

切换分割窗口焦点: <Ctrl-w> {hjkl}

增大/减小分割窗口尺寸: <Ctrl-w> {+或\_}

增大/减小字体大小: Ctrl {+或-}

多文件多窗口一次性保存退出

:qall (全部退出)。如果任何一个窗口没有存盘，Vim 都不会退出。除非:qall!

:wall 这表示 "write all" (全部保存)。但实际上，它只会保存修改过的文件。

打开新窗口最简单的命令如下：

:split (水平分割) || :vsplit (垂直分割)

这个命令把屏幕分解成两个窗口并把光标置于上面的窗口中.

移动窗口

如果在分割多个文件时候发现文件顺序不是你所期望的.通过如下来更改

CTRL-W K 会使当前窗口移动到上面并扩展到整屏的宽度。

CTRL-W H 把当前窗口移到最左边

CTRL-W J 把当前窗口移到最下边

CTRL-W L 把当前窗口移到最右边

在窗口间游走

在gvim或vim中，在窗口中移动其实非常简单，因为gvim已默认支持鼠标点击来换编辑窗口，而vim中，则可以打开mouse选项，

：set mouse = a 为命令、输入、导航都激活鼠标的使用

我们知道vim的特色就是可以脱离鼠标而工作，所以可以使用vim提供的全套导航命令，在会话中快速而准确的移动编辑窗口。

按住Ctrl + W，然后再加上h, j, k, l，分别表示向左、下、上、右移动窗口

Ctrl + w + h：向左移动窗口

Ctrl + w + j： 向下移动窗口

Ctrl + w + j： 向上移动窗口

Ctrl + w + l： 向右移动窗口

Ctrl + w + w：这个命令会在所有窗口中循环移动

Ctrl + w + t：移动到最左上角的窗口

Ctrl + w + b：移动到最右下角的窗口

Ctrl + w + p：移动到前一个访问的窗口

关闭所有其他窗口“:only”

这个命令关闭除当前窗口外的所有窗口。如果要关闭的窗口中有一个没有存盘，Vim 会显

示一个错误信息，并且那个窗口不会被关闭。

用分割窗口打开指定的文件(two.c) " :split two.c"

用分割形式新建文件" :new "

关闭当前窗口:":close"

任何退出编辑的命令都可以关闭窗口，象 ":quit" 和 "ZZ" 等。但 "close" 可

以避免你在剩下一个窗口的时候不小心退出 Vim 了。(十分重要的设定啊～)

移动窗口

vim中有两种移动窗口方式，一种只是简单地在屏幕上切换窗口，尺寸维持不变；另一种则是改变窗口的实际布局，还会调整尺寸，以填充它移动的位置。

1. 移动窗口本身（轮换或交换）

Ctrl + w + r：向右或向下方交换窗口，而Ctrl + w + R则和它方向相反。

光标会随着窗口而移动

Ctrl + w + x：交换同列或同行的窗口的位置。vim默认交换当前窗口的与它下一个窗口的位置，如果下方没有窗口，则试着与上一个窗口交换位置。亦可在此命令前加上数量，与制定的窗口交换位置。

2. 移动窗口并改变其布局

注：下面的第三个字母都是大写，按玩Ctrl + w之后，按shif + 相应的字母。后面用^代替Ctrl

Ctrl + w + K ：移动当前窗口至屏幕顶端，并占用全部宽度

^WJ： 移动窗口至屏幕底端，并占用全部宽度

^WH：移动窗口至屏幕左端，并占用全部高度

^WL：移动窗口至屏幕右端，并占用全部高度

^WT ：移动窗口至屏新的现有分页

3. 调整窗口尺寸

改变当前窗口的尺寸，同时当然也会影响到其他窗口。

在gvim和vim中，可以用鼠标点击窗口的顶部白色条并窗口直接调整尺寸。

也可以直接用命令，调整尺寸命令也是以Ctrl + W开头：

Ctrl + W + = ：让所有窗口调整至相同尺寸（平均划分）

Ctrl + W + -：将当前窗口的高度减少一行，也可在ex命令中，：resize -4明确指定减少的尺寸

Ctrl + W + +：将当前窗口的高度增加一行。同样在ex命令中，：resize +n 明确指定增加尺寸

Ctrl + W + < ：将当前窗口的宽度减少

Ctrl + W + > ：将当前窗口的宽度增加

Ctrl + W + |：将当前窗口的宽度调到最大，也可他哦你通过ex命令：vertical resize n明确指定改变宽度

关闭与离开窗口

有4种关闭窗口的方式，分别是：离开（quit）、关闭（close）、隐藏（hide）、关闭其他窗口

^代表Ctrl键

^Wq，离开当前窗口

^Wc，关闭当前的窗口

^Wo，关闭当前窗口以外的所有窗口

## 常用字符串设置

查找时忽略大小写：set ic

查找时不忽略大小写：set noic

iconv转换文件编码问题

工具集通过关键字: **iconv编码转换相关**找到该目录

基本语法：

iconv -f encoding [-t encoding] [inputfile]...

**-f encoding :把字符从encoding编码开始转换。**

**-t encoding :把字符转换到encoding编码。**

**-l :列出已知的编码字符集合**

**-o file :指定输出文件**

**-c :忽略输出的非法字符**

**-s :禁止警告信息，但不是错误信息**

**--verbose :显示进度信息**

**-f和-t所能指定的合法字符在-l选项的命令里面都列出来了。**

查看文件编码：vim中直接:set fileencoding即可显示文件编码格式

如果你想解决vim查看文件乱码的问题：编辑.vimrc

set encoding=utf-8 fileencodings=ucs-bom,utf-8,cp936

\*列出当前支持的字符编码：

#iconv -l

\*将文件file1转码,转后文件输出到fil2中：

#iconv -f EUC-JP-MS -t UTF-8 file1 -o file2 //没-o那么会输出到标准输出.

## 和Linux系统交互(执行外部命令)

:!pwd Execute the pwd command, then returns to vi

:!!pwd Execute the pwd command and insert output in file

:sh Temporary returns to Unix

$exit Returns to vi

## vim退出时的操作

**:wq** 保存并退出

**:wq!** 强制保存并退出

**:q!** 强制退出

**:wall 一次性保存所有编辑**

**:wall!**

**:qall 一次性退出所有编辑**

**:qall!**

**:wqall 一次性保存并退出所有编辑**

**:wqall!**

vim项目管理的基本配置

filetype plugin on "支持插件

syntax enable "语法高亮使能

syntax on "语法高亮开启

set number "显示行号

"set nonumber "关闭行号显示

"set nohlsearch "关闭关键字高亮

colorscheme evening "设置颜色主题(此处为夜间模式)

"下面是winmanager, taglist, minibuffer的集成基本配置

let winManagerWindowLayout = 'FileExplorer|TagList'

let Tlist\_Auto\_Highlight\_Tag = 1

let Tlist\_Exit\_OnlyWindow = 1

let Tlist\_Show\_Menu = 1

let Tlist\_Show\_One\_File = 1

let g:bufExplorerMaxHeight=30

let g:miniBufExplorerMoreThanOne=0

set cursorline "当前行有下划线

winmanager的配置：编辑.vimrc:

用winmanager时，FileExplorer窗口经常会自动变小，有时甚至会被Taglist窗口压缩得看不见。

可以通过如下设置解决：

    设置**taglistbuffer**的最高限制：

    let g:bufExplorerMaxHeight=30

    如果有**minibuffer**插件还需要设置：

let g:miniBufExplorerMoreThanOne=0

python-dev如何安装 sudo apt-get install python-dev 行不通

用aptitude 工具可以搞定  
先：sudo apt-get install aptitude   
然后：  
sudo aptitude install python-dev  
下列“新”软件包将被安装。   
 python-dev python2.7-dev{ab}   
0 个软件包被升级，新安装 2 个， 0 个将被删除， 同时 0 个将不升级。  
需要获取 29.5 MB 的存档。 解包后将要使用 39.3 MB。  
下列软件包存在未满足的依赖关系：  
 python2.7-dev : 依赖: python2.7 (= 2.7.3-0ubuntu3) 但是 2.7.3-0ubuntu3.1 已安装。  
 依赖: libpython2.7 (= 2.7.3-0ubuntu3) 但是 2.7.3-0ubuntu3.1 已安装。  
 依赖: libexpat1-dev 但它将不会被安装。  
 依赖: libssl-dev 但它将不会被安装。  
下列动作将解决这些依赖关系：  
  
 保持 下列软件包于其当前版本：  
1) python-dev [未安装的]   
2) python2.7-dev [未安装的]   
  
是否接受该解决方案？[Y/n/q/?] n  
下列动作将解决这些依赖关系：  
  
 安装 下列软件包：   
1) libexpat1-dev [2.0.1-7.2ubuntu1 (precise)]   
2) libssl-dev [1.0.1-4ubuntu3 (precise)]   
3) libssl-doc [1.0.1-4ubuntu3 (precise)]   
  
 降级 下列软件包：   
4) libexpat1 [2.0.1-7.2ubuntu1.1 (now) -> 2.0.1-7.2ubuntu1 (precise)]   
5) libpython2.7 [2.7.3-0ubuntu3.1 (now) -> 2.7.3-0ubuntu3 (precise)]   
6) libssl1.0.0 [1.0.1-4ubuntu5.7 (now) -> 1.0.1-4ubuntu3 (precise)]   
7) python2.7 [2.7.3-0ubuntu3.1 (now) -> 2.7.3-0ubuntu3 (precise)]   
8) python2.7-minimal [2.7.3-0ubuntu3.1 (now) -> 2.7.3-0ubuntu3 (precise)]  
  
是否接受该解决方案？[Y/n/q/?] y  
下列软件包将被“降级”：  
 libexpat1 libpython2.7 libssl1.0.0 python2.7 python2.7-minimal   
下列“新”软件包将被安装。  
 libexpat1-dev{a} libssl-dev{a} libssl-doc{a} python-dev python2.7-dev{a}   
0 个软件包被升级，新安装 5 个， 5 个被降级， 0 个将被删除， 同时 0 个将不升级。  
需要获取 39.0 MB 的存档。 解包后将要使用 47.8 MB。

## vimtool参考手册

### 前言

vimtool实质上就是一个vim+插件+配置的工具包，它尽可能实现Windows下的sourceinsight的功能需求，有人说它配置好了比sourceinsight更高效，我个人两个我都喜欢，sourceinsight是Windows下的得力助手，那么vim就是Linux下的得力助手就是vim。vimtool帮你完成大量繁琐的安装配置工作，以优化你的vim工作环境为理念。我们不重复造轮子，但我们可以灵活将这些轮子和部件组装成自己心仪的跑车!

vimtool的原作者seafly起初因为在使用虚拟机学习编程过程中，因为频繁用sourceinsight通过samba访问Linux内的项目代码，其中又要有samba的支持和网络的各种配置，所以作者在考虑Linux下是否也有类似sourceinsight这种工具呢？于是作者开始网上寻找并最终锁定自己熟悉的vim，根据以前的vim使用基础经验加上网络上新的操作的不断累积和完善，最终为自己编写自动化脚本，并打包为以后快速搭建vim环境而使用。是为不想屡次安装配置vim环境的繁琐过程为自己编写的。所以“懒惰使人进步”在这里又似乎变得合理了，哈哈`(\*∩\_∩\*)′。

该脚本的功能非常强大并不断被改进和完善，这里原作者本人也热烈欢迎希望广大vimtool受益者能帮忙完善和改进vimtool让更多使用人群受益。

目前vimtool有开发版和使用版，开发版本唯一明显区别是仅仅就是里面的vimtool参考手册为可编辑和多了一份可编辑的专门用于开发的vimtool开发手册，根据该手册来新增功能的同时也要修改vimtool里面的脚本和配置文件等来实现这个新增功能。该工具包完全免费开源，但是版权必须遵循原作者的版权，这也是尊重原作者劳动成果的一种方式吧。

前面的什么各种插件配置设置什么的大部分工作都已经由vimtool帮你完成了！

但你得有一定Linux使用基础，否则我这个教程对你也就没什么意义了~

vim各种操作中如果不熟悉的跟着下面这些操作自己动手跟着实验会更加容易掌握，印象会更加深刻，所以不熟悉的操作多动手跟着下面练练就OK了。因为我也是这样过来的。原作者QQ：1052061602或1989291498，邮箱：seafly0616@qq.com

下面这些问答操作你值得好好看看，因为在实际项目管理中都是用得到的。

### vimtool安装部分

初次使用者建议在解压后的vimtool目录下首先执行./build\_all help获取简单的安装帮助信息。并先提前浏览本篇文档大致内容，它能帮助你。

直接解压后进入vimtool目录，然后直接执行build\_all文件（没有权限增加执行权限即可）

### vimtool操作部分

为什么我输入部分字符Ctrl-n之后不能补全？

答：说明你使用的这个函数或类或结构体是你新增的，你需要在工程中再次运行object.sh来更新工程依赖文件，然后按F3,F4导入依赖文件。

如何在项目管理同时更新项目依赖文件呢？

vim命令模式下输入:AsyncRun object.sh就可以更新且不影响工程操作，更新过程可以通过打开quickfix窗口查看，更新完成后quickfix窗口会计时你的更新所花费的时间（也就是你这个object.sh命令执行完花费时间），更新完成之后再在vim命令模式下分别按F3和F4来重新导入更新好的依赖文件即可。

那么如何直接在工程中执行object.sh呢？

答：在vim命令模式下输入:AsyncRun object.sh即可。然后你稍等片刻，可以打开quickfix窗口查看进度。

问：如何高亮标签某个变量或函数名？

答：光标移到该变量名或函数名上，然后在vim命令模式下直接输入,m之后1秒钟左右时间，就会高亮标签你要标注的函数名或变量名。除了标注一个，vim能同时标注多个名字，比如我标注了这个变量名是紫色的，那么我标注另一个变量名就是其它颜色了。

问：如何取消高亮标签某个变量或函数名？

答：光标移到该变量名或函数名上，然后在vim命令模式下直接输入,m之后1秒钟左右时间，就会取消高亮标签你要标注的函数名或变量名。

问：通过/或?查找关键字之后这个关键字高亮，怎么取消它高亮？

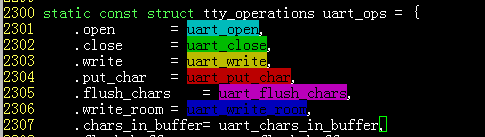
答：这个只需在vim命令模式下输入:set nohlsearch即可。

如果想再次打开高亮，则输入:set hlsearch即可。

请问标签高亮最多同时支持多少个标签高亮呢？

答：目前（2016版本vimtools）最多支持同时6种高亮，分别会以不同背景色区分。

如下图：



能支持代码块标签高亮吗？

答：能，选中代码块(V j)然后进行标签高亮操作(,m)即可。取消高亮必须选中和刚刚加高亮的代码块时一样范围，否则会认为你是选择新的代码块并会加上新的高亮标签。

如何给源文件添加文件头？

答：源文件一般都要添加文件头的，所谓文件头就是表明作者和许可证等信息。文件头类型有Apache许可证类型的，GNU许可证类型的，还有精简版的等等。vimtool配置默认使用的文件头是GNULicense类型的文件头，比如我们添加默认的GNULicense文件头，我们直接在vim命令模式下输入:AddGNULicense或者vim命令模式下直接按F2即可。

如何打开/关闭quickfix窗口？

打开quickfix窗口直接在vim命令模式下输入:copen即可打开。

关闭quickfix窗口直接在vim命令模式下输入:cclose即可打开。

调整quickfix窗口大小直接在vim命令模式下输入:copen 15即可调整为15行大小。

如何在工程中打开某个文件？

方法01：

vim命令模式下输入:edit filename 直接在该编辑区打开某个文件（别用tabedit等tab系列命令！）

方法02：

通过winmanager窗口的左边上面的文件窗口选择你要打开的文件回车打开它。

方法03：

vim命令模式下按F5，然后输入你要打开的文件名。

如何正确退出vimtool管理？

step01 如果打开winmanager那就最先退出winmanager窗口。

step02 如果打开了quickfix窗口那就先退出quickfix窗口。

step03 如果minibuf区有标签就全部必须清空,一个都不剩。(清空前所有文件都保存)

step04 接下来你就可以像退出普通vim那样退出vimtool管理了。

下面是一些插件常用vim命令：

:cscope find g usage\_point #找到usage\_point函数实现处(定义处)

:cscope help 打开cscope帮助信息

:cscope reset 重置所有连接

:cscope show 显示连接

:cscope ./cscope.out ./ 添加一个新的数据库

Ctrl – w – [h|i|j|k] 切换窗口（焦点）

Ctrl + ] #跟踪到宏或变量定义处或函数实现处

Ctrl + t 或 Ctrl+o #返回跟踪

,m 高亮当前word，取消高亮就在当前word处再次执行(可以高亮多词)

F5 或 :LookupFile 从项目中查找并打开某个文件

（lookupfile）Ctrl – n|p 从列表中选择，回车表示确认打开

:LUBufs 从缓冲中查找并打开某个文件

:LUWalk 打开并显示当前目录

<ESC><Ctrl-w> q 退出lookupfile窗口，或直接ESC之后:q退出lookupfile

minibuffer窗口管理

minibuffer文件标签切换:<Tab>

minibuffer文件标签的删除：d

minibuffer在某个编辑区打开特定标签: :bnumber

FileExplorer窗口管理

vim命令模式下按<F5> 更新列表

Taglist窗口管理

vim命令模式下输入u 更新列表

项目管理经验以及注意事项

\*\* 养成闲置时把光标（焦点）切换到编辑区的好习惯: <Ctrl+w> {vim方向键}。

\*\* 打开一个新源文件: 先touch, 然后更新FileExplorer<F5>, 然后从FileExplorer内打开。或者直接:edit newfile

\*\* 其他文件打开: 用edit命令,千万别用tabedit, :edit filename就可以了。

\*\* 在该编辑区内直接跳转到某个标签: bn, 比如跳转到10号标签, :b10

\*\* 全选内容: VG

\*\* 选中光标以下所有内容: VG

\*\* 批量注释源码: <Ctrl+v>选中目标行,然后块右移(>),然后把Tab替换为/(r/)。

\*\* 最后退出工程时: 先退出winmanager(:WMToggle), 然后删除minibuf所有标签, 最后退出vim.

## vimtool插件介绍

### vim插件：quickfix窗口

如何打开quickfix窗口呢？

答：在vim命令模式下输入:copen即可打开quickfix窗口查看生成进度。

如何调整quickfix窗口大小呢？

答：在vim命令模式下输入:copen 15即可调整quickfix显示15行大小。

### 多文件编辑管理：minibuf

MiniBufExplorer现在是vim默认安装的插件了吧。

记住几个常用的命令（从SO上学来的）：

:e <filename> 打开文件

:edit <filename> 打开文件

www.2cto.com

:ls 当前打开的buf

:bn 下一个buf

:bp 前一个buf

:b<n> n是数字，第n个buf

:b<tab> 自动补齐

:bd <n> 删除特定buf

### 索索补全：SearchComplete.vim

" SearchComplete.vim

" Author: Chris Russell

" Version: 1.1

" License: GPL v2.0

"

" Description:

" This script defineds functions and key mappings for Tab completion in

" searches.

"

" Help:

" This script catches the <Tab> character when using the '/' search

" command. Pressing Tab will expand the current partial word to the

" next matching word starting with the partial word.

"

" If you want to match a tab, use the '\t' pattern.

"

" Installation:

" Simply drop this file into your $HOME/.vim/plugin directory.

"

" Changelog:

" 2002-11-08 v1.1

" Convert to unix eol

" 2002-11-05 v1.0

" Initial release

"

" TODO:

"

### 函数参数提示器：code\_complete.vim

created by

Ming Bai

script type

utility

description

Get the latest version from Github:

https://github.com/mbbill/code\_complete

Demo :http://files.myopera.com/mbbill/files/code\_complete.gif

It shows what this script can do.

In insert mode, when you type "<tab>"(default value of g:completekey) after function name with a "(" , function parameters will be append behind, use "<tab>" key again to switch between parameters.

This key is also used to complete code snippets.

Example:

press <tab> after function name and (

foo ( <tab>

becomes:

foo ( `<first param>`,`<second param>` )

press <tab> after code template

if <tab>

becomes:

if( `<...>` )

{

`<...>`

}

install details

1. Put code\_complete.vim to plugin directory.

2. Use the command below to create tags file including signature field.

ctags -R --c-kinds=+p --fields=+S . (配置在object.sh里)

"==================================================

" File: code\_complete.vim

" Brief: function parameter complete, code snippets, and much more.

" Author: Mingbai <mbbill AT gmail DOT com>

" Last Change: 2009-06-09 00:09:03

" Version: 2.9

"

" Install: 1. Put code\_complete.vim to plugin

" directory.

" 2. Use the command below to create tags

" file including signature field.

" ctags -R --c-kinds=+p --fields=+S .

"

" Usage:

" hotkey:

""<tab>" (default value of g:completekey)

" Do all the jobs with this key, see

" example:

" press <tab> after function name and (

" foo ( <tab>

" becomes:

" foo ( `<first param>`,`<second param>` )

" press <tab> after code template

" if <tab>

" becomes:

" if( `<...>` )

" {

" `<...>`

" }

"

"

" variables:

"

" g:completekey

" the key used to complete function

" parameters and key words.

"

" g:rs, g:re

" region start and stop

" you can change them as you like.

"

" g:user\_defined\_snippets

" file name of user defined snippets.

"

" key words:

" see "templates" section.

"==================================================

### Sourceinsight:代码预览:autopreview.vim

created by

Baohua Yang

script type

utility

description

The "autopreview" plugin is a light script plugin for Vim that could automatically preview the definition of variables or functions in a preview window like source insight.

I think this script should be helpful when you're insighting a lot of code files and wanna to see the definition of a new variable or function in the source codes immediately without having to input "ptag" each time.

Taglist plugin is needed for this plugin which is alreasy integrated into the latest version of Gvim.

#Usage

If you add "let g:AutoPreview\_enabled = 1" into your vimrc file, the autopreview window will be open when vim is started; otherwise add "let g:AutoPreview\_enabled = 0" (which is default if you do not set this value) into your vimrc file, the autopreview window will be open only after you press your mapped hotkey.

Any time you press your mapped hotkey, you could enable or disable the autopreview window

install details

If you haven't installed taglist, please install it first: http://vim.sourceforge.net/scripts/script.php?script\_id=273

#1 Download autopreview.vim into your favorite plugin directory or source the script from your .vimrc file (\_vimrc for windows)

#2 Add the following context into your .vimrc file

" Automatically enable the preview function with vim starting

let g:AutoPreview\_enabled =1

" make you could press F5 key to enable or disable the preview window, you can also set to other favorite hotkey here

nnoremap <F5> :AutoPreviewToggle<CR>

inoremap <F5><ESC>:AutoPreviewToggle<CR>i

" set the time(ms) break to refresh the preview window, I recommend 500ms~1000ms with good experience

set updatetime=500

#3 Restart Vim.

(Press F5 to enable the plugin if you set "let g:AutoPreview\_enabled =0")

Just Enjoy It!

用vim打开工程源文件之后先按F3，F4导入工程依赖的数据库文件，首先需要数据库文件，由object.sh生成，直接在工程顶层目录执行object.sh即可生成数据库文件，然后按F8打开sourceinsight集成界面。

### Vim补全功能:智能补全

本节所用的命令的帮助入口

**:help omnicppcomplete.txt**

使用过sourceinsight的人一定对它的自动补全功能印象深刻，在很多的集成开发环境中，也都支持自动补全。vim作为一个出色的编辑器，这样的功能当然不能少啦。而且作为一个通用的编辑器，vim实现的补全功能不仅仅限于对程序的补全，它可以对文件名补全、根据字典进行补全、根据本缓冲区或其它缓冲区类似的内容进行补全、根据文件语法补全等等，它甚至允许用户自己编写函数来实现定制的补全。

确保你已关闭vi兼容模式，并允许进行文件检测：

**set nocp**

**filetype plugin on**

vim的omni补全（智能补全）可以支持多种编程语言，

详参**：help compl-omni-filetypes**。本文中主要介绍C、C++的智能补全。

智能补全会根据文件类型设置不同的补全函数。所以需要打开文件类型检测功能：

**filetype plugin indent on**

使用智能补全还需要ctags5.6版本以上的ctags生成相应的tags文件:**ctags -RV**

这样之后打开vim就可以进行智能补全了。

比如我们输入结构体变量，然后按下**Ctrl-x Ctrl-o**，会弹出一个下拉菜单，显示所有匹配的标签。你可以使用**Ctrl-p Ctrl-n**上下选择，在选择的同时，所选中的项就被放在光标位置，无需再按回车来把它放在光标位置（就像SourceInsight一样）。可以使用Ctrl-E停止补全并回到原来录入的文字。Ctr-y可以停止补全并接受当前选项。

去掉预览窗口显示：**set completeopt=longest,menu**

如果要支持C++的全能补全，就需要OmnicppComplete插件的支持。

下面这句配置是对C++语言的特殊配置：

**ctags –R --c++-kinds=+p --fields=+iaS --extra=+q <YOUR\_SOURCE\_DIR>**

### 语法高亮以及高亮标签:Mark

vim并不是只有黑色两色。正相反，它提供了非常灵活的机制允许用户自定义色彩。运行在终端中的vim，由于终端本身的限制，只能使用若干种固定的颜色；但对于gvim来讲，你可以根据你的喜好调出任意的颜色。

首先，把下面的Ex命令加入你的vimrc，打开vim的语法高亮功能：

**syntax enable**

记得Source Insight中有一个功能，按**SHIFT+F8**可以把光标下的词高亮出来，在看代码时非常有用。vim下也有一个插件可以完成此功能，而且比Source Insight的这个功能强大多了。

created by

Yuheng Xie

script type

utility

description

This script is written to highlight several words in different colors simultaneously. For example, when you are browsing a big program file, you could highlight some key variables. This will make it easier to trace the source code.

I found Ingo Karkat has some further development on mark.vim and you may be interest to check it out:

http://www.vim.org/scripts/script.php?script\_id=2666

mark.vim 1.1.9-g version fanally uses VIM's matchadd(), now highlighting keywords is possible.

Usage:

Highlighting:

Normal mode:

\m mark or unmark the word under (or before) the cursor

Place the cursor under the word to be highlighted, press \m, then the word will be colored.

\r manually input a regular expression

To highlight an arbitrary regular expression, press \r and input the regexp.

\n clear this mark (i.e. the mark under the cursor), or clear all highlighted marks

Visual mode:

\m mark or unmark a visual selection

Select some text in Visual mode, press \m, then the selection will be colored.

\r manually input a regular expression (base on the selection text)

Command line:

:Mark regexp to mark a regular expression

:Mark regexp with exactly the same regexp to unmark it

:Mark to clear all marks

Searching:

Normal mode:

\* # \\* \# \/ \? use these six keys to jump to the other marks

and you could also use VIM's / and ? to search, since the mark patterns have

been added to the search history.

If you want to use another leader instead of default "\", you may say in your .vimrc

let mapleader = ","

That will use **,m ,r ,n instead of \m \r \n** as the trigger.

### 工程管理基础部件:ctags插件

ctags 在使用vim编程和浏览代码是非常有用。可以用CTRL+]和CTRL+t 来回跳转关键字。

先生成自己工作目录的tags。最简单粗暴用法：

**$ cd yourdir**

**$ ctags –R \***

这样会生成一个tags文件。  
不过，这种有个问题，成员变量没有包含在里面。所以自动完成对象的成员时没有提示。  
解决办法：

**$ ctags –R --fields=+iaS --extra=+q \***

–fields=[+|-]flags  
–fields指定tags的可用扩展域（extension fields），以包含到tags入口。  
i:继承信息Inheritance information  
a：类成员的访问控制信息 Access (or export) of class members  
S： 常规签名信息，如原型或参数表 Signature of routine(e.g. prototype or parameter list)  
–extra=[+|-]flags  
指定是否包含某种扩展信息到tags入口。  
q：包含类成员信息（如c++,java,Eiffel)。  
但就算是C 语言的结构，也需要这两个参数设置才能获取成员信息。

这样就能自动完成结构和类的成员了。

但是，对于系统的函数，还是没有跳转。如socket定义，inetaddr\_in这样的结构没有自动变量完成。  
最简单做法：

**$ ctags –R --fields=+iaS --extra=+q –R –f ~/.vim/systags /usr/include**

然后在.vimrc里设置

**set tags+=~/.vim/systags**

这样虽然基本能跳转到系统函数定义，一个问题是某些系统函数并没有加入到systags里。  
如/usr/incluce/socket.h的socket系列函数,memset等很多关键函数都没有到tag里：extern int listen (int \_\_fd, int \_\_n) \_\_THROW;

这是因为 \_\_THROW的宏定义让ctags不再认为该系列函数是函数。  
同理，如memcpy系列函数：  
如/usr/include/string.h的

extern int strcmp(\_\_const char \*\_\_s1, \_\_const char \*\_\_s2) \_\_THROW \_\_attribute\_pure\_\_ \_\_nonnull ((1,2));

还有attribute\_pure ，nonull等属性，都需要忽略。如果需要#if 0里面的定义，可以–if0=yes来忽略 #if 0这样的定义。这样.vim/systags里面是全的，但内容过多。一个函数定义的跳转，会有几十个候选。这时我们可以简化一下，将-R去掉，自己指定目录

**$ ctags –I \_\_THROW –I \_\_attribute\_pure\_\_ -I \_\_nonnull –I \_\_attribute\_\_ --file-scope=yes --langmap=c:+.h --languages=c,c++ --links=yes –c-kinds=+p --c++-kinds=+p --fields=+iaS --extra=+q -R –f ~/.vim/systags /usr/include/\* /usr/include/sys/\* /usr/include/bits/\***

还可以包含一些自己编程需要的路径。注意后面加\*号。  
这样生成的系统tags就少多了。不会有太多不相干的定义。

使用**:tag vimmain**之后你就会看到vim打开项目代码src/main.c文件，并把光标定位到第167行vimmain上。

tag命令和相关知识，可以**:help tagsrch.txt**

### Vim文件查找:lookupfile插件

本节课所用命令的帮助入口：**help lookupfile**

下面我们这里来实现如何让vim像在source Insight中查找文件一样，只要输入部分的文件名，就可以找到符合条件的文件。这个插件是：**lookupfile！**

它要求vim版本必须在7.0以上使用。安装好该插件后，在vim中执行**:helptags ~/.vim/doc**命令，生成help文件索引，然后就可以使用**:help lookupfile**帮助了。

lookupfile还需要最新的**genutils**的支持，

我们本课的lookupfile插件，其实它不仅能用来查找文件，还可以在打开的缓冲区中查找，按目录查找，等等。下面就是各种查找的分类使用。

项目文件查找

lookupfile在查找文件时，需要使用tag文件。它可以使用ctags生成的tag文件，但查找效率低。因此我们专门为它生成一个包含项目中所有文件名的tag文件。

下面是范例：生成一个文件名tag文件

#!/bin/sh

echo ‘!\_TAG\_FILE\_SORTED’> filenametags

**#!/bin/sh**

**#generate tag file for lookupfile plugin**

**echo –e “!\_TAG\_FILE\_SORTED\t2\t/2=foldcase/”> filenametags**

**find ./ -not -regex ‘.\*\.\(png\|gif\)’–type f –printf “%f\t%p\t1\n” | sort –f >> filenametags**

注释：

\*\*echo命令用来生成filenametags文件中的!\_TAG\_FILE\_SORTED行，表明此tag文件是经过排序的。

\*\*find命令用来查找所有类型为普通的文件，文件后缀名不是.png和.gif的文件，找到的文件按照“文件名\t文件路径\t1”的格式输出来。

\*\*sort命令则把find命令的输出重排序，然后写入filenametags文件中。

下面我们要让lookupfile去找到这个filenametags文件：

**let g:LookupFile\_TagExpr=‘“./filenametags”’**

现在我们可以启动lookupfile插件了，按**F5**或输入**:LookupFile**即在当前窗口上方打开一个lookupfile小窗口，开始输入文件名（至少4个字符），随着你的输入，符合条件的文件就列在下拉列表中了。文件名可以使用vim的正则表达式，这极大方便了文件的查找。你可以用**Ctrl-n**和**Ctrl-p**来在下拉列表中选择你所需的文件。选中文件后，按回车，就可以在之前的窗口中打开此文件。

缓冲区查找

在开发过程中，我经常会同时打开数十甚至上百个文件。即使是使用BufExplorer插件，想在这么多buffer中切换到自己所要的文件，也不是件容易的事。

Lookupfile插件提供了一个按缓冲区名字查找缓冲区的方式，只要输入缓冲区的名字（可以是正则表达式），它就可以把匹配的缓冲区列在下拉列表中，同时还会列出该缓冲区内文件的路径，当你的buffer中有多个同名文件时，这可以帮你迅速找到你想要的文件。

使用”**:LUBufs**“命令开始在缓冲区中查找，输入缓冲区的名字，在你输入的过程中，符合条件的缓冲区就显示在下拉列表中了，选中所需缓冲区后，按回车，就会切换你所选的缓冲区。

浏览目录

Lookupfile插件还提供了目录浏览的功能，使用”**:LUWalk**“打开lookupfile窗口后，就可以输入目录，lookupfile会在下拉列表中列出这个目录中的所有子目录及文件供选择，如果选择了目录，就会显示这个目录下的子目录和文件；如果选择了文件，就在vim中打开这个文件。

Lookupfile插件还提供了**LUPath**和**LUArgs**两个功能，这两个功能我用的不多，就不在这里介绍了。感兴趣的朋友读一下lookupfile的手册。

lookupfile参考配置

Lookupfile插件提供了一些配置选项，通过调整这些配置选项，使它更符合你的工作习惯。下面是我的vimrc中关于lookupfile的设置，供参考：

**""""""""""""""""""""""""""""""**

**" lookupfile setting**

**""""""""""""""""""""""""""""""**

**let g:LookupFile\_MinPatLength = 2 "最少输入2个字符才开始查找**

**let g:LookupFile\_PreserveLastPattern = 0 "不保存上次查找的字符串**

**let g:LookupFile\_PreservePatternHistory = 1 "保存查找历史**

**let g:LookupFile\_AlwaysAcceptFirst = 1 "回车打开第一个匹配项目**

**let g:LookupFile\_AllowNewFiles = 0 "不允许创建不存在的文件**

**if filereadable("./filenametags") "设置tag文件的名字**

**let g:LookupFile\_TagExpr = '"./filenametags"'**

**endif**

**"映射LookupFile为,lk**

**nmap <silent><leader>lk :LUTags<cr>**

**"映射LUBufs为,ll**

**nmap <silent><leader>ll :LUBufs<cr>**

**"映射LUWalk为,lw**

**nmap <silent><leader>lw :LUWalk<cr>**

有了上面的定义，当我输入”**,lk**“时，就会在tag文件中查找指定的文件名；当输入”**,ll**“时，就会在当前已打开的buffer中查找指定名字的buffer；当输入”**,lw**“时，就会在指定目录结构中查找。

另外，我还在项目相关的配置文件vim70sx.vim中加入了lookupfile所使用的tag文件的信息：

**" lookup file tag file**

**let g:LookupFile\_TagExpr = '"filenametags"'**

在用lookupfile插件查找文件时，是区分文件名的大小写的，如果想进行忽略大小写的匹配，可以使用vim忽略大小写的正则表达式，即在文件名的前面加上”\c”字符。举个例子，当你输入”**\cab.c**”时，你可能会得到”ab.c”、”Ab.c”、”AB.c”…

通常情况下我都进行忽略大小写的查找，每次都输入”\c”很麻烦。没关系，lookupfile插件提供了扩展功能，把下面这段代码加入你的vimrc中，就可以每次在查找文件时都忽略大小写查找了：

**" lookup file with ignore case**

**function! LookupFile\_IgnoreCaseFunc(pattern)**

**let \_tags = &tags**

**try**

**let &tags = eval(g:LookupFile\_TagExpr)**

**let newpattern = '\c' . a:pattern**

**let tags = taglist(newpattern)**

**catch**

**echohl ErrorMsg | echo "Exception: " . v:exception | echohl NONE**

**return ""**

**finally**

**let &tags = \_tags**

**endtry**

**" Show the matches for what is typed so far.**

**let files = map(tags, 'v:val["filename"]')**

**return files**

**endfunction**

**let g:LookupFile\_LookupFunc = 'LookupFile\_IgnoreCaseFunc'**

有时在LUBufs时也需要忽略缓冲区名字的大小写，我是通过直接修改lookupfile插件的方法，在LUBufs查找的字符串前都加上”\c”，使之忽略大小写。如果你不想这样，可以每次在缓冲区名字前手动加上”\c”。

本文关于Lookupfile插件就介绍这么多，请阅读手册获取更多的信息。

### 工程基础部件:taglist插件

在taglist窗口中，可以使用下面的快捷键：

<CR> 跳到光标下tag所定义的位置，用鼠标双击此tag功能也一样

o 在一个新打开的窗口中显示光标下tag

<Space> 显示光标下tag的原型定义

u 更新taglist窗口中的tag

s 更改排序方式，在按名字排序和按出现顺序排序间切换

x taglist窗口放大和缩小，方便查看较长的tag

+ 打开一个折叠，同zo

- 将tag折叠起来，同zc

\* 打开所有的折叠，同zR

= 将所有tag折叠起来，同zM

[[ 跳到前一个文件

]] 跳到后一个文件

q 关闭taglist窗口

<F1> 显示帮助

### 目录树管理：NEOTREE

？ 打开帮助

:vsplit +2 调整NEORTEE列宽

o 打开该目录

O

x 关闭父目录

X

P go to root

p go to parent

K go to first child

J go to last child

<tab> 打开该文件内容（千万别用o打开，否则窗口布局会混乱掉）

g<tab> 预览该文件内容

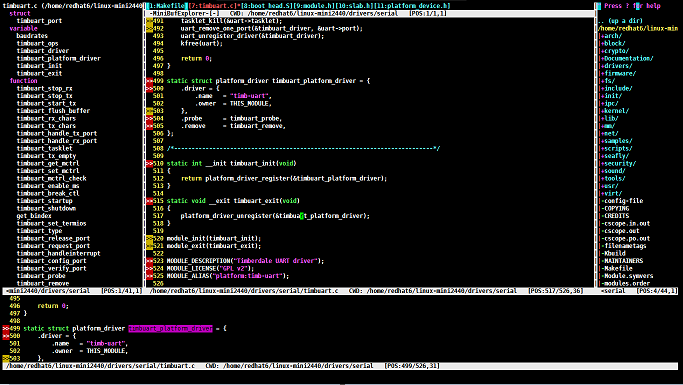
! 执行命令或文件

### vim也能像sourceinsight那样

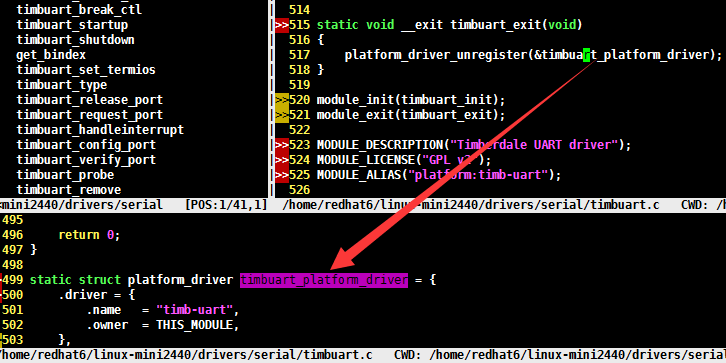
vimtool在安装时候自动集成了类似sourceinsight功能，

#### vimtool-sourceinsight使用

vimtool-sourceinsight整体界面效果截图如下：



放大界面图局部如下：



打开sourceinsight集成窗口

窗口大小调整

由于窗口大小会被我们各个操作动作所改变或变形，所以我们常用的一个vim命令是resize命令，用法是:resize +5把光标当前所在区高度增加5行。

跳转到该标签定义处

把光标切换到下方的定义处或实现处，并通过按回车来跳转到该位置，如果想往回跳转就按空格。

## vimtool开发手册

### 开发相关目录简介

\*\*config目录

里面就是vimrc配置文件和工程配置文件，自动安装时默认把vimrc文件配置进你的vimrc配置文件里，里面其他的vimrc文件作为开发者的参考样本，并可以从中提炼对他们自己有用的配置。

工程配置文件是在开发者使用工程项目的时候需要用到的文件，其实就是一个自动帮你生成工程依赖的数据库文件和tags文件。

\*\*doc目录

面向使用者：里面仅包含vimtool使用手册。

面向开发者：里面包含了vimtool开发手册和可修改的vimtool使用手册。

\*\*plugin目录

里面分为源码插件和脚本插件，脚本插件基本都是安装在你的~/.vim/目录下的；源码插件就是需要源码安装的插件，比如ctags,cscope等。

\*\*vim目录

该目录里主要包含vim主编辑器和vim相关的辅助编辑器等，该目录对于开发者来说就相当于用于编辑器的更新目录，目前是以8.0的版本为基础，如果以后有更高更强大的版本，那么可以通过这个目录来更新它。

### 工程脚本文件:object.sh

#LookupFile: dependent file

**echo** -e "!\_TAG\_FILE\_SORTED\t2\t/2=foldcase" **>./**filenametags

**find** **./** **-**type f **-**printf "%f\t%p\t1\n" **|** **sort** -f **>>./**filenametags

#ctags: dependent file

TAGARGS**=$\***

**ctags** **\**

**-**I \_\_THROW **\**

**-**I \_\_attribute\_pure\_\_ **\**

**-**I \_\_nonnull **\**

**-**I \_\_attribute\_\_ **\**

**--**file-scope**=**yes **\**

**--**langmap**=**c**:+.**h **\**

**--**languages**=**C**,**C++**,**Asm**,**HTML**,**Java **\**

**--**links**=**yes **\**

**--**c-kinds**=+**p **\**

**--**c++-kinds**=+**p **\**

**--**fields**=+**iaS **\**

**--**extra**=+**q **-**V -f **~/.**vim**/**systags **\**

**/**usr**/**include**/\*** **\**

**/**usr**/**include**/**alsa**/\*** **\**

**/**usr**/**include**/**arpa**/\*** **\**

**/**usr**/**include**/**asm**/\*** **\**

**/**usr**/**include**/**asm-generic**/\*** **\**

**/**usr**/**include**/**bits**/\*** **\**

**/**usr**/**include**/**linux**/\*** **\**

**/**usr**/**include**/**sys**/\*** **\**

**/**usr**/**include**/**net**/\*** **\**

**/**usr**/**include**/**netinet**/\*** **\**

${TAGARGS}

#build tags object file

#ctags --list-languages (list languages of ctags supporter)

**ctags** **-**RV **\**

**--**file-scope**=**yes **\**

**--**langmap**=**c**:+.**h **\**

**--**languages**=**Asm**,**Awk**,**C**,**C++**,**C#**,**HTML**,**Java**,**JavaScript**,**Make**,**PHP**,**Sh**,**python **\**

**--**links**=**yes **\**

**--**c-kinds**=+**p **\**

**--**c++-kinds**=+**p **\**

**--**fields**=+**iaS **\**

**--**extra**=+**q **.**

# --etags-include=~/.vim/systags \

#ctags -R --sort=1 --c++-kinds=+p --fields=+iaS --extra=+q --language-force=C++ -f cpp cpp\_src

#cscope: dependent file(s)

#cscope -Rbqv

cscope **-**Rbqv

### Vim配置文件:.vimrc

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

"vimtool:vimrc: vimtool vimrc configuration file

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

"tabel: configuration of tabel page

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

"Platform

function! MySys()

if has("win32")

return "windows"

else

return "linux"

endif

endfunction

function! SwitchToBuf(filename)

"let fullfn **=** substitute**(**a**:**filename**,** "^\\~/"**,** **$HOME** **.** "/"**,** ""**)**

"find in current tab

let bufwinnr = bufwinnr(a:filename)

if bufwinnr != -1

exec bufwinnr . "wincmd w"

return

else

"Find in each tab

tabfirst

**let** tab **=** 1

**while** tab **<=** tabpagenr**(**"$"**)**

**let** bufwinnr **=** bufwinnr**(**a**:**filename**)**

**if** bufwinnr **!=** **-**1

**exec** "normal " **.** tab **.** "gt"

**exec** bufwinnr **.** "wincmd w"

**return**

endif

tabnext

**let** tab **=** tab **+** 1

endwhile

"Not exist, new tab

exec "tabnew " . a:filename

endif

endfunction

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

"vimrc: configure itself

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

"set mapleader

"let mapleader **=** “**,**” default was "\" as value of mapleader

let mapleader = "**,**"

let g:mapleader = "**,**"

"fast reloading of the **.**vimrc

map **<**silent**>** **<**leader**>**ss **:**source **~/.**vimrc**<**cr**>**

"fast editing of .vimrc

map <silent> <leader>ee :e ~/.vimrc<cr>

"when **.**vimrc is edited**,** reload it.

autocmd**!** bufwritepost **.**vimrc source **~/.**vimrc

"Fast edit vimrc

if MySys() == 'linux'

"Fast reloading of **.**vimrc

map **<**silent**>** **<**leader**>**ss **:**source **~/.**vimrc**<**cr**>**

"Fast editing of .vimrc

map <silent> <leader>ee :call SwitchToBuf("**~/.**vimrc")<cr>

"When **.**vimrc is edited**,** reload it

autocmd**!** bufwritepost **.**vimrc source **~/.**vimrc

elseif MySys**()** **==** 'windows'

"Set helplang

set helplang=cn

"Fast reloading of the \_vimrc

map **<**silent**>** **<**leader**>**ss **:**source **~/**\_vimrc**<**cr**>**

"Fast editing of \_vimrc

map <silent> <leader>ee :call SwitchToBuf("**~/**\_vimrc")<cr>

"When \_vimrc is edited**,** reload it

autocmd**!** bufwritepost \_vimrc source **~/**\_vimrc

endif

"For windows version

if MySys() == 'windows'

source $VIMRUNTIME/mswin.vim

behave mswin

endif

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

"vim**:** vim's basic configuration for object-management

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

set history=700

syntax enable "syntax switch enable

syntax on "syntax switch on

set nocp "vi compatible mode

" Enable filetype plugins

filetype plugin on "Allow file type check

filetype plugin indent on "Allow indent file type check

filetype indent on

set number "Show line number

"set nonumber "Don't show line number

"set nohlsearch "No high light search key word**(**s**)**

colorscheme evening "Set colorscheme (Default evening)

" Set utf8 as standard encoding and en\_US as the standard language

**set** encoding**=**utf8

" Use Unix as the standard file type

set ffs=unix,dos,mac

set completeopt=longest,menu ""Close preview window

" Set to auto read when a file is changed from the outside

**set** autoread

" Ignore compiled files

set wildignore=\*.o,\*~,\*.pyc

"Always show current position

**set** ruler

" Ignore case when searching

set ignorecase

" When searching try to be smart about cases

**set** smartcase

" Highlight search results

set hlsearch

" Makes search act like search in modern browsers

**set** incsearch

" For regular expressions turn magic on

set magic

" Turn backup off**,** since most stuff is in SVN**,** git et.c anyway...

**set** nobackup

**set** nowb

**set** noswapfile

" Use spaces instead of tabs

set expandtab

" Be smart when using tabs **;)**

**set** smarttab

" 1 tab == 4 spaces

set shiftwidth=4

set tabstop=4

set ai "Auto indent

**set** si "Smart indent

set wrap "Wrap lines

**set** mouse**=**a "就可以在vim下使用鼠标了。

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

"vim-map**:** vim's map configuration for object-management

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

" Visual mode pressing \* or # searches for the current selection

" Super useful! From an idea by Michael Naumann

vnoremap <silent> \* :call VisualSelection('f')<CR>

vnoremap <silent> # :call VisualSelection('b')<CR>

" Map <Space> to / (search) and Ctrl-<Space> to ? (backwards search)

map <space> /

map <c-space> ?

" Disable highlight when <leader><cr> is pressed

map <silent> <leader><cr> :noh<cr>

" Smart way to move between windows

map <C-j> <C-W>j

map <C-k> <C-W>k

map <C-h> <C-W>h

map <C-l> <C-W>l

" Close the current buffer

map <leader>bd :Bclose<cr>

" Close all the buffers

map <leader>ba :1,1000 bd!<cr>

" Return to last edit position when opening files (You want this!)

autocmd BufReadPost \*

\ if line("'\"") > 0 && line("'\"") <= line("$") |

\ exe "normal! g`\"" |

\ endif

" Remember info about open buffers on close

"set viminfo^=%

" Format the status line

"set statusline=\ %{HasPaste()}%F%m%r%h\ %w\ \ CWD:\ %r%{getcwd()}%h\ \ \ Line:\ %l

set statusline=\ %{HasPaste()}%F%m%r%h\ %w\ \ CWD:\ %r%{getcwd()}%h\ \ \ [POS:%l/%L,%c]

set laststatus=2

set ruler

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

"LookupFile: => Spell checking

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

" Pressing ,ss will toggle and untoggle spell checking

map <leader>ss :setlocal spell!<cr>

" Shortcuts using <leader>

map <leader>sn ]s

map <leader>sp [s

map <leader>sa zg

map <leader>s? z=

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

"LookupFile: => Helper functions

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

function! CmdLine(str)

exe "menu Foo.Bar :" . a:str

emenu Foo.Bar

unmenu Foo

endfunction

function! VisualSelection(direction) range

let l:saved\_reg = @"

execute "normal! vgvy"

let l:pattern = escape(@", '\\**/.\*$^~[]**')

let l:pattern = substitute(l:pattern, "\n$", "", "")

if a:direction == 'b'

execute "normal ?" . l:pattern . "^M"

elseif a:direction == 'gv'

call CmdLine("vimgrep " . '**/**'. l:pattern . '**/**' . ' **\*\*/\*.**')

elseif a:direction == 'replace'

call CmdLine("%s" . '**/**'. l:pattern . '**/**')

elseif a:direction == 'f'

execute "normal /" . l:pattern . "^M"

endif

let @/ = l:pattern

let @" = l:saved\_reg

endfunction

" Returns true if paste mode is enabled

function! HasPaste()

if &paste

return 'PASTE MODE '

en

return ''

endfunction

" Don't close window**,** when deleting a buffer

command**!** Bclose call **<**SID**>**BufcloseCloseIt**()**

function**!** **<**SID**>**BufcloseCloseIt**()**

**let** l**:**currentBufNum **=** bufnr**(**"%"**)**

**let** l**:**alternateBufNum **=** bufnr**(**"#"**)**

**if** buflisted**(**l**:**alternateBufNum**)**

buffer #

**else**

bnext

endif

**if** bufnr**(**"%"**)** **==** l**:**currentBufNum

new

endif

**if** buflisted**(**l**:**currentBufNum**)**

execute**(**"bdelete! "**.**l**:**currentBufNum**)**

endif

endfunction

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

"LookupFile: lookupfile plugin configuration

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

let g:LookupFile\_TagExpr = '"**./**filenametags"'

"let g**:**LookupFile\_TagExpr **=** string**(**'./filenametags'**)**

""""let g**:**myLookupFileTagExpr **=** './filenanmetags'

""""let g**:**LookupFile\_TagExpr **=** 'g:myLookupFileTagExpr'

"let g:LookupFile\_MinPatLength = 2 "最少输入2个字符才开始查找

**let** g**:**LookupFile\_MinPatLength **=** 1 "最少输入1个字符才开始查找

let g:LookupFile\_PreserveLastPattern = 0 "不保存上次查找的字符串

**let** g**:**LookupFile\_PreservePatternHistory **=** 1 "保存查找历史

let g:LookupFile\_AlwaysAcceptFirst = 1 "回车打开第一个匹配项目

**let** g**:**LookupFile\_AllowNewFiles **=** 0 "不允许创建不存在的文件

let g:LookupFile\_RecentFileListSize = 30

""""" Don't display binary files

let g:LookupFile\_FileFilter = '\.class**$\|**\.o**$\|**\.obj**$\|**\.exe**$\|**\.jar**$\|**\.zip**$\|**\.war**$\|**\.ear$'

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

"LookupFile: lookup file with ignore case

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

function! LookupFile\_IgnoreCaseFunc(pattern)

let \_tags = &tags

try

let &tags = eval(g:LookupFile\_TagExpr)

let newpattern = '\c' . a:pattern

let tags = taglist(newpattern)

catch

echohl ErrorMsg | echo "Exception: " . v:exception | echohl NONE

return ""

finally

let &tags = \_tags

endtry

"show the matches for what is typed so far.

let files = map(tags, 'v**:**val**[**"filename"**]**')

return files

endfunction

let g:LookupFile\_LookupFunc = 'LookupFile\_IgnoreCaseFunc'

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

"winmanager: configuration of taglist, Explorebuf, minibuf etc.

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

let winManagerWindowLayout = 'FileExplorer**|**TagList'

let Tlist\_Auto\_Highlight\_Tag = 1

let Tlist\_Exit\_OnlyWindow = 1

let Tlist\_Show\_Menu = 1

let Tlist\_Show\_One\_File = 1

let g:bufExplorerMaxHeight=30

let g:miniBufExplorerMoreThanOne=0

"Press <F12> open the winmanager diagram framework

"map <F12> :WMToggle<CR>

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

"vim-header: You can add these lines into your `.vimrc`

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

"Easily Adds Brief Author Info and License Headers

"Usage

"=====

"This is a general usage example.

"You can add these lines into your `.vimrc`

" let g:header\_field\_author = 'Your Name'

" let g:header\_field\_author\_email = 'your**@**mail'

" map <F4> :AddHeader<CR>

"Pressing `F4` in normal mode will add a brief author information at the top of your buffer.

"Examples

"========

"For example, when you open a file named `start.sh` and press `F4` after above settings, plugin will add these lines at the top of your buffer

" #!/bin/bash

" # start.sh

" # Author: Your Name <your@mail>

" # Date: 13.03.2016

"or for a file named `index.php`

" <?php

" /\*

" \* index.php

" \* Author: Your Name <your@mail>

" \* Date: 13.03.2016

" \*/

"Commands

"========

"Adding Brief Headers

"- `:AddHeader` Adds brief author information

"- `:AddMinHeader` Adds minified version of author information

"Adding Lincenses

"- `:AddMITLicense` Adds MIT License with author info

"- `:AddApacheLicense` Adds Apache License with author info

"- `:AddGNULicense` Adds GNU License with author info

"Settings

"========

"These settings are for your `.vimrc`

" let g:header\_field\_filename = 0

"It disables to add filename line in header. Default is 1.

" let g:header\_field\_author = 'Your Name'

"It adds your name as author. Default is ''. Empty string means to disable adding it.

" let g:header\_field\_author\_email = 'your**@**mail'

"It adds your email after author name with surrounding `<``>` chars. If you don't define your author name**,** defined email also won't be shown. Default is ''. Empty string means to disable adding it.

" let g:header\_field\_timestamp = 0

"It disables to add timestamp line of generating header date in header. Default is 1.

" let g:header\_field\_timestamp\_format = '**%**d.**%**m.**%**Y'

"It sets timestamp format for your locale. Default is '**%**d.**%**m.**%**Y'.

"Support

"=======

"Supported filetypes are;

"- c

"- cpp

"- css

"- java

"- javascript

"- php

"- perl

"- python

"- sh

"- vim

"And licenses are;

"- MIT

"- Apache

"- GNU

"If you want more filetypes or licenses,

"you can open issues or provide any improvements by pull requests on

"[alpertuna/vim-header](https://github.com/alpertuna/vim-header).

"Also you can correct my English on README file or at comments in source code.

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

"vim-header: configuration

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

let g:header\_field\_author = 'seafly'

let g:header\_field\_author\_email = 'seafly0616**@**qq.com'

let g:header\_field\_filename = 1

let g:header\_field\_timestamp = 1

let g:header\_field\_timestamp\_format = '**%**Y.**%**m.**%**d'

map <F2> :AddGNULicense<CR>

"Adding Brief Headers(You can modify it)

"map <F2> :AddHeader<CR> #Adds brief author information

"map <F2> :AddMinHeader<CR> #Adds minified version of author information

"Adding Lincenses

"map <F2> :AddMITLicense<CR> # Adds MIT License with author info

"map <F2> :AddApacheLicense<CR> # Adds Apache License with author info

"map <F2> :AddGNULicense<CR> # Adds GNU License with author info

""""""""""""""""""""""""""""""""""""""

"SrcExpl: vimrc setting

""""""""""""""""""""""""""""""""""""""

" // The switch of the Source Explorer "

" nmap <F8> :SrcExplToggle<CR>

" "

" // Set the height of Source Explorer window "

let g:SrcExpl\_winHeight = 8

" "

" // Set 100 ms for refreshing the Source Explorer "

let g:SrcExpl\_refreshTime = 100

" "

" // Set "Enter" key to jump into the exact definition context "

let g:SrcExpl\_jumpKey = "<ENTER>"

" "

" // Set "Space" key for back from the definition context "

let g:SrcExpl\_gobackKey = "<SPACE>"

" "

" // In order to avoid conflicts, the Source Explorer should know what plugins "

" // except itself are using buffers. And you need add their buffer names into "

" // below listaccording to the command ":buffers!" "

"let g:SrcExpl\_pluginList = [

" \ "\_\_Tag\_List\_\_",

" \ "\_NERD\_tree\_"

" \ ]

" "

" // Enable/Disable the local definition searching, and note that this is not "

" // guaranteed to work, the Source Explorer doesn't check the syntax for now. "

" **//** It only searches for a match with the keyword according to command 'gd' "

let g:SrcExpl\_searchLocalDef = 1

" "

" **//** Do not let the Source Explorer update the tags file when opening "

let g:SrcExpl\_isUpdateTags = 0

" "

" **//** Use 'Exuberant Ctags' with '--sort=foldcase -R .' or '-L cscope.files' to "

" **//** create**/**update a tags file "

let g:SrcExpl\_updateTagsCmd = "ctags **--**sort**=**foldcase -R **.**"

" "

" **//** Set "<F12>" key for updating the tags file artificially "

let g:SrcExpl\_updateTagsKey = "**<**F12**>**"

" "

" **//** Set "<F6>" key for displaying the previous definition in the jump list "

let g:SrcExpl\_prevDefKey = "**<**F6**>**"

" "

" **//** Set "<F7>" key for displaying the next definition in the jump list "

let g:SrcExpl\_nextDefKey = "**<**F7**>**"

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

"project.sh**:** configuration for vim

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""

map **<**F3**>** **:**set tags+**=~/.**vim**/**systags**,./**tags**,./**filenametags**<**cr**>**

map **<**F4**>** **:**cscope add **./**cscope.out **./<**cr**>**

"<F5> "run LookupFile

" Open and close all the three plugins on the same time

nmap <F8> :TrinityToggleAll<CR>

" Open and close the srcexpl.vim separately

nmap **<**F9**>** **:**TrinityToggleSourceExplorer**<**CR**>**

" Open and close the taglist.vim separately

nmap <F10> :TrinityToggleTagList<CR>

" Open and close the NERD\_tree.vim separately

nmap **<**F11**>** **:**TrinityToggleNERDTree**<**CR**>**

### Vimtool自动安装脚本:build\_all

#!/bin/bash

#Warning:package name mustn't be illegal characters or space blank

#--------------------------------------------------------------------------

#(configuration item)

#--------------------------------------------------------------------------

CUR**=**

MAKE**=**make

MKCLEAN**=**clean

MKINSTALL**=**install

MKARGS**=-**j4

SUDO**=**

CONFIG**=./**configure

CFG\_ARGS**=--**prefix**=/**usr

#=================================================================================

#vimtool configuration

#=================================================================================

VIMTOOL\_ROOT**=`pwd`** #vimtool顶层目录

VIMTOOL\_PLUGIN**=$VIMTOOL\_ROOT/**plugin

VIMTOOL\_PLG\_SCRIPT**=$VIMTOOL\_PLUGIN/**script

VIMTOOL\_PLG\_SOURCE**=$VIMTOOL\_PLUGIN/**source

VIMTOOL\_VIM**=$VIMTOOL\_ROOT/**vim

VIMTOOL\_DOC**=$VIMTOOL\_ROOT/**doc

VIMTOOL\_CONFIG**=$VIMTOOL\_ROOT/**config

VIMRC**=**vimrc #config/vimrc

OBJECT\_TOOL**=**object.sh #config/object.sh

VIM\_CFG\_DIR**=$HOME/.**vim

VIM\_CFG\_DIR\_PLUGIN**=$VIM\_CFG\_DIR/**plugin

VIM\_CFG\_DIR\_DOC**=$VIM\_CFG\_DIR/**doc

VIM\_CFG\_DIR\_AUTOLOAD**=$VIM\_CFG\_DIR/**autoload

OBJECT\_TOOL\_PATH**=/**usr**/**bin

TEMP\_FILE**=$HOME/**seafly\_temp

TAR**=**tar

UNZIP**=**unzip

DEL\_DIR**=**"rm -rf"

ARG\_DOT\_TAR**=-**xvf #xxx.tar

ARG\_DOT\_TAR\_GZ**=-**xzvf #xxx.tar.gz or xxx.tgz

ARG\_DOT\_TAR\_BZ2**=-**xjvf #xxx.tar.bz2

#=================================================================================

#vim editor

#=================================================================================

VIM**=**vim-8.0.tar.bz2 #vim source package name

TAR\_VIM**=$TAR** #decompress package tool

TAR\_VIM\_ARGS**=$ARG\_DOT\_TAR\_BZ2** #decompress command arguments

VIM\_DIR**=**vim80 #vim\_dir after decompression

#auto get python directory name

list**=`ls /usr/lib | grep "python"`**

**ls** **/**usr**/**lib **|** **grep** "python" **>** ${TEMP\_FILE}

python\_version**=`awk -F . '{print $1}' ${TEMP\_FILE}`** #python2

value**=**

ret**=**

**for** value **in** **$list**

**do**

ret**=$value**

**echo** "$value" **>** ${TEMP\_FILE}

value**=`awk -F . '{print $1}' ${TEMP\_FILE}`**

**if** **[** "$value" **==** "${python\_version}" **]** **;** #main support python2.x

**then**

**break**

**fi**

**done**

PY\_VERSION**=$ret**

VIM\_CONFIG**=(./**configure **\**

**--**prefix**=/**usr **\**

**--**enable-pythoninterp**=**yes **\**

**--**with-python-config-dir**=/**usr**/**lib**/**${PY\_VERSION}**/**config**)**

#echo "VIM\_CONFIG: ${VIM\_CONFIG[\*]}" #print all element of array

#======================================================================

####--with-features=huge \

####--with-features=big \

####--enable-cscope \

####--enable-multibyte \

BVI**=**bvi-1.4.0-src-11.31.tar.gz #vim source package name

TAR\_BVI**=$TAR** #decompress package tool

TAR\_BVI\_ARGS**=$ARG\_DOT\_TAR\_GZ** #decompress command arguments

BVI\_DIR**=**bvi-1.4.0 #vim\_dir after decompression

#auto get python directory name

BVI\_CONFIG**=(./**configure **\**

**--**prefix**=/**usr**)**

#==============================================================================

#Following are source packages (need to be compiled) plugins

#You should follow the format of following arguments if you want to modify it or add others

CTAGS**=**ctags-5.8.tar.gz

TAR\_CTAGS**=$TAR** #decompressing with tar

TAR\_CTAGS\_ARGS**=$ARG\_DOT\_TAR\_GZ**

CTAGS\_DIR**=**ctags-5.8

CSCOPE**=**cscope-15.8b.tar.gz

TAR\_CSCOPE**=$TAR**

TAR\_CSCOPE\_ARGS**=$ARG\_DOT\_TAR\_GZ**

CSCOPE\_DIR**=**cscope-15.8b

################################################################################

#Following configuration of YCM

####step01: install those libraries

####sudo apt-get install libncurses5-dev libgnome2-dev libgnomeui-dev

####libgtk2.0-dev libatk1.0-dev libbonoboui2-dev libcairo2-dev

####libx11-dev libxpm-dev libxt-dev python-dev ruby-dev mercurial

####step02: configure the vim editor

####make distclean

####./configure

####--with-features=huge

####--enable-pythoninterp

####--with-python-config-dir=/usr/lib/python2.7/config

####--enable-cscope

####--enable-multibyte

####--prefix=/usr

####sudo make VIMRUNTIMEDIR=/usr/share/vim/vim80

####sudo make install

####You should install 'python-dev' before install vim

####VIM\_CONFIG=(./configure \

#### --with-features=huge \

#### --enable-cscope \

#### --enable-multibyte \

#### --prefix=/usr \

#### --enable-pythoninterp \

#### --with-python-config-dir=/usr/lib/python2.6/config)

####echo "VIM\_CONFIG: ${VIM\_CONFIG[\*]}" #how to use this array

####step02: compile and install llvm-clang(Let YCM support C/C++ semanteme complete)

####mkdir ~/ycm\_tmp

####tar -xvf clang3.2-x86-linux-ubuntu-12.04.tar.gz

####step03: use the 'vundle' download YouCompleteMe

####vundle is a managerment of vim-plugin

####install vundle: unzip it in ~/.vim directory

####after installization, please add following message into ~/.vimrc

####Plugin 'Valloric/YouCompleteMe'

####After above, run ':BundleInstall' command in vim

####and if you download successfully, you can see 'YouCompleteMe' directory in your vundle.

####step04: compile YouCompleteMe

####install cmake，because YCM need it

####sudo apt-get install cmake

####make directory 'ycm\_build', and then cmake need it, and the name must be 'ycm\_build'

####cd ~

####mkdir ycm\_build

####cd ycm\_build

####cmake -G "Unix Makefiles"

####-DEXTERNAL\_LIBCLANG\_PATH=

####~/ycm\_tmp/clang+llvm-3.2-x86-linux-ubuntu-12.04/lib/libclang.so .

####~/.vim/bundle/YouCompleteMe/third\_party/ycmd/cpp

####make ycm\_support\_libs

####step05: configure YouCompleteMe

####YCM needs a file named '.ycm\_extra\_conf.py' with work-well

####copy it into your item's root directory or your home directory

####cp ~/.vim/bundle/YouCompleteMe/third\_party/ycmd/cpp/ycm/.ycm\_extra\_conf.py your\_dir

####and modify this file:

####add following message into 'flags':

####'-I' '/usr/include/c++/4.8 '

####'-I ' '/usr/include/i386-linux-gnu/c++/4.8'

####and then find out the following code block:

####try:

####final\_flags.remove('-stdlib=libc++')

####except ValueError:

####pass

####and comment it

####step06: add path into your vimrc

####add following message into your ~/.vimrc :

####let g:ycm\_global\_ycm\_extra\_conf='~/.ycm\_extra\_conf.py'

################################################################################

#--------------------------------------------------------------------------

#functions' implement

#--------------------------------------------------------------------------

**function** debug\_vimtool**()**

**{**

**echo** "function debug\_vimtool()>>>complete installation total list"

**read** temporary

**echo** "function install\_vimtool()>>>installation main function"

**echo** "function install\_vimtool()>>>complete installation"

**echo** "function install\_vim()>>>install vim"

**echo** "function only\_vim()>>>only install vim"

**echo** "function install\_plugin()>>>install plugins"

**echo** "function source\_plugin()>>>source code plugins"

**echo** "function source\_tar\_plugin()>>>\*.tar format source plugin packages"

**echo** "function source\_zip\_plugin()>>>\*.zip format source plugin packages"

**echo** "function source\_tar\_gz\_plugin()>>>\*.tar.gz format source plugin packages"

**echo** "function source\_tar\_bz2\_plugin()>>>\*.tar.bz2 format source plugin packages"

**echo** "function script\_plugin()>>> script plugins"

**echo** "function script\_vim\_plugin()>>>\*.vim format plugins"

**echo** "function script\_tar\_plugin()>>>\*.tar format plugins"

**echo** "function script\_zip\_plugin()>>>\*.zip format plugins"

**echo** "function script\_tar\_gz\_plugin():\*.tar.gz format plugins"

**echo** "function script\_tar\_bz2\_plugin()>>>\*.tar.bz2 format plugins"

**echo** "function install\_config()>>>install configuration files"

**echo** "function config\_object()>>>install object configuration file(s)"

**echo** "function config\_vimrc()>>>install vimrc configuration file(s)"

**echo** " "

**echo** " "

**echo** debugingdebugingdebugingdebugingdebuging **;read** debuging

**echo** debugingdebugingdebugingdebugingdebuging **;read** debuging

**echo** debugingdebugingdebugingdebugingdebuging **;read** debuging

**echo** debugingdebugingdebugingdebugingdebuging **;read** debuging

**echo** debugingdebugingdebugingdebugingdebuging **;read** debuging

**echo** debugingdebugingdebugingdebugingdebuging **;read** debuging

**echo** debugingdebugingdebugingdebugingdebuging **;read** debuging

**echo** debugingdebugingdebugingdebugingdebuging **;read** debuging

**echo** debugingdebugingdebugingdebugingdebuging **;read** debuging

**echo** debugingdebugingdebugingdebugingdebuging **;read** debuging

**echo** debugingdebugingdebugingdebugingdebuging **;read** debuging

**echo** debugingdebugingdebugingdebugingdebuging **;read** debuging

**exit** 0

**return** 0

**}**

**function** vimtool\_finish**()**

**{**

#=================================================================

#installation finished

#=================================================================

**echo** "CUR: ${CUR}" #/root/home/user1/vimtool

**echo** "Finish installation! Please read doc/xxx.pdf for reference"

**echo** "Note"

**echo** " Run vim first time, you need goto ~/.vim/doc, likes following commands:"

**echo** " # cd ~/.vim/doc"

**echo** " # vim"

**echo** " "

**echo** " and then run following 'vim-command':"

**echo** " :helptags ./ (It can load help files into vim)"

**echo** " (If you add other plugins help file(s), replay above stage)"

**echo** " "

**echo** "Press <Enter> look following message...(Press <Ctrl-c> to ignore)"

**read** temporary

**clear**

**echo** "Manual:(Readme!)"

**echo** ""

**echo** " step01 cd ~/your\_object\_dir"

**echo** " step02 run object.sh directly"

**echo** " (execute object.sh if you want to update object\_depend\_file)"

**echo** " step03 run vim"

**echo** " Press <F3> load tags file"

**echo** " Press <F4> load cscope file"

**echo** " "

**echo** " Press <F12> open the winmanager window(selectable)"

**echo** " "

**echo** " After above operations, we have finished initialazation of object"

**echo** " and then you can see vimtool/doc/xxx.pdf for details"

**echo** " "

**echo** " (Installation details: vimtool/build\_all)"

**echo** " (Plugin configuration: vimtool/config/vimrc)"

**echo** " (Object configuration: vimtool/config/object.sh)"

**echo** " E-mail: seafly0616@qq.com"

#=================================================================

**}**

**function** build\_all\_help**()**

**{**

**echo** "Simple installation information:"

**echo** " ./build\_all #Complete install vimtool(Recommend first use)"

**echo** " ./build\_all only\_vim #Only install vim"

**echo** " ./build\_all no\_vim #Only install plugins"

**echo** " ./build\_all script\_plugin #Only install script plugins"

**echo** " ./build\_all source\_plugin #Only install source plugins"

**echo** " ./build\_all update\_config #Only install configuration files"

**echo** "Helpful information of first-use"

**echo** " # cd ~/.vim/doc"

**echo** " # vim"

**echo** " "

**echo** " Then run following command to load help files"

**echo** " :helptags ./ "

**echo** " (If you add other new plugins, replay above stage)"

**echo** " "

**echo** "Press <Enter> to continue...(Press <Ctrl-c> to ignore)"

**read** temporary

**clear**

**echo** "Manual:(Readme!)"

**echo** ""

**echo** " step01 cd ~/your\_object\_dir"

**echo** " step02 run object.sh directly"

**echo** " (execute object.sh if you want to update object\_depend\_file)"

**echo** " step03 run vim"

**echo** " Press <F3> load tags file"

**echo** " Press <F4> load cscope file"

**echo** " "

**echo** " Press <F12> open the winmanager window(selectable)"

**echo** " "

**echo** " After above operations, we have finished initialazation of object"

**echo** " and then you can see vimtool/doc/xxx.pdf for details"

**echo** " "

**echo** " (Installation details: vimtool/build\_all)"

**echo** " (Plugin configuration: vimtool/config/vimrc)"

**echo** " (Object configuration: vimtool/config/object.sh)"

**echo** " E-mail: seafly0616@qq.com"

#=================================================================

**}**

**function** only\_vim**()**

**{**

**echo** "function only\_vim()>>>only install vim"

CUR**=$VIMTOOL\_VIM**

**cd** **$CUR** #vimtool/vim

**echo** "CUR: $CUR" #CUR: vimtool

**$TAR\_VIM** **$TAR\_VIM\_ARGS** **$VIM**

**cd** **$VIMTOOL\_VIM/$VIM\_DIR** #vimtool/vim/vim80

**$MAKE** **$MKCLEAN**

**$VIM\_CONFIG** #configuration for python

**$MAKE** **$MKARGS**

**$SUDO** **$MAKE** **$MKINSTALL**

**cd** **$VIMTOOL\_VIM**

**$DEL\_DIR** **$VIM\_DIR**

CUR**=$VIMTOOL\_VIM**

**cd** **$CUR** #vimtool/vim

**echo** "CUR: $CUR" #CUR: vimtool

**$TAR\_BVI** **$TAR\_BVI\_ARGS** **$BVI**

**cd** **$VIMTOOL\_VIM/$BVI\_DIR** #vimtool/vim/vim80

**$MAKE** **$MKCLEAN**

**$BVI\_CONFIG**

**$MAKE** **$MKARGS**

**$SUDO** **$MAKE** **$MKINSTALL**

**cd** **$VIMTOOL\_VIM**

**$DEL\_DIR** **$BVI\_DIR**

**echo** "only\_vim():successfully!"

**return** 0

**}**

**function** source\_tar\_gz\_plugin**()**

**{**

**echo** "function source\_tar\_gz\_plugin()>>>\*.tar.gz script plugins"

**[** **-d** **$VIM\_CFG\_DIR** **]**

**if** **[** **$?** **-**ne 0 **]** **;**

**then**

**mkdir** -p **$VIM\_CFG\_DIR**

**fi**

**cp** **-**v **$VIMTOOL\_PLG\_SOURCE/\*.**tar.gz **$VIM\_CFG\_DIR**

**cd** **$VIM\_CFG\_DIR**

**local** value**=**

**local** value\_dir**=**

**local** list**=`ls \*.tar.gz`**

**echo** "list: $list"

**for** value **in** **$list**

**do**

**if** **[** **-z** **$value** **]** **;**

**then**

**break**

**fi**

#检查value是否为普通 script plugins

**if** **[** **$value** **==** "netrw-93.tar.gz" **]** **;**

**then**

**echo** "$value isn't a source package"

**elif** **[** **$value** **==** "vimcdoc-1.5.0.tar.gz" **]** **;**

**then**

**echo** "$value isn't a source package"

**else**

**echo** ""

**fi**

#解压压缩包

**$TAR** **$ARG\_DOT\_TAR\_GZ** **$value**

#获取压缩包解压之后的目录:这里直接根据命名规则获取

#还有其它获取方式等用到再说

**echo** "$value" **>** **$TEMP\_FILE**

**echo** "value: $value"

value\_dir**=`awk -F . '{print $1}' $TEMP\_FILE`**

value\_dir**=**${value\_dir}**.**

value\_dir**=**${value\_dir}**`awk -F . '{print $2}' $TEMP\_FILE`**

**echo** "value\_dir: $value\_dir"

#测试解压之后的目录的有效性#失败，则通过另一种方式(\*)获取

**cd** **$VIM\_CFG\_DIR/$value\_dir**

**if** **[** **$?** **-**ne 0 **]** **;**

**then**

**echo** "$value" **>** **$TEMP\_FILE**

value\_dir**=`awk -F - '{print $1}' $TEMP\_FILE`**

value\_dir**=**${value\_dir}**\***

**cd** ${value\_dir}

**if** **[** **$?** **-**ne 0 **]** **;**

**then**

#手动指定解压之后的目录名:不用写绝对路径:vim80

**echo** "无法获取解压之后的目录名,"

**echo** "当前压缩包解压名: $value"

**echo** "请手动输入当前压缩包解压之后的目录名,"

**echo** -e "(请直接填目录名(相对路径)): \c"

**read** value\_dir

**fi**

**fi**

#进入解压之后的目录开始配置编译安装

**cd** **$VIM\_CFG\_DIR/$value\_dir**

**$CONFIG** **$CFG\_ARGS** **$CFG\_OTHER\_ARGS**

**$MAKE** **$MKARGS**

**$MAKE** **$MKINSTALL**

#安装完成一个就清除目录和临时文件并为下一个安装做准备

**cd** **$VIM\_CFG\_DIR**

**$DEL\_DIR** **$TEMP\_FILE**

**$DEL\_DIR** **$VIM\_CFG\_DIR/$value\_dir**

**done**

**$DEL\_DIR** **$VIM\_CFG\_DIR/\*.**tar.gz

**return** 0

**}**

**function** source\_tar\_bz2\_plugin**()**

**{**

**echo** "function source\_tar\_bz2\_plugin()>>>\*.tar.bz2 script plugins"

**return** 0

**}**

**function** source\_tar\_plugin**()**

**{**

**echo** "function source\_tar\_plugin()>>>\*.tar script plugins"

**return** 0

**}**

**function** source\_zip\_plugin**()**

**{**

**echo** "function source\_zip\_plugin()>>>\*.zip script plugins"

**return** 0

**}**

**function** source\_plugin**()**

**{**

**echo** "function source\_plugin()>>> script plugins"

**uname** **-**v **|** **grep** "Ubuntu"

**if** **[** **$?** **-**eq 0 **]** **;**

**then**

host www.baidu.com 1**>/**dev**/**null 2**>/**dev**/**null

**if** **[** **$?** **-**ne 0 **]** **;**

**then**

**echo** "Error: Network unavailable!"

**exit** 1

**fi**

**$SUDO** apt-get install ctags

**$SUDO** apt-get install cscope

**return** 0

**fi**

source\_tar\_plugin

source\_zip\_plugin

source\_tar\_gz\_plugin

source\_tar\_bz2\_plugin

**return** 0

**}**

**function** script\_vim\_plugin**()**

**{**

**echo** "function script\_vim\_plugin()>>>\*.vim script plugins"

**[** **-d** **$VIM\_CFG\_DIR\_PLUGIN** **]** **;**

**if** **[** **$?** **-**ne 0 **]** **;**

**then**

**mkdir** -p **$VIM\_CFG\_DIR\_PLUGIN**

**fi**

**cp** **-**v **$VIMTOOL\_PLG\_SCRIPT/\*.**vim **$VIM\_CFG\_DIR\_PLUGIN**

**if** **[** **$?** **-**ne 0 **]** **;**

**then**

**return** 2**;**

**fi**

**return** 0

**}**

**function** script\_zip\_plugin**()**

**{**

**echo** "function script\_zip\_plugin()>>>\*.zip script plugins"

**[** **-d** **$VIM\_CFG\_DIR** **]** **;**

**if** **[** **$?** **-**ne 0 **]** **;**

**then**

**mkdir** -p **$VIM\_CFG\_DIR**

**fi**

**cp** **-**v **$VIMTOOL\_PLG\_SCRIPT/\*.**zip **$VIM\_CFG\_DIR** 1**>/**dev**/**null 2**>/**dev**/**null

**if** **[** **$?** **-**ne 0 **]** **;**

**then**

**return** 2**;**

**fi**

**cd** **$VIM\_CFG\_DIR**

#unzip one by one

**local** value**=**

**local** list**=`ls \*.zip`**

**for** value **in** **$list**

**do**

**if** **[** **-z** **$value** **]** **;**

**then**

**break**

**fi**

**$UNZIP** **$value**

**done**

**$DEL\_DIR** **$VIM\_CFG\_DIR/\*.**zip

**return** 0

**}**

**function** script\_tar\_plugin**()**

**{**

**echo** "function script\_tar\_plugin()>>>\*.tar script plugins"

**[** **-d** **$VIM\_CFG\_DIR** **]** **;**

**if** **[** **$?** **-**ne 0 **]** **;**

**then**

**mkdir** -p **$VIM\_CFG\_DIR**

**fi**

**cp** **-**v **$VIMTOOL\_PLG\_SCRIPT/\*.**tar **$VIM\_CFG\_DIR** 1**>/**dev**/**null 2**>/**dev**/**null

**if** **[** **$?** **-**ne 0 **]** **;**

**then**

**return** 2**;**

**fi**

**cd** **$VIM\_CFG\_DIR**

#decompress \*.tar one by one

**local** value**=**

**local** list**=`ls \*.tar`**

**for** value **in** **$list**

**do**

**if** **[** **-z** **$value** **]** **;**

**then**

**break**

**fi**

**$TAR** **$ARG\_DOT\_TAR** **$value**

**done**

**$DEL\_DIR** **$VIM\_CFG\_DIR/\*.**tar

#Following special vim-header plugins

**cp** **-**rv **$VIMTOOL\_PLG\_SCRIPT/**vim-header**\*** **$VIM\_CFG\_DIR**

**cd** **$VIM\_CFG\_DIR**

**$TAR** **$ARG\_DOT\_TAR** vim-header**\*.**tar

**cp** **-**rv **$VIM\_CFG\_DIR/**vim-header**/**autoload**/\*** **$VIM\_CFG\_DIR\_AUTOLOAD/**

**cp** **-**rv **$VIM\_CFG\_DIR/**vim-header**/**plugin**/\*** **$VIM\_CFG\_DIR\_PLUGIN/**

**cp** **-**rv **$VIM\_CFG\_DIR/**vim-header**/**licensefiles **$VIM\_CFG\_DIR**

**return** 0

**}**

**function** script\_tar\_gz\_plugin**()**

**{**

**echo** "function script\_tar\_gz\_plugin():\*.tar.gz script plugins"

**[** **-d** **$VIM\_CFG\_DIR** **]** **;**

**if** **[** **$?** **-**ne 0 **]** **;**

**then**

**mkdir** -p **$VIM\_CFG\_DIR**

**fi**

**cp** **-**v **$VIMTOOL\_PLG\_SCRIPT/\*.**tar.gz **$VIM\_CFG\_DIR** 1**>/**dev**/**null 2**>/**dev**/**null

**if** **[** **$?** **-**ne 0 **]** **;**

**then**

**return** 2**;**

**fi**

**cd** **$VIM\_CFG\_DIR**

#decompress \*.tar.gz one by one

**local** value**=**

**local** list**=`ls \*.tar.gz`** 1**>/**dev**/**null 2**>/**dev**/**null

**if** **[** **$?** **-**ne 0 **]** **;**

**then**

**return** 2**;**

**fi**

**for** value **in** **$list**

**do**

**if** **[** **-z** **$value** **]** **;**

**then**

**break**

**fi**

**$TAR** **$ARG\_DOT\_TAR\_GZ** **$value**

**done**

**$DEL\_DIR** **$VIM\_CFG\_DIR/\*.**tar.gz

**return** 0

**}**

**function** script\_tar\_bz2\_plugin**()**

**{**

**echo** "function script\_tar\_bz2\_plugin()>>>\*.tar.bz2 script plugins"

**[** **-d** **$VIM\_CFG\_DIR** **]** **;**

**if** **[** **$?** **-**ne 0 **]** **;**

**then**

**mkdir** -p **$VIM\_CFG\_DIR**

**fi**

**cp** **-**v **$VIMTOOL\_PLG\_SCRIPT/\*.**tar.bz2 **$VIM\_CFG\_DIR** 1**>/**dev**/**null 2**>/**dev**/**null

**if** **[** **$?** **-**ne 0 **]** **;**

**then**

**return** 2**;**

**fi**

**cd** **$VIM\_CFG\_DIR**

#decompress \*.tar.bz2 one by one

**local** value**=**

**local** list**=`ls \*.tar.bz2`** 1**>/**dev**/**null 2**>/**dev**/**null

**for** value **in** **$list**

**do**

**if** **[** **-z** **$value** **]** **;**

**then**

**break**

**fi**

**$TAR** **$ARG\_DOT\_TAR\_BZ2** **$value**

**done**

**$DEL\_DIR** **$VIM\_CFG\_DIR/\*.**tar.bz2

**return** 0

**}**

**function** script\_plugin**()**

**{**

**echo** "function script\_plugin()>>> script plugins"

script\_tar\_bz2\_plugin

script\_tar\_gz\_plugin

script\_tar\_plugin

script\_zip\_plugin

script\_vim\_plugin

**return** 0

**}**

#Install plugins(both source and script)"

**function** install\_plugin**()**

**{**

**echo** "function install\_plugin()>>>Install plugins(both source and script)"

script\_plugin

source\_plugin

**return** 0

**}**

#Install vimtool/config/object.sh configuration files"

**function** config\_object**()**

**{**

**echo** "function config\_object()>>>Install object configuration files"

**$SUDO** cp **-**v **$VIMTOOL\_CONFIG/$OBJECT\_TOOL** **$OBJECT\_TOOL\_PATH**

**return** 0

**}**

#Install 'vimtool/config/vimrc' configuration file"

**function** config\_vimrc**()**

**{**

**echo** "function config\_vimrc()>>>Install vimrc configuration file"

**cat** **$VIMTOOL\_CONFIG/$VIMRC** **>** **$HOME/.**vimrc

**return** 0

**}**

#Install configuration files"

**function** install\_config**()**

**{**

**echo** "function install\_config()>>>Install configuration files"

config\_object

config\_vimrc

**}**

#Don't install vim"

**function** no\_vim**()**

**{**

**echo** "function no\_vim()>>>Don't install vim"

CUR**=`pwd`**

**echo** "CUR: $CUR"

install\_plugin

**return** 0

**}**

#Install vim"

**function** install\_vim**()**

**{**

**echo** "function install\_vim()>>>Install vim"

only\_vim

**return** 0

**}**

#Complete installation function

**function** complete\_install**()**

**{**

**echo** "function complete\_install()>>>Complete installation function"

install\_vim #step01 install vim editor

install\_plugin #step02 install plugin

install\_config #step03 install configuration file

**}**

**function** build\_vimconf\_dir**()**

**{**

**mkdir** -p **$VIM\_CFG\_DIR** 1**>/**dev**/**null 2**>/**dev**/**null

**mkdir** -p **$VIM\_CFG\_DIR\_PLUGIN** 1**>/**dev**/**null 2**>/**dev**/**null

**mkdir** -p **$VIM\_CFG\_DIR\_AUTOLOAD** 1**>/**dev**/**null 2**>/**dev**/**null

**mkdir** -p **$VIM\_CFG\_DIR\_DOC** 1**>/**dev**/**null 2**>/**dev**/**null

**return** 0

**}**

#The main function(entry)

**function** install\_vimtool**()**

**{**

**echo** "function install\_vimtool()>>>安装主函数"

**if** **[** **$UID** **-**ne 0 **]** **;**

**then**

**echo** "You had better to run build\_all as a root user"

**echo** "Press Enter to continue..."

SUDO**=**sudo

**fi**

INSTALL\_ARG**=$1**

**if** **[** **$#** **-**eq 0 **]** **;**

**then**

**echo** "Complete installation"

**$DEL\_DIR** **$VIM\_CFG\_DIR**

build\_vimconf\_dir

complete\_install

vimtool\_finish

**else**

**echo** "Incomplete installation"

**if** **[** **$INSTALL\_ARG** **==** "only\_vim" **]** **;**

**then**

**echo** "Only install vim editor"

only\_vim

vimtool\_finish

**elif** **[** **$INSTALL\_ARG** **==** "no\_vim" **]** **;**

**then**

**echo** "Only install plugins (both script and source)"

**$DEL\_DIR** **$VIM\_CFG\_DIR**

build\_vimconf\_dir

no\_vim

vimtool\_finish

**elif** **[** **$INSTALL\_ARG** **==** "script\_plugin" **]** **;**

**then**

**echo** "Only install script plugins"

script\_plugin

vimtool\_finish

**elif** **[** **$INSTALL\_ARG** **==** "source\_plugin" **]** **;**

**then**

**echo** "Only install source code plugins"

source\_plugin

vimtool\_finish

**elif** **[** **$INSTALL\_ARG** **==** "update\_config" **]** **;**

**then**

**echo** "Only update configuration files"

install\_config

vimtool\_finish

**elif** **[** **$INSTALL\_ARG** **==** "help" **]** **;**

**then**

**echo** "Display installation information:"

build\_all\_help

**else**

**echo** "ERROR: $INSTALL\_ARG unknown argument!!!"

**echo** "Press <Enter> to continue ..."

**read** temporary

build\_all\_help

**exit** 1

**fi**

**fi**

**return** 0

**}**

#--------------------------------------------------------------------------

#--------------------------------------------------------------------------

#开发调试部分

#--------------------------------------------------------------------------

#install\_vim:only\_vim (pass)

#install\_plugin:script\_plugin:script\_vim\_plugin (pass)

#install\_plugin:script\_plugin:script\_tar\_plugin (pass)

#install\_plugin:script\_plugin:script\_zip\_plugin (pass)

#install\_plugin:script\_plugin:script\_tar\_gz\_plugin (pass)

#install\_plugin:script\_plugin:script\_tar\_bz2\_plugin (pass)

#install\_plugin:source\_plugin:source\_tar\_gz\_plugin (pass)

#install\_config:config\_object() (pass)

#install\_config:config\_vimrc() (pass)

#VIM\_CONFIG=(./configure \

# --with-features=huge \

# --enable-cscope \

# --enable-multibyte \

# --prefix=/usr \

# --enable-pythoninterp \

# --with-python-config-dir=/usr/lib/python2.6/config)

#echo "VIM\_CONFIG: ${VIM\_CONFIG[\*]}"

#echo "configlen: ${#VIM\_CONFIG[\*]}"

#debug\_vimtool #类似断点:只能执行这个之上代码

#--------------------------------------------------------------------------

#--------------------------------------------------------------------------

#函数执行部分

#--------------------------------------------------------------------------

install\_vimtool **$1**

#--------------------------------------------------------------------------

# 第\*\*课-Linux网络配置

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

## 虚拟机网络设置

Vmware网络模式：桥接模式（Bridged）、网络地址转换模式（NAT）、主机模式（Host）

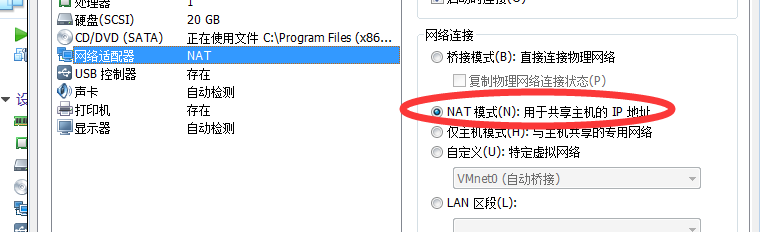
如果网络中能提供多个IP地址，则使用桥接模式，虚拟机<==>外部网络。

如果网络中只提供一个IP地址，则使用NAT模式，虚拟机==>外部网络。

如果主机处于单机状态，则使用主机模式，虚拟机<==>主机。

### vmware的NAT模式：先设置vmware网络方式NAT

选择vmware的网络连接方式为NAT模式：

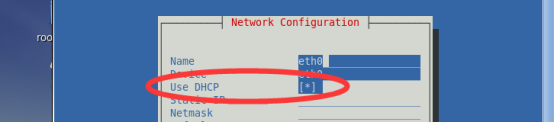


### vmware的NAT模式：启动vmwareNAT服务

“运行“ 》》services.msc



### vmware的NAT模式：虚拟机IP设置为DHCP获取



然后虚拟机重启网络就OK了。

## Linux网络设置

[root@redhat6 Desktop]#/etc/init.d/network restart#如果启动失败有如下原因

[root@redhat6 Desktop]#/etc/init.d/NetworkManager stop#关闭冲突服务网络管理员即可重启成功

[root@redhat6 Desktop]# ifconfig #运行之后少了eth0网卡

[root@redhat6 Desktop]# vim /etc/sysconfig/network-scripts/ifcfg-eth0

[root@redhat6 Desktop]#

DEVICE=eth0

BOOTPROTO=none

HWADDR=00:0c:29:fb:39:c1

NM\_CONTROLLED=yes

ONBOOT=yes

TYPE=Ethernet

UUID="b85d8ea2-2919-4426-809f-ef7e8de013e2"

IPADDR=192.168.1.10

NETMASK=255.255.255.0

GATEWAY=192.168.1.1

IPV6INIT=no

USERCTL=no

[root@redhat6 Desktop]#

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

# 第\*\*课-远程登录Linux系统

## 远程登录前的准备

登录前提1：从Windows能够ping通Linux

$ ping 192.168.1.15 #IP地址为Linux的IP地址

登录前提2：Linux关闭防火墙和selinux

# /etc/init.d/iptables stop

# setenforce 0

## 登录方式1：ssh方式来登录Linux终端

1. windows下使用putty.exe
2. Linux下使用ssh(openSSH.tar.gz)

ssh口令登录

$ ssh user@host #默认登录端口22

$ ssh -p 2222 user@host #可以使用-p参数指定登录端口

ssh公钥登录

这种方法要求用户必须提供自己的公钥。如果没有现成的，可以直接用ssh-keygen生成一个：$ ssh-keygen

这时再输入下面的命令，将公钥传送到远程主机host上面：

$ ssh-copy-id user@host

好了，从此你再登录，就不需要输入密码了。

如果还是不行，就打开远程主机的/etc/ssh/sshd\_config这个文件，检查下面几行前面”#”注释是否取掉。

RSAAuthentication yes

PubkeyAuthentication yes

AuthorizedKeysFile .ssh/authorized\_keys

然后，重启远程主机的ssh服务。

## 登录方式2：VNC来登录图形化Linux

**VNC概述**

   VNC (Virtual Network Computing)是[虚拟网络](http://baike.baidu.com/view/747782.htm)[计算机](http://baike.baidu.com/view/3314.htm)的缩写。VNC 是一款优秀的[远程控制](http://baike.baidu.com/view/51293.htm)工具软件，由著名的 [AT&T](http://baike.baidu.com/view/259956.htm) 的欧洲研究实验室开发的。VNC 是在基于 [UNIX](http://baike.baidu.com/view/8095.htm) 和 [Linux](http://baike.baidu.com/view/1634.htm)[操作系统](http://baike.baidu.com/view/880.htm)的免费的[开源软件](http://baike.baidu.com/view/444964.htm)，[远程控制](http://baike.baidu.com/view/51293.htm)能力强大，高效实用，其性能可以和 [Windows](http://baike.baidu.com/view/4821.htm) 或 [MAC](http://baike.baidu.com/view/32702.htm) 中的任何远程控制软件媲美。在 Linux 中，VNC 包括以下四个命令：vncserver，vncviewer，vncpasswd，和 vncconnect。大多数情况下只需要其中的两个命令：vncserver和 vncviewer。目前，原来的AT&T版本已经不再使用，因为更多有重大改善的分支版本已经出现， 像是RealVNC， VNC tight 和UltraVNC。 Real VNC是当前最活跃和强大的主流应用。

**VNC原理**

VNC系统由客户端，服务端和一个协议组成。VNC的服务端目的是分享其所运行机器的屏幕， 服务端被动的允许客户端控制它。 VNC客户端（或Viewer） 观察控制服务端，与服务端交互。 VNC 协议 Protocol (RFB)是一个简单的协议，传送服务端的原始图像到客户端（一个X,Y 位置上的正方形的点阵数据）， 客户端传送事件消息到服务端。

服务器发送小方块的帧缓存给客户端，在最简单的情况，VNC协议使用大量的带宽，因此各种各样的方法被发明出来减少通讯的开支，举例来说，有各种各样的编码方法来决定最有效率的方法来传送这些点阵方块）

协议允许客户端和服务端去协议哪种编码会被使用，最简单的编码，被大多数客户端和服务端所支持的是， 从左到右的像素扫描数据的原始编码， 当原始的满屏被发送后，只发送变化的方块区域。这种编码在幁间只有小部分屏幕变化的情况下工作的非常好（像是鼠标键在桌面移动的情况，或在光标处敲击文字），不过如果大量的像素同时变化带宽将会增加的非常高，像是拖动一个窗口或观看全屏录像。

VNC默认使用[TCP](http://zh.wikipedia.org/wiki/TCP)端口5900至5906，而JAVA的VNC客户端使用5800至5806。一个服务端可以在5500口用“监听模式”连接一个客户端，使用监听模式的一个好处是服务端不需要设置防火墙。

[UNIX](http://zh.wikipedia.org/wiki/UNIX)上的VNC称为xvnc，同时扮演两种角色，对[X窗口系统](http://zh.wikipedia.org/wiki/X_Window系統)的应用程序来说它是X server，对于VNC客户端来说它是VNC服务器程序。

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

**实验环境**

**VNC服务端：**

             操作系统：Red Hat Enterprise Linux Server release 5.7 (Tikanga)

**VNC客户端：**

             操作系统：Windows 7专业版  64位操作系统

## 安装VNC-server

光盘镜像中找到packages目录中找到tigervnc-server-xxx.rpm

此处涉及到iso的挂载：# mount -t iso9660 -o loop xxx.iso /media/isodir

安装该rpm包：# rpm –ivh vnc-server-xxx.rpm

[主目录](file:///C:\\Users\\Administrator\\Desktop\\专业体系2-精通嵌入式.docx" \l "_目录)

[root@redhat6 Desktop]# rpm -qa | grep "vnc"#检查VNC是否安装

gtk-vnc-0.3.10-3.el6.i686

tigervnc-1.0.90-0.17.20110314svn4359.el6.i686

tigervnc-server-1.0.90-0.17.20110314svn4359.el6.i686

[root@redhat6 Desktop]#

[root@redhat6 Desktop]# rpm -qal | grep "vnc" | grep "vncserver"

/etc/rc.d/init.d/vncserver

/etc/sysconfig/vncservers

/usr/bin/vncserver

/usr/bin/x0vncserver

/usr/share/man/man1/vncserver.1.gz

/usr/share/man/man1/x0vncserver.1.gz

[root@redhat6 Desktop]#

## 添加/修改VNC访问的密码

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

　　使用命令vncpasswd对不同用户的VNC的密码进行修改，一定要注意，如果配置了不同用户的VNC需要分别到各自用户中进行修改，例如在我的这个实验中，root用户和oracle用户需要分别修改，修改过程如下：  
　　[root@testdb ~]# vncpasswd  
　　Password:  
　　Verify:

## 设置或关闭防火墙

3、VNC服务使用的端口号与桌面号的关系

VNC服务使用的端口号与桌面号相关，VNC使用TCP端口从5900开始，对应关系如下：

桌面号为“1”  ---- 端口号为5901

桌面号为“2”  ---- 端口号为5902

桌面号为“3”  ---- 端口号为5903

基于上面的介绍，如果Linux开启了防火墙功能，就需要手工开启相应的端口，以开启桌面号为“1” “2”相应的端口为例，命令如下：

[root@redhat6 Desktop]# service iptables restart

iptables: Flushing firewall rules: [ OK ]

iptables: Setting chains to policy ACCEPT: filter [ OK ]

iptables: Unloading modules: [ OK ]

iptables: Applying firewall rules: [ OK ]

[root@redhat6 Desktop]# iptables -I INPUT -p tcp--dport 5901 -j ACCEPT

[root@redhat6 Desktop]# iptables -I INPUT -p tcp --dport 5902 -j ACCEPT

[root@redhat6 Desktop]#[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

## 修改VNC配置文件

[root@redhat6 Desktop]# vim /root/.vnc/xstartup

注释掉最后一行#twm &

末行添加：gnome-session &

先以桌面1为root用户桌面2为redhat6用户为例进行配置：  
[root@redhat6 Desktop]# vim /etc/sysconfig/vncservers

VNCSERVERS="1:root 2:redhat6"#VNCSERVERS="桌面号:使用的用户名桌面号:使用的用户名"

VNCSERVERARGS[1]="-geometry 1366x768 -nolisten tcp -localhost"

VNCSERVERARGS[2]="-geometry 1366x768 -nolisten tcp -localhost"

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

## 启动VNC服务端

[root@redhat6 Desktop]# vncserver &#然后后台运行这个VNC程序

[2] 4261

[root@redhat6 Desktop]# xauth: (stdin):1: bad display name "redhat6:2" in "add" command

[root@redhat6 Desktop]#

New 'redhat6:2 (root)' desktop is redhat6:2

Starting applications specified in /root/.vnc/xstartup

Log file is /root/.vnc/redhat6:2.log

[2]- Done vncserver

[root@redhat6 Desktop]# ps -aux | grep "vncserver"

Warning: bad syntax, perhaps a bogus '-'? See /usr/share/doc/procps-3.2.8/FAQ

root 3750 0.0 0.1 7048 2580 pts/0 T 16:44 0:00 perl /usr/bin/vncserver

root 4380 0.0 0.0 4340 760 pts/0 S+ 16:49 0:00 grep vncserver

[root@redhat6 Desktop]#

# vncserver :1 #打开1号桌面号

# vncserver –list #列出桌面号

# vncserver –kill :1 #关掉2号桌面号

# man vncserver | col –b > vncserver.man

# vncpasswd #直接运行此命令修改VNC服务器登录密码

启动多桌面服务程序

# vncserver :1

# vncserver :2

# vncserver :3

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

## 开机自启动vncserver服务

#chkconfig vncserver on

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

## 登录VNC

### windows作为客户端连入linux服务端

第一种方法是使用VNC Viewer软件登陆测试，启动VNC Viewer软件 ， Server输入“IP:1”，输入密码即可登录。

Windows中登录VNC：vnc-xxx.exe

Server:192.168.1.10:域名数字N

Server:192.168.1.10:1 <CR>

密码：就是Linux中为VNC设置的密码。

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

### Linux作为客户端连入linux服务端

[root@redhat6 Desktop]# ls

Makefile

tigervnc-1.0.90-0.17.20110314svn4359.el6.i686.rpm

tigervnc-server-1.0.90-0.17.20110314svn4359.el6.i686.rpm

[root@redhat6 Desktop]#

光盘镜像中找到packages目录中找到tigervnc-server-xxx.rpm

此处涉及到iso的挂载：# mount -t iso9660-o loop xxx.iso /media/isodir

安装该rpm包：# rpm –ivh vnc-xxx.rpm

[root@redhat6 Desktop]# vncviewer#启动LinuxVNC客户端或者

[root@redhat6 Desktop]# vncviewer 192.168.1.10:1

第二种方法是使用Web浏览器（如Firefox,IE,Safari）登陆，

地址栏输入http://IP:5801/，出现VNC viewer for Java（此工具是使用Java编写的VNC客户端程序）即可登录。

（注：VNC viewer for Java需要JRE支持）

**效果图**

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

**关于参数配置说明：**

1：-geometry 表示桌面分辨率，默认为1024x768，所以上面的1024x768也可以不写。

2：-nohttpd  表示不监听HTTP端口（58xx）。

3：-nolisten tcp 表示不监听TCP端口（60xx）

4：-localhost 只运行从本机访问。

5：AlwaysShared 默认只允许一个VNCVIEWER连接，此参数表示同一个显示端口允许多用户同时登录.

6：-depth  表示色深，参数有8,16,24,32.

7: SecurityTypes None 登录不需要密码认证VncAuth默认值,要密码认证。

5、配置VNC图形桌面环境为KDE

默认配置登陆到桌面后显示是非常简单的，是因为VNC服务默认使用的是twm图形桌面环境的，可以在VNC的配置文件xstartup中对其进行修改，取消配置文件中以下两项的#号。

# vim /root/.vnc/xstartup

unset SESSION\_MANAGER

exec /etc/X11/xinit/xinitrc

再重新启动vncserver服务后就可以登陆到KDE桌面环境  
[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

# 第\*\*课-Windows与Linux共享

## 共享方式1：winSCP的使用

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

## 共享方式2：samba服务方式来共享

## 安装samba服务器

光盘镜像中找到packages目录中找到samba-xxx.rpm

此处涉及到iso的挂载：# mount -t iso9660 -o loop xxx.iso /media/isodir

安装该rpm包：# rpm –ivh samba-xxx.rpm

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

[root@redhat6 Desktop]# rpm -qa | grep "samba"#检查samba是否安装

[root@redhat6 Desktop]#vim /etc/samba/smb.conf (:255) #配置samba

[redhat6]

comment = redhat6

browseable = yes

writable = yes

path = /home/redhat6

valid users = redhat6

; valid users = MYDOMAIN\%S//顺便注释掉下面的打印机和传真共享服务

[root@redhat6 Desktop]# useradd redhat6 (如果有该用户就省略此步)

[root@redhat6 Desktop]# passwd redhat6

[root@redhat6 Desktop]#smbpasswd –a redhat6 (为此用户添加samba密码)

[root@redhat6 Desktop]#smbpasswd –x redhat6 (删除此用户samba密码)

[root@redhat6 Desktop]#/etc/init.d/iptables stop

[root@redhat6 Desktop]#setenforce 0

[root@redhat6 Desktop]# /etc/init.d/smb restart

Windows登录Samba账户方式1：“运行”中输入：[\\192.168.1.10](file:///\\\\192.168.1.10)

Windows登录samba账户方式2：在IE浏览器地址栏中输入：\\192.168.1.10

常见问题：

访问权限不够解决方法：# chown redhat6:redhat6 /home/redhat6 –R

如果遇到需要使用防火墙，可以通过配置防火墙来使用samba服务，samba默认服务端口是445。

[root@redhat6 Desktop]# grep -Hn "445" /etc/\*

/etc/services:202:microsoft-ds 445/tcp

/etc/services:203:microsoft-ds 445/udp

[root@redhat6 Desktop]# /etc/init.d/iptables restart

iptables: Applying firewall rules: [ OK ]

[root@redhat6 Desktop]#

[root@redhat6 Desktop]# iptables -I INPUT -p tcp --dport 445 -j ACCEPT

[root@redhat6 Desktop]# iptables -I INPUT -p udp --dport 445 -j ACCEPT

[root@redhat6 Desktop]# setenforce 0

[root@redhat6 Desktop]# /etc/init.d/smb restart

Shutting down SMB services: [ OK ]

Starting SMB services: [ OK ]

[root@redhat6 Desktop]#

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

# 第\*\*课-TFTP与NFS服务器搭建

交叉开发概念

宿主机（PC机）：产生嵌入式软件的机器，目标机（开发板）：运行嵌入式软件的平台。

开发板程序下载：基于网络下载（TFTP、NFS）[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

## 配置TFTP服务器

光盘镜像中找到packages目录中找到tftp-server-xxx.rpm

此处涉及到iso的挂载：# mount -t iso9660 -o loop xxx.iso /media/isodir

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

[root@redhat6 Desktop]# rpm -qa | grep "tftp"#检查tftp是否安装

[root@redhat6 Desktop]#rpm –ivh xinetd-xxx.rpm#tftp服务器的依赖包

[root@redhat6 Desktop]#rpm –ivh tftp-server-xxx.rpm

[root@redhat6 Desktop]#vim /etc/xinetd.d/tftp#配置tftp

service tftp

{

socket\_type = dgram

protocol = udp

wait = yes

user = root

server = /usr/sbin/in.tftpd

server\_args = -s /tftpboot

disable = no

per\_source = 11

cps = 100 2

flags = IPv4

}

[root@redhat6 Desktop]#/etc/init.d/xinetd restart

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

## 配置NFS服务器

[root@redhat6 Desktop]# vim/etc/exports

/home/redhat6/rootfs 192.168.1.\*(rw,sync,no\_root\_squash)

[root@redhat6 Desktop]# /etc/init.d/iptables stop

[root@redhat6 Desktop]# setenforce 0

[root@redhat6 Desktop]# /etc/init.d/nfs restart

[root@redhat6 Desktop]#mount –t nfs 192.168.1.10:/home/redhat6/rootfs ./nfs

[root@redhat6 Desktop]# 上面挂载的NFS服务目录必须和NFS配置文件中的共享目录相同。

### Ubuntu下配置NFS

安装NFS相关服务软件：

sudo apt-get install nfs<TAB>

修改NFS配置文件：

/home/ubuntu \*(rw,sync,no\_root\_squash) >>> /etc/exports

重启NFS服务：

sudo

seafly@seafly-virtual-machine:~$ sudo service nfs-kernel-server restart

\* Stopping NFS kernel daemon [ OK ]

\* Unexporting directories for NFS kernel daemon... [ OK ]

\* Exporting directories for NFS kernel daemon... exportfs: /etc/exports [1]: Neither 'subtree\_check' or 'no\_subtree\_check' specified for export "\*:/home/ubuntu".

Assuming default behaviour ('no\_subtree\_check').

NOTE: this default has changed since nfs-utils version 1.0.x

[ OK ]

\* Starting NFS kernel daemon [ OK ]

seafly@seafly-virtual-machine:~$ ls /etc/init.d/nfs-kernel-server

测试NFS服务是否搭建成功：

seafly@seafly-virtual-machine:~$ sudo mount -t nfs localhost:/home/ubuntu /mnt/nfs

# 第\*\*课-Linux在线升级管理yum

前提：Linux处于联网状态

软件包仓库源：/etc/yum.repos.d/xxx.repo

安装免费仓库源：rpmforge-release-xxx.rpm

查看仓库中的软件包：yum list | grep “tftp”

安装仓库中的软件包：yum install 包名yum install atftp-server.i686

删除已经安装的软件包：yum remove 包名yum remove atftp-server.i686

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

其他国内yum源列表如下：

1. 企业贡献：

搜狐开源镜像站：http://mirrors.sohu.com/

网易开源镜像站：http://mirrors.163.com/

2. 大学教学：

北京理工大学：

http://mirror.bit.edu.cn (IPv4 only)

http://mirror.bit6.edu.cn (IPv6 only)

北京交通大学：

http://mirror.bjtu.edu.cn (IPv4 only)

http://mirror6.bjtu.edu.cn (IPv6 only)

http://debian.bjtu.edu.cn (IPv4+IPv6)

兰州大学：http://mirror.lzu.edu.cn/

厦门大学：http://mirrors.xmu.edu.cn/

清华大学：

http://mirrors.tuna.tsinghua.edu.cn/ (IPv4+IPv6)

http://mirrors.6.tuna.tsinghua.edu.cn/ (IPv6 only)

http://mirrors.4.tuna.tsinghua.edu.cn/ (IPv4 only)

天津大学：http://mirror.tju.edu.cn/

中国科学技术大学：

http://mirrors.ustc.edu.cn/ (IPv4+IPv6)

http://mirrors4.ustc.edu.cn/

http://mirrors6.ustc.edu.cn/

东北大学：

http://mirror.neu.edu.cn/ (IPv4 only)

http://mirror.neu6.edu.cn/ (IPv6 only)

电子科技大学：http://ubuntu.uestc.edu.cn/

## Ubuntu网络服务器配置

## CentOS6-Base-163.repo

# CentOS-Base.repo

#

# The mirror system uses the connecting IP address of the client and the

# update status of each mirror to pick mirrors that are updated to and

# geographically close to the client. You should use this for CentOS updates

# unless you are manually picking other mirrors.

#

# If the mirrorlist= does not work for you, as a fall back you can try the

# remarked out baseurl= line instead.

#

# cd /etc/yum.repos.d/

# yum clean all

# yum makecache

#

**[**base**]**

name**=**CentOS-**$releasever** **-** Base **-** 163**.**com

#baseurl=http://mirrors.163.com/centos/$releasever/os/$basearch/

baseurl**=**http**://**mirrors.163.com**/**centos**/**6**.**8**/**os**/$basearch/**

#mirrorlist=http://mirrorlist.centos.org/6.8?release=$releasever&arch=$basearch&repo=os

gpgcheck**=**1

gpgkey**=**http**://**mirror.centos.org**/**centos**/**RPM-GPG-KEY-CentOS-6

#released updates

**[**updates**]**

name**=**CentOS-**$releasever** **-** Updates **-** 163**.**com

baseurl**=**http**://**mirrors.163.com**/**centos**/**6**.**8**/**updates**/$basearch/**

#mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=updates

gpgcheck**=**1

gpgkey**=**http**://**mirror.centos.org**/**centos**/**RPM-GPG-KEY-CentOS-6

#additional packages that may be useful

**[**extras**]**

name**=**CentOS-**$releasever** **-** Extras **-** 163**.**com

baseurl**=**http**://**mirrors.163.com**/**centos**/**6**.**8**/**extras**/$basearch/**

#mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=extras

gpgcheck**=**1

gpgkey**=**http**://**mirror.centos.org**/**centos**/**RPM-GPG-KEY-CentOS-6

#additional packages that extend functionality of existing packages

**[**centosplus**]**

name**=**CentOS-**$releasever** **-** Plus **-** 163**.**com

baseurl**=**http**://**mirrors.163.com**/**centos**/**6**.**8**/**centosplus**/$basearch/**

#mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=centosplus

gpgcheck**=**1

enabled**=**0

gpgkey**=**http**://**mirror.centos.org**/**centos**/**RPM-GPG-KEY-CentOS-6

#contrib - packages by Centos Users

**[**contrib**]**

name**=**CentOS-**$releasever** **-** Contrib **-** 163**.**com

baseurl**=**http**://**mirrors.163.com**/**centos**/**6**.**8**/**contrib**/$basearch/**

#mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo=contrib

gpgcheck**=**1

enabled**=**0

gpgkey**=**http**://**mirror.centos.org**/**centos**/**RPM-GPG-KEY-CentOS-6

## 使用本地的光盘来制作一个yum源

有时候你的linux系统不能联网，当然就不能很便捷的使用联网的yum源了，这时候就需要你自己会利用linux系统光盘制作一个yum源。具体步骤如下：

a.挂载光盘

[root@fortest Server]# mount -t iso9660 -o loop /dev/cdrom /mnt

b.删除/etc/yum.repos.d目录所有的repo文件

[root@fortest Server]# rm -rf /etc/yum.repos.d/\*

c.创建新文件dvd.repo

[root@fortest Server]# vim /etc/yum.repos.d/dvd.repo

加入以下内容：

[dvd]

name=install dvd

baseurl=file:///mnt

enabled=1

gpgcheck=0

d.刷新repos,生成缓存

[root@fortest Server]#yum makecache

然后就可以使用yum命令安装你所需要的软件包了

2 利用yum工具下载一个rpm包

有时，我们需要下载一个rpm包，只是下载下来，拷贝给其他机器使用，前面也介绍过yum安装rpm包的时候，首先得下载这个rpm包然后再去安装，所以使用yum完全可以做到只下载而不安装。

a. 首选要安装 yum-downloadonly

# yum install -y yum-downloadonly.noarch

b. 下载一个rpm包而不安装

# yum install test.rpm -y --downloadonly //这样虽然下载了，但是并没有保存到我们想要的目录下，那么如何指定目录呢？

c. 下载到指定目录

# yum install test.rpm -y --downloadonly --downloaddir=/usr/local/src

# 第\*\*课-sudo用户管理

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

[root@redhat6 Desktop]# vim /etc/sudoers

用户名 主机名=（运行用户名） 可运行的命令

例如：

smb ALL=(ALL) /usr/sbin/useradd

smb ALL=(ALL) NOPASSWD: /usr/sbin/useradd

## Next comes the main part: which users can run what software on

## which machines (the sudoers file can be shared between multiple

## systems).

## Syntax:

## user MACHINE=COMMANDS

## The COMMANDS section may have other options added to it.

## Allow root to run any commands anywhere

root ALL=(ALL) ALL

## Allows members of the 'sys' group to run networking, software,

## service management apps and more.

# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROCESSES, LOCATE, DRIVERS

## Allows people in group wheel to run all commands

# %wheel ALL=(ALL) ALL

## Same thing without a password

# %wheel ALL=(ALL) NOPASSWD: ALL

## Allows members of the users group to mount and unmount the

## cdrom as root

# %users ALL=/sbin/mount /mnt/cdrom, /sbin/umount /mnt/cdrom

## Allows members of the users group to shutdown this system

# %users localhost=/sbin/shutdown -h now

# 第\*\*课-Linux密码故障排除

## 忘记root密码破解步骤

在系统启动倒计时马上回车进入grub选项菜单

在grub选项菜单按e进入编辑模式

编辑kernel那行添加 /init 1

按b重启系统即可进入单用户模式

编辑/etc/passwd的root那行，删除root:x:0..之间的那个x变成root::0..

重启系统登录Linux后设置新密码：# passwd root

## 单用户模式下无法修改文件

# cat /etc/fstab 或者 # cat /etc/mtab

# mount –o remount,rw /

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

## Root用户也无法修改文件

我们来查看一下该文件/目录权限：恩，是root可读可写的

[root@redhat6 test]# ls -l

total 4

drwxr-xr-x. 2 root root 4096 Nov 9 09:39 testdir

-rw-r--r--. 1 root root 0 Nov 9 09:39 test.txt

[root@redhat6 test]# cp test.txt testdir/#无法向目录写入文件？？

cp: cannot create regular file `testdir/test.txt': Permission denied

[root@redhat6 test]# echo "abc123"> test.txt

bash: test.txt: Permission denied#也无法向文件写入内容？？

[root@redhat6 test]# rm -rf test.txt #无法强制删除文件？这是几个意思？

rm: cannot remove `test.txt': Operation not permitted

[root@redhat6 test]# rm -rf testdir/#无法强制删除目录？这又是几个意思？

rm: cannot remove `testdir': Operation not permitted

[root@redhat6 test]# ls –l#再看看权限，简直不敢相信自己眼睛，原来亲眼看到的也不一定真是啊

total 4

drwxr-xr-x. 2 root root 4096 Nov 9 09:39 testdir

-rw-r--r--. 1 root root 0 Nov 9 09:39 test.txt

[root@redhat6 test]# lsattr -l

./test.txt Immutable, Extents

./testdir Immutable, Extents

[root@redhat6 test]# lsattr

----i--------e- ./test.txt

----i--------e- ./testdir

[root@redhat6 test]# chattr -i test.txt

[root@redhat6 test]# chattr -i testdir/

[root@redhat6 test]# chattr +i testdir/

[root@redhat6 test]# chattr =i testdir -Rv

testdir test.txt

[root@redhat6 test]# lsattr

-------------e- ./test.txt

----i--------e- ./testdir

[root@redhat6 test]# cp test.txt testdir/

cp: cannot create regular file `testdir/test.txt': Permission denied

[root@redhat6 test]# chattr -i testdir/

[root@redhat6 test]#

[root@redhat6 test]# mount

/dev/mapper/vg\_redhat6-lv\_root on / type ext4 (rw)

proc on /proc type proc (rw)

sysfs on /sys type sysfs (rw)

devpts on /dev/pts type devpts (rw,gid=5,mode=620)

tmpfs on /dev/shm type tmpfs (rw,rootcontext="system\_u:object\_r:tmpfs\_t:s0")

/dev/sda1 on /boot type ext4 (rw)

none on /proc/sys/fs/binfmt\_misc type binfmt\_misc (rw)

vmware-vmblock on /var/run/vmblock-fuse type fuse.vmware-vmblock (rw,nosuid,nodev,default\_permissions,allow\_other)

none on /sys/kernel/config type configfs (rw)

sunrpc on /var/lib/nfs/rpc\_pipefs type rpc\_pipefs (rw)

gvfs-fuse-daemon on /root/.gvfs type fuse.gvfs-fuse-daemon (rw,nosuid,nodev)

[root@redhat6 test]#

[root@redhat6 test]# chattr --help

Usage: chattr [-RVf] [-+=AacDdeijsSu] [-v version] files...

[root@redhat6 test]#

[root@redhat6 test]# man chattr | col -b > chattr.man

[root@redhat6 test]#

# 第\*\*课-GCC程序编译

GCC特点[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

GCC（GNU C Compiler）是GNU推出的功能强大、性能优越的多平台编译器，是GNU的代表作之一。GCC可以在多种硬件平台编译出可执行程序，其执行效率比一般的编译器相比平均效率要高20%~30%。

GCC范例gcc通过后缀来区别输入文件的类别

[root@redhat6 Desktop]#gcc编译流程

[root@redhat6 Desktop]# gcc -E hello.c –o hello.i 生成预编译文件

[root@redhat6 Desktop]# gcc -S hello.i –o hello.s 生成汇编源文件

[root@redhat6 Desktop]# gcc -c hello.s –o hello.o 生成链接文件（调试请加-g）

[root@redhat6 Desktop]# gcc -g hello.o –o hello.elf 生成执行文件（调试请加-g）

[root@redhat6 Desktop]#gcc其他编译选项

[root@redhat6 Desktop]# gcc hello.c –I /home/include–o hello.elf （包含其他头文件目录）

[root@redhat6 Desktop]# gcc –DMACRONAME hello.c –o hello.elf

（等效于在hello.c中使用#define MACRONAME）

[root@redhat6 Desktop]#

[root@redhat6 Desktop]# gcc -Wall -Os –g hello.c –o hello.elf

[root@redhat6 Desktop]# gcc -Wall -Os –g –static hello.c –o hello.elf （静态编译）

[root@redhat6 Desktop]# gcc –g –c hello.c && gcc –g hello.o –o hello.elf（调试阶段）

[root@redhat6 Desktop]#

[root@redhat6 Desktop]#

[root@redhat6 Desktop]#

# 第\*\*课-开发板介绍

系统资源：处理器、norflash、nandflash、RAM、LCD

开发板板载接口资源

电源、串口、网口、USB从口、USB主口、音频口、系统总线、JTAG、SD卡、GPIO、LCD、摄像头、复位键

CPU、内存、norflash、nandflash

启动方式

从norflash启动、从nandflash启动、从SD卡启动。

支持的操作系统:Linux、winCE、Android

系统安装方式

1 通过JTAG将引导程序烧写到norflash

2 通过norflash启动来获取操作系统下载到nandflash

3 通过nandflash启动操作系统

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

# 第\*\*课-开发板串口连接

开发板串口：相当于开发板输入输出。

笔记本串口：通过USB转串口驱动+USB转串口线来连接开发板。

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

串口连接准备：

1 硬件连接（驱动预安装（绿联USB转串口驱动））

2 计算机 >>属性 >>设备管理器 >>端口(COM3)

3 设置secureCRT>>协议serial >>端口COM3 >>波特率115200 >>数据位8 >>取消流控

# 第\*\*课-开发板系统安装准备

开发板软件构成：bootloader >>> kernel >>>文件系统

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

2440开发板系统安装步骤：引导程序 >>> JTAG >>> norflash >>> USB下载线 >>> nandflash

JTAG硬件准备：USB线 + JLINK主板 + JTAG线 + 转接板 + 开发板JTAG转接线

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

Windows的JLINK软件：SEGGER 》》 J-LINK ARM 》》 J-Flash ARM

烧写前检查：flash开关拨到norflash开关,连接好JLINK与板子,打开开发板电源

载入工程配置：File 》》 Open Project》》2440.jflash

建立连接：Target 》》 Connect

载入要引导的程序：File 》》 Open》》 supervivi-128.bin》》0

芯片擦除：Target >>> Erase chip >>> 0x0

芯片烧写：Target >>> Program

关闭开发板电源 》》 拔掉JLINK线 》》 连接串口 》》 打开电源 》》 查看串口输出

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

# 第\*\*课-安装Linux到开发板

安装USB下载驱动：dnw.ko

启动开发板 》》 格式化nandflash 》》 等待下载裸机程序 》》 Linux中安装dnw驱动 》》 通过Linux中dnw下载内核文件 》》 下载完成 》》 继续下载文件系统（donwload root\_yaffs..） 》》 从nandflash启动操作系统。

# 第\*\*课-C语言基础与扩展

[主目录](#_目录)

# 让你不再害怕指针(201612061859)

复杂类型说明

要了解指针，多多少少会出现一些比较复杂的类型，所以我先介绍一下如何完全理解一个复杂类型，要理解复杂类型其实很简单，一个类型里会出现很多运算符，他们也像普通的表达式一样，有优先级，其优先级和运算优先级一样：**从变量名处起,根据运算符优先级先后结合，一步一步分析**。--(摘自网络经验)

下面我们根据这条原则来做实例分析

int p; //这是一个普通的整型变量

int \*p; //首先从p处开始,先与\*结合,所以得到p是一个指针;

//其次再与int结合,可以得出该指针指向一个整型数据;

//所以p是一个返回整型数据的指针。

int p[3]; //(1)首先从p开始,先与[]结合,说明p是一个数组;

//(2)然后与int结合,说明数组元素是整型;

//所以p是一个整型数据组成的数组。

int \*p[3]; //(1)p处开始,先与[]结合,得到p是一个数组;

//(2)再与\*结合,得到p数组里的元素是指针类型;

//(3)再与int结合,说明指针指向的内容是整型的;

//所以p是一个由返回整型数据的指针所组成的数组。

int (\*p)[3]; //(1)p处开始,先与\*结合,得到p是一个指针;

//(2)再与[]结合,说明该指针指向一个数组;

//(3)再与int结合,说明数组里的元素是整型的;

//所以p是一个由整型数据组成的数组的指针。

int \*\*p; //(1)从p开始,先与\*结合,说明p是一个指针;

//(2)再与\*结合,说明指针指向的元素是指针;

//(3)然后与int结合,说明该指针指向一个整型数据;

int p(int); //(1)从p开始,先与()结合,说明p是一个函数;

//(2)然后进入()里分析,说明该函数有个整型变量参数;

//(3)然后再与外面的int结合,说明该函数的返回值是一个整型数据。

int (\*p)(int); //(1)从p开始,先与\*结合,说明p是一个指针;

//(2)然后与()结合,说明该指针指向一个函数;

//(3)然后与()里的int结合,说明该函数有一个int类型参数;

//(4)再与最外的int结合,说明函数的返回类型是整型。

int \*(\*p(int))[3]; //(1)从p开始,先与()结合,得到p是一个函数;

//(2)再与()内int结合,说明该函数有一个int型参数;

//(3)再与\*结合,说明函数返回的是一个指针;

//(4)然后到最外层,先与[]结合,说明返回的指针指向一个数组;

//(5)然后再与\*结合,说明数组里的元素是指针;

//(6)然后再与int结合,说明指针指向的内容是整型数据.

## const\*只读定义

const在\*的左边：数据就是只读的

const在\*的右边：指针就是只读的 （左数右指）

char a,b,c;

const char \*ptr2=&a; //指针可以改变,数据为只读

char \* const ptr1=&a; //数据可以改变,指针为只读

\*ptr1 = 15;

\*ptr1 = 16;

ptr2 = &b;

ptr2 = &c;

## 结构体指针定义和结构内指针

本小结知识点：

代码段：代码、全局常量、字符串常量

数据段：全局变量、静态变量

BSS段：除了栈以外的所有未初始化变量

堆：动态分配的空间

栈：局部变量（不含静态变量），局部只读变量，局部常量

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct tag\_student **{**

int num**;**

char **\***name**;**

**};**

int main **(**void**)**

**{**

struct tag\_student **\***st1 **=** **NULL;**

st1 **=** **(**struct tag\_student **\*)**malloc**(sizeof(**struct tag\_student**));**

st1**->**num **=** 10**;**

st1**->**name **=** "seafly"**;**//这种方式为字符串常量方式,存储在代码段

//st1->name = (char \*)malloc(sizeof(char)\*10);

//memcpy(st1->name, "seafly", 6);//这种数组操作型就必须为其动态分配内存

printf **(**"num: %d\n"**,** st1**->**num**);**

printf **(**"name: %s\n"**,** st1**->**name**);**

**return** 0**;**

**}**

## 数字菜单实现原理

#include <stdio.h>

int main **(**void**)**

**{**

//char a = 0;

int a **=** 0**;**

int flag **=** 1**;**

printf **(**"1.print 1\n"**);**

printf **(**"2.print 2\n"**);**

printf **(**"3.print 3\n"**);**

**while** **(**flag**)**

**{**

printf **(**"choose: "**);**

scanf **(**"%d"**,** **&**a**);**

**switch** **(**a**)**

**{**

**case** 1**:**

printf **(**"111111111111\n"**);**

printf **(**"111111111111\n"**);**

**break;**

**case** 2**:**

printf **(**"222222222222\n"**);**

printf **(**"222222222222\n"**);**

**break;**

**case** 3**:**

printf **(**"333333333333\n"**);**

printf **(**"333333333333\n"**);**

**break;**

**default:**

printf **(**"EEEEEEEEEEEE\n"**);**

printf **(**"EEEEEEEEEEEE\n"**);**

flag **=** 0**;**

**break;**

**}**

**}**

**return** 0**;**

**}**

## 共用体union的用法

由于目前的硬件系统都采用覆盖技术，所以联合体中只有最后被赋值的成员才是有效值。

## 处理器大小端(大端小端)的测试

1 #include <stdio.h>

2 union C

3 {

4 int i;

5 char c;

6 };

7

8 int main (void)

9 {

10 union C c;

11 c.i = 1;

12 if (1 == c.c)

13 printf ("little endian");

14 else

15 printf ("big endian");

16 printf ("< i=%d, c=%d >\n", c.i, c.c);

17

18 return 0;

19 }

20

[root@redhat6 Desktop]# gcc test.c

[root@redhat6 Desktop]# ./a.out

little endian < i=1, c=1 >

[root@redhat6 Desktop]#

神总结：因为大小端区别主要是由2个字节的数据，其他字节不用管。

# 第\*\*课-C语言底层原理

相关资源网站

吾爱破解官网：<http://www.52pojie.cn/>

吾爱网盘：<http://down.52pojie.cn/>

常用反汇编工具:OD(OllyDbg v2.01.zip)

## C语言函数传参底层汇编实现原理

1.给函数传递参数  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<stdio.h>

int Add(int x, int y)

{

                int sum;

                sum = x + y ;

                return sum;

}

int main()

{

                int z;

                z = Add(1, 2);

                printf( "z = %d\n", z);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

013F182E 6A 02                        push        2

013F1830 6A 01                        push        1

013F1832 E8 20 FB FF FF       call        013F1357

013F1837 83 C4 08                   add         esp,8

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

c语言的参数传递是从右到左的  
在执行 z = Add(1, 2);的时候  
先push了2,esp = 0028F89C   
然后push了1,esp = 0028F898  
最后call了01341357 esp = 0028F894  
0028F894 中保存的地址是函数的返回地址37 18 3f 01,也就是013f1837  
01341357 是一个跳转地址,jmp         013F16D0  
跳转至 函数的入口地址 013F16D0  
  
2.函数获取参数  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<stdio.h>

int Add(int x, int y)

{

                int sum;

                sum = x + y ;

                return sum;

}

int main()

{

                int z;

                z = Add(1, 2);

                printf( "z = %d\n", z);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
在函数执行之前,首先push了两个参数,  
然后call到了跳转语句,每次esp-4  
所以1的存储地址就是esp+4,2的存储地址就是esp+8  
但是由于ESP随栈的变化而变化,所以为了计算简便,可以将该值存储到另一个寄存器,如EBP  
那么刚才的两个地址就是EBP+4和EBP+8了  
不过由于EBP本身是有值的,如果直接赋值修改它,那么当需要使用EBP的原值得时候,就没办法了  
例如函数A调用后返回,如果不还原EBP,函数B用EBP作为基点就会出错,因此需要一个保存和灰度EBP值得过程,  
下面的代码完成了这一过程,将EBP压栈,最后pop出来,赋值给ebp,从而还原  
push ebp  
...  
pop ebp  
  
所以在函数开始的地方存在如下代码:  
push ebp       //将ebp压栈  
mov ebp,esp  //将esp的值给ebp  
  
因为在栈里又压入了ebp的缘故,所以esp将再减少4字节  
此时再计算参数1和2的存储地址,存储1的地址是ebp+8,存储2的地址是ebp+0ch(12)  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

sum = x + y;

013F16EE 8B 45 08             mov         eax,dword ptr [ebp+8]

013F16F1 03 45 0C             add         eax,dword ptr [ebp+0Ch]

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**注释:**

**mov 将ebp + 8的值给eax,ebp + 8是1,也就是把1给eax**

**然后ebp + 0ch的值加上eax的值,ebp+0ch的值是2,eax的值是1,1+2 = 3,将结果保存在eax中**

## C语言局部变量底层汇编实现原理

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<stdio.h>

int Add(int x, int y)

{

                int sum;

                sum = x + y ;

                return sum;

}

int main()

{

                int z;

                z = Add(1, 2);

                printf( "z = %d\n", z);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
注释:  
局部变量的特点和参数一样,当函数调用完毕就不再使用,所以效仿参数,将其分配在栈上.  
栈上方已经被参数等等使用,我们是有使用栈更低地址的空间,也就是继续压栈来分配局部变量  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

sum = x + y;

013F16EE 8B 45 08             mov         eax,dword ptr [ebp+8]

013F16F1 03 45 0C             add         eax,dword ptr [ebp+0Ch]

013F16F4 89 45 F8             mov         dword ptr [ebp-8],eax //将结果放入ebp-8,也就是sum中

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
注释:  
之所以是ebp-8而不是ebp-4是因为在VC6以后为了防止溢出攻击,所以产生了该现象  
不过至此也有了一个简单的结论:**在使用EBP寻址的函数中,ebp+偏移量就是参数,ebp-偏移量,就是局部变量**

## C语言指针的底层汇编实现原理

**指针的赋值过程与指针类型的真相**

**int指针:**

int gi;

int \*p;

p = &gi;

00EF166E C7 05 3C 91 EF 00 38 91 EF 00  mov         dword ptr ds:[00EF913Ch],0EF9138h

\*p = 12;

00EF1678 A1 3C 91 EF 00                           mov         eax,dword ptr ds:[00EF913Ch]

00EF167D C7 00 0C 00 00 00                      mov         dword ptr [eax],0Ch

**注释:**

p = &gi;

mov         dword ptr ds:[00EF913Ch],0EF9138h       将gi的地址0EF9138h放入00EF913Ch所指向的地址中保存

\*p = 12;

mov          eax,dword ptr ds:[00EF913Ch ]                 将00EF913Ch所存储的地址,也就是gi的地址0EF9138h放入eax中.

mov          dword ptr [eax],0Ch                                    将0ch放入eax所存储的地址中,也就是把0ch放在了gi所代表的地址中,dword也就代表了向eax中写入4字节数据  
  
  
**short型指针:**

short gi;

short \*p;

p = &gi;

0034166E C7 05 3C 91 34 00 38 91 34 00 mov         dword ptr ds:[0034913Ch],349138h

\*p = 12;

00341678 B8 0C 00 00 00                                          mov         eax,0Ch

0034167D 8B 0D 3C 91 34 00                                   mov         ecx,dword ptr ds:[0034913Ch]

00341683 66 89 01                                                      mov         word ptr [ecx],ax

**注释:**

p = &gi;

mov         dword ptr ds:[0034913Ch],349138h         将gi的地址0EF9138h放入00EF913Ch所指向的地址中保存

\*p = 12;

mov         eax,0Ch                                                        将0Ch放入eax中,由于eax是4字节,所以0ch放到了eax的低2字节,ax中

mov         ecx,dword ptr ds:[0034913Ch]                    将p所存储的地址,也就是gi的地址放入ecx中.

mov         word ptr [ecx],ax                                          将ax所存储的内容写入到ecx所指向的地址,也就是gi的地址,.word说明将向gi的地址写入2字节数据

**char型指针:**

char gi;

char \*p;

p = &gi;

00CC166E C7 05 3C 91 CC 00 38 91 CC 00 mov         dword ptr ds:[00CC913Ch],0CC9138h

\*p = 12;

00CC1678 A1 3C 91 CC 00                                           mov         eax,dword ptr ds:[00CC913Ch]

00CC167D C6 00 0C                                                      mov         byte ptr [eax],0Ch

**注释:**

p = &gi;

mov         dword ptr ds:[00CC913Ch],0CC9138h        将gi的地址0EF9138h放入00EF913Ch所指向的地址中保存

\*p = 12;

mov         eax,dword ptr ds:[00CC913Ch]                     将00EF913Ch所存储的地址,也就是gi的地址0EF9138h放入eax中.

mov         byte ptr [eax],0Ch                                           将0ch放入eax所存储的地址中,也就是把0ch放在了gi所代表的地址中,byte 也就代表了向eax所存储的地址中写入1字节数据

**总结:**

     C语言指针包含两方面信息,一个是地址,存放在指针变量中,另一个是类型信息,决定了读写的长度,没有存储在指针变量中,位于指针读写过程所使用的mov指令中,不同的指针类型对应着读写的字节数.

这也间接解释了指针为什么+-1不是+-1个字节,而是加减长度是指针类型的字节数,void类型的指针之所以无法加减,就是因为它没有类型信息,无法确定加减的长度,代表的仅仅是一块内存地址.

指针类型强制转换

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**程序:**

int i;

int \* pi;

short \* ps;

char \* pc;

int main()

{

                pi = &i;

                ps = ( short \*)&i;

                pc = ( char \*)&i;

                \*pi = 0x1234;

                \*ps = 0x1234;

                \*pc = 0x12;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
**反汇编:**

pi = &i;

00941A8E C7 05 E0 94 94 00 DC 94 94 00 mov           dword ptr ds:[009494E0h],9494DCh

ps = (short \*)&i;

00941A98 C7 05 E4 94 94 00 DC 94 94 00 mov           dword ptr ds:[009494E4h],9494DCh

pc = (char \*)&i;

00941AA2 C7 05 E8 94 94 00 DC 94 94 00 mov          dword ptr ds:[009494E8h],9494DCh

\*pi = 0x1234;

**注释:**

在把i赋值其他指针变量的过程中,强制转换并没有生成任何指令,可见,强制转换并不是在这里产生效果的

00941AAC A1 E0 94 94 00                                            mov         eax,dword ptr ds:[009494E0h]

00941AB1 C7 00 34 12 00 00                                        mov         **dword** ptr [eax],1234h

\*ps = 0x1234;

00941AB7 B8 34 12 00 00                                              mov         eax,1234h

00941ABC 8B 0D E4 94 94 00                                       mov         ecx,dword ptr ds:[009494E4h]

00941AC2 66 89 01                                                         mov         **word** ptr [ecx],ax

\*pc = 0x12;

00941AC5 A1 E8 94 94 00                                             mov         eax,dword ptr ds:[009494E8h]

00941ACA C6 00 12                                                       mov         **byte** ptr [eax],12h

**注释:**

可以看出,在这里,写入的字节数发生了变化,所以,强制转换的效果不在转换过程中体现,而是体现在转换后去访问内存时的指令中

**注释2:**

如果转换后指针指向的数据类型大小小于原来的数据类型大小,那么用该转换后的指针访问就不会越过原数据的内存,是安全的,否则危险,要越界

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**程序:**

#include<stdio.h>

int main()

{

                int \* pi;

                short si = 12;

                pi = ( int \*)&si;

                printf( "%d,%x", \*pi, \*pi);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

注释:

正常情况下这段程序应该输入的是12,0c

但是实际上的运行结果是-859045876,cccc000c

这是由于将si的地址强制转换为int \*类型,然后赋值给pi;那么pi会访问4字节,这时越界了,将si后的2字节纳入了范围,即是cc cc,他们和0c 00合在一起正好与结果吻合

这就是违反了强制转换的原则,越界了

## 汇编代码求常数寻找方法

汇编遇到这样的代码：  
mov dword ptr ds:[eax],0x65821458  
mov dword ptr ds:[eax+0x4],0x58595655  
mov dword ptr ds:[eax+0xC],0xC4B0DBB8  
这三句代码中的三个常数（65821458，58595655，C4B0DBB8）

## 让我们通过享受来学习堆栈的知识

首先你可以把栈当作一摞书，很多本层叠起来的一摞书。

其次

## 进程与线程的简单解释

**1计算机的核心是CPU，它承担了所有的计算任务。它就像一座工厂，时刻在运行。**

**2假定工厂的电力有限，一次只能供给一个车间使用。也就是说，一个车间开工的时候，其他车间都必须停工。背后的含义就是，单个CPU一次只能运行一个任务。**

**3进程就好比工厂的车间，它代表CPU所能处理的单个任务。任一时刻，CPU总是运行一个进程，其他进程处于非运行状态。**

4一个车间里，可以有很多工人。他们协同完成一个任务。

**5线程就好比车间里的工人。一个进程可以包括多个线程。**

**6车间的空间是工人们共享的，比如许多房间是每个工人都可以进出的。这象征一个进程的内存空间是共享的，每个线程都可以使用这些共享内存。**

**7可是，每间房间的大小不同，有些房间最多只能容纳一个人，比如厕所。里面有人的时候，其他人就不能进去了。**

**这代表一个线程使用某些共享内存时，其他线程必须等它结束，才能使用这一块内存。**

**8一个防止他人进入的简单方法，就是门口加一把锁。先到的人锁上门，后到的人看到上锁，就在门口排队，等锁打开再进去。**  
**这就叫"互斥锁"（Mutual exclusion，缩写 Mutex），防止多个线程同时读写某一块内存区域。**

**9还有些房间，可以同时容纳n个人，比如厨房。也就是说，如果人数大于n，多出来的人只能在外面等着。这好比某些内存区域，只能供给固定数目的线程使用。**

**10这时的解决方法，就是在门口挂n把钥匙。进去的人就取一把钥匙，出来时再把钥匙挂回原处。后到的人发现钥匙架空了，就知道必须在门口排队等着了。这种做法叫做"信号量"（Semaphore），用来保证多个线程不会互相冲突。**

不难看出，mutex是semaphore的一种特殊情况（n=1时）。也就是说，完全可以用后者替代前者。但是，因为mutex较为简单，且效率高，所以在必须保证资源独占的情况下，还是采用这种设计。

**0x03 操作系统的设计，因此可以归结为三点**

（1）以多进程形式，允许多个任务同时运行；

（2）以多线程形式，允许单个任务分成不同的部分运行；

（3）提供协调机制，一方面防止进程之间和线程之间产生冲突，另一方面允许进程之间和线程之间共享资源。

# C语言基础(传智播客扫地僧)

# C语言提高(传智播客扫地僧)

day01

## C工程开发需要什么能力？

企业需要能干活的人

企业里面不缺人，企业里缺能干活的人

C学到什么程度可以找工作？

C工程开发需要什么（培养什么能力）

成熟的、商业化的信息系统需要分区和分层

信息系统的技术模型也分层

做项目才能入行

**接口的封装和设计**

**接口的封装和设计**

**模块的划分**

**断层思维**

**接口API的使用能力**

**接口API的查找能力**

**接口API的实现能力**

**建立正确程序运行内存布局图**

**内存四区模型图**

**函数调用模型图**

C项目开发的套路（一套接口）

BS模型：http协议、https协议

CS模型：数据报文格式

客户端-----------------------自定义报文---------------------服务端

模块的划分：A工程师去做上层应用，B工程师做底层接口开发

## 指针标准和知识体系搭建和在项目中的应用

typedef struct \_sockclient {

char version[16];

char serverip[64];

int port;

unsigned char \*pbuf; //数据指针

int ilen;

}SOCK\_HANDLE;

int client\_socket\_init1 (void \*\*handle)

{

char ret = 0;

SOCK\_HANDLE \*hd1 = NULL;

hd1 = (SOCK\_HANDLE \*)malloc(sizeof(SOCK\_HANDLE));

if (NULL == hd1) {printf();return ret=-1;}

memset (hd1, 0, sizeof(SOCK\_HANDLE));

hd1->ilen = 0;

hd1->pbuf = NULL;

strcpy (hd1->serverip, “192.168.1.10”);

hd1->port = 8081;

\*handle = (void \*)hd1;

return 0;

}

void \*client\_socket\_init2 (void \*handle)

{

SOCK\_HANDLE \*hd1 = NULL;

hd1 = (SOCK\_HANDLE \*)malloc(sizeof(SOCK\_HANDLE));

if (NULL == hd1) {printf();return hd1;}

memset (hd1, 0, sizeof(SOCK\_HANDLE));

hd1->ilen = 0;

hd1->pbuf = NULL;

strcpy (hd1->serverip, “192.168.1.10”);

hd1->port = 8081;

\*handle = (void \*)hd1;

return handle;

}

int main (void)

{

SOCK\_HANDLE \*handle = NULL;

我们常见的使用方式1:

if (0 != client\_socket\_init1 (&handle)) exit (-1);

使用方式2:

if ((handle = client\_socket\_init2 (handle))==NULL) exit(-1);

return 0;

}

指针步长概念

## 指针和链表在项目开发中的实战应用

传统链表范例结构：

typedef struct \_list {

void \*data;

struct \_list \*prev, \*next;

};

Linux内核链表范例结构：

struct \_list {

struct \_list \*prev, \*next;

};

struct \_data {

void \*data;

struct \_list list;

};

企业级通用链表范例结构：

struct \_list { //节点模型

struct \_list \*prev, \*next;

};

typedef struct \_node {

struct \_node \*prev, \*next;

}NODE;

struct \_head {

NODE header;

int length;

};

struct \_data { //业务模型

NODE linker; //链表节点域(指针域)

void \*data; //链表的数据域(链表算法与业务节点分离思想)

};

常用链表算法：创建、删除节点、插入节点、取出节点、销毁

链表算法设计经验：抽象的地方用简单例子来理解。

# C++基础(传智播客扫地僧)

## 视频教程索引：C++基础核心/01\_C++基础/c++基础day01~c++基础day08

## C++程序框架

//头文件包含

#include <iostream>

//使用命名空间std

using namespace std;

int main(void)

{

//cout: 标准输出,黑屏幕

//其中左移操作符是C++语言操作重载

//endl: 代表回车换行

cout << “hello,world” << endl;

return 0;  
}

## 面向过程和面向对象的区别

用面向过程方法求圆面积

using namespace std;

double r = 0;

double s = 0;

cout << “请输入圆半径: ” << endl;

cin >> r; //cin标准输入,一般为键盘

s = 3.14\*r\*r;

cout << “圆面积: ” << s << endl;

用面向对象方法求圆面积

//C语言中抽象一个对象的方法

struct Circle { //抽象圆这个对象

double m\_s; //圆的面积

double m\_r; //圆的半径

};

//C++中抽象一个对象的方法:

class mycircle { //通过类来抽象圆这个对象

public: //定义该类的属性

double m\_s; //圆的面积

double m\_s; //圆的面积

public: //定义该类的成员函数

void setr (double r) //成员函数

{

m\_r = r;

}

double getr (void)

{

return m\_r;

}

void sets (void)

{

m\_s = 3.14\*m\_r\*m\_r;

return ;

}

double gets(void)

{

return m\_s;

}

protected:

private:

};

int main (void)

{

mycircle c1,c2,c3; //用类 定义 变量 对象

double r = 0;

cout << “请输入圆半径: ” << endl;

cin >> r; //cin标准输入,一般为键盘

c1.setr(r);//给C1圆的属性赋值

c1.sets();

cout << “圆的面积为: ” << c1.gets() << endl;

return 0;

}

//结论1：面向过程加工的是一个一个的函数,面向对象加工的是一个一个的类

//结论2：面向对象类的调用==》不是一步一步执行的。当调用成员函数该函数才执行。

//结论3：类是一个数据类型(固定大小内存块的别名)。

//结论4：定义一个类，是一个抽象的概念，不会给你分配内存，

//结论5：用数据类型定义变量的时候，才会分配内存。

//结论6：class类是一个数据类型,和对象关系是”1:n”

//课后问题抛出:C++编译器是如何处理多个对象，具体一点，就是如何区分是c1,c2,还是c3调用了类中的函数

## 初学者经典错误:类中不写成员函数

问题引出:为什么要有成员函数?

//上一节课我们的类使用了成员函数

class circle { //此类没有成员函数

public:

double r;

double pi = 3.1415926;

double s = pi\*r\*r;

};

int main (void)

{

circle c1;

cin >> c1.r;

count << c1.s << endl; //运行之后乱码或者运行出错

//因为类中这个在初始化的时候已经执行了，且当时其中的r是一个乱码

//当执行c1.s时只是从其中取值,并没有执行pi\*r\*r;

}

//结论1：类中增加成员函数的作用:当调用的时候就是执行该运算了。

class circle {

public:

double r;

double pi = 3.1415926;

double s = pi\*r\*r;

double sets (void) //修改:向该类增加相关成员函数

{

s = pi\*r\*r;

return s;

}

};

//结论2：类中的成员函数，只有对象在调用的时候才执行成员函数。

## 程序设计方法发展历程

面向过程:树状图类型逐步细化的过程的处理方式

面向对象:比如界面程序菜单功能，它们一开始就存在，但该功能并没有被执行，只有用户需要点开它的时候它才会执行并处理相关的子任务。

C语言中的struct只定义了属性, C++中的类把属性和方法做了一个封装

面向对象软件工程思想：

OOA：面向对象的分析(Analysis)

OOD: 面向对象的设计(Design)

OOP: 面向对象的编程(Program)

OOT: 面向对象的测试(Test)

OOSM:面向对象的软件维护(Software Managerment)

面向对象程序设计：由现实世界建立软件模型

## 工作经验分享-C和C++关系

C语言是在实践过程中逐步完善起来的。没有深思熟虑的设计过程，使用时存在很多“灰色地带”，残留过多低级语言的特性，直接利用指针进行内存操作。

C语言的目标是高效。

C语言 + 面向对象方法论 ===》C++、Obj-C

C++是C语言的加强，并且完全兼容C语言的特性。

学习C++可以接触更多的软件设计方法，并带来更多的机会，C++是java/C#高级语言的学习基础，C++是各大知名软件企业挑选人才的标准之一。

工作经验小常识：

做C的或C++的程序员普遍没有JAVA的程序快速接受新知识的能力强，由于C/C++大部分做底层，所以接触的算法什么的都比较经典也就那么几种算法和数据结构。

软件和硬件打交道的接口：寄存器(控制寄存器、读寄存器、写寄存器)，然后在寄存器之上，就是C语言接口，如果这个接口套上一层windows驱动，则可以在插在windows之下，如果接口套上Linux驱动，则可以用在Linux系统上；驱动之上就是应用接口(国际标准)，

再上面就是信息系统(业务流).

没有看不懂的代码，只有看不懂的设计。

硬件工程师设计硬件电路，并印刷电路板，制作开发板，测试开发板硬件功能。

## 命名空间namespace概念

C中的命名空间：在C语言中只有一个全局作用域，C语言中所有全局标识符共享同一个作用域，当项目过于庞大之后标识符之间很可能容易发生冲突。

C++中的命名空间：命名空间将全局作用域分成不同的部分，不同命名空间的标识符可以同名而不会发生冲突，命名空间可以相互嵌套，全局作用域也叫默认命名空间。

命名空间的定义：namespace name { ... }

C++命名空间编程实践

namespace NameSpaceA { int a = 0; }

namespace NameSpaceB {

int a = 1;

namespace NameSpaceC { //命名空间的嵌套

struct Teacher {

char name[10];

int age;

};

}

}

int main (void)

{

using namespace NameSpaceA;

using namespace NameSpaceB::NameSpaceC::Teacher;//就可以使用Teacher

printf (“a = %d\n”, a);

printf (“a = %d\n”, NameSpaceB::a);

Teacher t1 = {“aaa”, 30};

printf (“t1.name: %s\n”, t1.name);

printf (“t1.age: %d\n”, t1.age);

}

<iostream>文件中没有引用标准的命名空间,也就是该文件中没有这句”using namespace std;”

结论1：C++标准为了区分C，也为了正确使用命名空间，规定头文件不使用后缀.h

结论2：C++命名空间定义：namespace name { ...; }

结论3：C++命名空间使用：using namespace namespaceA;

结论4：namespace定义可嵌套。

## C++实用性加强

//C语言中的变量必须在作用域开始的位置定义！

//C++中更强调语言的“实用性”，所有的变量都可以在需要使用时再定义。

## register关键字增强

//C语言中使用register定义的寄存器变量r1,不能读取r1地址。

//C++中使用register定义的寄存器变量r1,但是可以读取r1地址；并且使用比较频繁的变量它也会把该变量自动register.

## 变量检测的增强

在C语言中，重复定义多个同名的全局变量是合法的；在C++中，不允许定义多个同名的全局变量，C语言中多个同名的全局变量最终会被链接到全局数据区的同一个地址空间上。

## struct关键字类型增强

C语言中的struct定义了一组变量的集合，C编译器并不认为这是一种新的类型；

C++中的struct是一个新类型的定义声明。

struct Teacher {

char name[32]; int age;

};

//C中使用的时候必须是这样: struct Teacher t1,t2;

//C++中使用时会可以直接定义：Teacher t1,t2;

//struct 和 class关键字完成的功能是一样的，也就是struct中也能用public:等。

## C++中的变量和函数都必须有类型

C++中所有变量和函数都必须有类型，C语言中的默认类型在C++中是不合法的。

比如C语言中可以这样定义函数：

func(i)

{

printf (“%d”, i);

return 100;

}

## C++中新增的数据类型:bool类型

sizeof(bool)占用一个字节内存空间，

bool定义的变量值要么是1(true)要么是0(false),且负数也是非零(true)。

## C/C++中的三目运算符

int a=10,b=20;

(a < b? a:b)=30; //在C中该表达式返回是一个值:10

//C语言中的三目运算符是一个表达式，表达式不能做左值。

//C语言中表达式的返回值放在寄存器中。

//C语言中表达式的返回值是一个数，所以不能做左值。

//C++中表达式返回的是变量的本身，所以上述代码中在C++中允许使用。

(a < b? a:b)=30; //在C中该表达式返回是一个变量本身:a

如何让C语言中的三目运算法当左值？

int a=10,b=20;

(a < b? a:b)=30; //在C中该表达式返回是一个值:10

//当左值必须条件:必须是寄存器或内存空间！

//能当左值，其属性应当是一个变量(内存空间)的性质。

//(a < b? a:b)=30;要返回一个内存空间，也就是内存首地址，也就是指针。

//我们不想让它返回a的值，而是返回a的地址，返回b的地址，所以我们的修改如下：

//\*(a < b? &a:&b) = 30;//C指针的灵活运用！

## C/C++中的const

C中的左数右指

//const 在 \* 左边，则作用于该内存空间中的具体数据。

//const 在 \* 右边，则作用于指针本身.

const为C中的冒牌货

//C语言中的const定义的常量是一个冒牌常量，因为其可以通过指针间接改变。

//C++中的const定义的常量是一个货真价实的常量，即使通过指针间接也改变不了其值。

//C++中const符号表实现机制：C++编译器扫描到const定义时候，不会像C一样单独给它分配内存，而是把该常量放在一个符号表里，我们不是C++编译器开发者我们也不知道编译器把该常量放到内存四区的哪个区里了，我们只需要知道C++编译器只是对这个进行特别特殊处理。即使我们使用指针取该常量地址，C++编译器会单独给我们这个指针指向的地址单独分配一个内存，所以我们对该内存的修改和那个符号表里的常量没有半毛钱关系。

## const和#define的对比

int a = 10;

int b = 20;

int array[a+b] = {0};

//C/C++都不支持，数组下标不能是变量，但Linux内核除外，因为Linux内核的gcc编译器支持该写法。

const int a = 10; 等同于#define a 10

const int b = 20; 等同于#define b 20

int array[a+b] = {0};

//C/C++都支持，编译是可以通过的。

//const用途:在C++中是用来替换#define的一个手段

//const常量是由编译器处理，提供类型检查和作用域检查。

//#define宏定义由预处理处理，单纯的文本替换。

//如何#define作用域限制在某个函数里面？如下：

void func1 (void)

{

#define a 10

const int b = 20;

#undef a

}

## 普通引用

变量名回顾

变量名实质上是一段连续存储空间的别名，是一个门牌标号。

程序中通过变量来申请并命名内存空间。

通过变量的名字可以使用存储空间。

问题1：对一段连续的内存空间只能取一个别名吗？

引用：就是给一个已定义的变量别名。

引用的语法：Type &name = var;

int a = 10;

int &b = a; //b就是a的别名

b = 100; //相当于把a改成100了。

//引用是C++的语法范畴，所以当你看到引用的时候，你就不能用C的语法来分析程序了。

//普通引用必须奥初始化，也就是必须要依附一个右值。

//引用做函数参数

void swap (int &a, int &b)

{

int c = a;

a = b;

b = c;

c = a;

}

int main (void)

{

int x=10, y=30;

swap (x, y);

}

## 复杂数据类型如何引用

范例代码：

struct Teacher {

int age;

};

void print1(Teacher \*pt) { printf(“age: %d\n”, pt->age);}

void print2(Teacher &pt) { printf(“age: %d\n”, pt.age);}

void print3(Teacher pt) { printf(“age: %d\n”, pt.age);}

int main (void)

{

Teacher t1; t1.age = 35;

print1(&t1); //pt为指针指向t1地址

print2(t1); //pt是t1别名,相当于t1，C++编译器已经帮我们取地址了

print3(t1); //此处是形参，相当于pt = t1

}

## 引用的本质分析

//引用作为其他变量的别名而存在，因此在一些场合可以代替指针。

//引用相对于指针来说具有更好的可读性和实用性。

//思考：C++编译器为引用背后做了什么呢？

int a = 10;

int &b = a;

//我们上面讨论过，单独定义引用时候，必须初始化，所以它的定义行为很像一个常量。

//回顾引用定义：给一个内存空间取一个别名。

//上面一句验证：读取a,b地址，结果显示两地址相同。

struct Teacher {

int a[10]; //40字节

int b[5]; //20字节

int &c; //4 0 //通过sizeof()的检测，结果为4字节，说明很类似指针。

int &d; //4 0

};

所以综上结论得出：

引用在C++中的内部实现是一个常量指针：Type \* const name;

C++编译器在编译中用常指针作为引用的内部实现，因此引用所占空间和指针相同。

从使用角度，引用会让人误会其只是一个别名，没有自己的存储空间，这是C++为了实用性而做出的细节隐藏。

//当我们使用引用语法时，我们不去关心编译器引用是怎么做的。

//当我们奇怪的语法现象时，我们才去考虑C++编译器是怎么做的。

## 引用作为函数返回值

int getaa1(void)

{

int a = 10;

return a;

}

//返回a的本身

int &getaa2(void)

{

int a = 10;

return a;

}

int \*getaa3(void)

{

int a = 10;

return &a;

}

int main (void)

{

int a1 = getaa1();

int a2 = getaa2(); //接引用方法1：此处它把该函数里面a的副本返回给a2,所以其值就是a2的值。

int &a3 = getaa2(); //接引用方法2：此处它把该函数里面的a地址赋给a3,而该函数运行完毕后就清空释放内存，但内存门牌号还实实在在存在，所以a3指向这个区域的数据是乱码。

}

C++引用时的难点

当函数返回值为引用时，若返回栈变量，不能成为其他引用的初始值，不能作为左值使用。

若返回静态变量或者全局变量，可以成为其他引用的初始值，既可作为右值使用，也可作为左值使用。

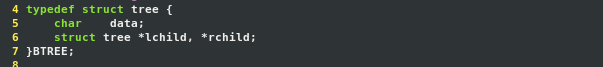
C++链式编程中，经常用到引用，运算符重载专题。

# C++进阶(传智播客扫地僧)

# C和C++与数据结构(传智播客扫地僧)

## 二叉树与递归

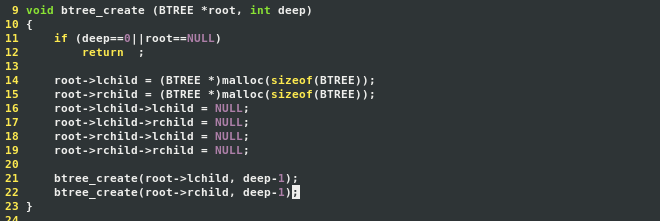
二叉树数据结构定义：摘自OS-大一运维



二叉树数据结构定义：总结：

数据结构的定义还是比较自由的，只要有左右指针域就可以了。

二叉树递归法创建：



二叉树递归法创建：总结：

其中重要的思想有递归以及二叉树的结构性质；递归的核心思想就是自己只干自己的事，处理上面传递进来的参数以及条件，如果无法完成任务就传给孩子去完成这项使命；二叉树的核心理论就是它的结构性质，也就是说当前节点的左右孩子的左右孩子也必须要初始化（NULL），也就是说当前节点的孙子都要指向空指针来表示结尾，否则永远找不到树顶，直到栈溢出也无济于事。传递经来的root是在主调用函数中已经初始化的，

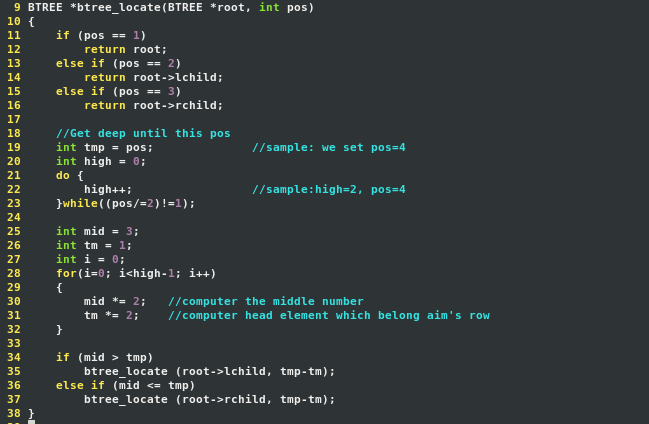
二叉树遍历：



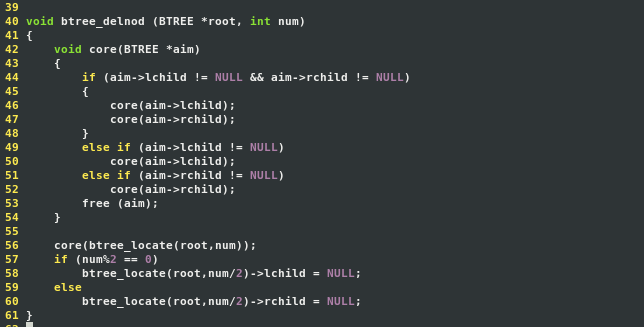
二叉树递归遍历：总结：

通过二叉树的结构图来辅助解析这段代码，你的思路会异常清晰。

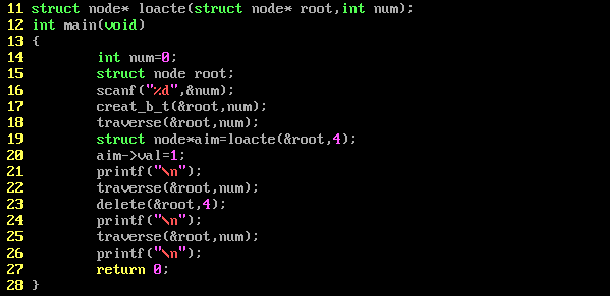
二叉树定位：



二叉树删除操作：



二叉树主函数初始化root:



二叉树深度和节点数的关系公式总结：

deep=01, node=01

deep=02, node=02

deep=03, node=04

deep=04, node=08

deep=05, node=16

最后总结出来的数学公式为：2^(deep-1)，即2的(deep-1)次方。

图的最优路径

图的最优路径都是根据前面的递归基础引申来来的，所以前面的递归和二叉树一定要熟练掌握它的核心思想，二分法查找也是递归的应用一种，二分法排序也是递归的应用。

# C和C++与设计模式基础(传智播客扫地僧)

# 第\*\*课-逆向工程基础

## 汇编语言的可移植性

汇编语言依赖处理器架构,不同处理器对应一种汇编语言,但大多汇编语言都有共性。

## 寄存器的位数演变

[= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =] EAX(32bit)

[= = = = = = = = = = = = = = = =] AX(16bi6)

[= = = = = = = =|= = = = = = = =]

---AH(8bit(H))-----AL(8bit(L))---

一些寄存器是别的寄存器的一部分：

例如，如果EAX保存了值0x12345678，这里是其他寄存器的值。

EAX: 12345678

AX: 12345678

AH: 12345678

AL: 12345678

## 通用寄存器

学习理论秘籍：用最有效，最简单，最快速的技巧：“记忆或者理解”。

1. EAX寄存器：EAX称为累加器，常用于算数运算、布尔操作、逻辑操作、返回函数结果等。
2. EBX寄存器：EBX称为基址寄存器，常用于存档内存地址。(Base)
3. ECX寄存器：ECX称为计数寄存器，常用于存放循环语句的循环次数，字符串操作中也常用。(Count)
4. EDX寄存器：称为数据寄存器，常常和EAX一起使用。(Data)

注意：上面所述的4个通用寄存器的专门用途不是一成不变的，编译器在编译程序的时候会根据很多因素，例如编译器、编译条件、操作系统等做出相应的改变，读者要知道着手研究的程序是用什么编译器编译，然后针对具体的编译器参考该编译器的说明。

## 变址寄存器

变址寄存器的含义是内存地址会变动的，在80386架构中有两个变址寄存器(ESI和EDI)

ESI: ESI称为源变址寄存器，通常存放将要处理的数据的内存地址。(Source)

EDI: EDI称为目的变址寄存器，通常存放处理后的数据的内存地址。(Destination)

## 指针寄存器

80386的寄存器有基址寄存器(EBP)，堆栈指针寄存器(ESP)，指令指针寄存器(EIP)。其中我们重视EBP和ESP，而EIP一般总是指向下一条要执行的指令的地址，一般情况无需修改EIP。

EBP:也可作为通用寄存器，也可访问堆栈里的数据。

ESP:不能作为通用寄存器，ESP存放当前堆栈栈顶地址。通常这两个寄存器我们来访问函数中的参数和局部变量。

## 标志寄存器

标志寄存器EFLAGS一共有32位，在这32位中大部分是保留和给编写操作系统的人用的，一般情况下只需知道32位的低16位中的8位即可，图1-3列出了标志寄存器EFLAGS中需要了解的8个位的位置。

\* OF（Overflow Flag）:溢出标志，溢出时为1，否则置0。

\* DF （Direction Flag）:方向标志，在串处理指令中控制信息的方向。

\* IF (Interrupt Flag) :中断标志

\* AF (Auxiliary carry Flag) :辅助进位标志，有进位时置1，否则置0。

\* ZF (Zero Flag) :零标志，运算结构为0时ZF位位置1，否则置0。

\* SF (Sign Flag):符号标志，结果为负时置1，否则置0。

\* CF (Carry Flag): 进位标志，进位时置1，否则置0。

\* PF (Parity Flag): 奇偶标志。结果操作数中1的个数为偶数时置1，否则置0。

EFLAGS寄存器的用途

正如上面所说EFLAGS是实现条件判断和逻辑判断的一种机制，在汇编语言中一般不直接访问EFLAGS寄存器，而是通过指令的操作隐含访问EFLAGS寄存器，下面是一个利用EFLAGS寄存器的例子。

Cmp dword ptr [ebp+8],0 ;影响标志CF、ZF、SF、OF、AF和PF

Jz 00405898 ; 如果ZF等于1，则跳转到00405898

## 寻址方式

1. 立即寻址

示例：

Mov eax,56H

作用：通常用于赋值。

1. 直接寻址

示例：

Mov eax,[12558878H]

作用：通常用于处理变量。

1. 寄存器寻址

示例

Mov eax,[edi]

作用：地址在寄存器中。

（4）寄存器相对寻址

示例

Mov EAX,[EDI+32H]

作用：常用于访问数组和结构。

1. 基址加变址寻址

示例

Mov EAX,[EBP+ESI]

作用：常用于访问数组

1. 相对基址加变址寻址。

示例

MOV EAX,[EBX+EDI-10H]

作用：常用于访问结构。

## 高级语言中的数据结构和80386寻址方式的关系

|  |  |  |  |
| --- | --- | --- | --- |
| 存储类型 | 结构类型 | 寻址方式 | 范例 |
| 静态分配类型  全局结构类型 | 变量 | 常量 | DS:[58FF58] |
| 数组 | 变址+常量 | DS:[ESI+58FF58] |
| 记录 | 常量 | DS:[58FF58] |
| 记录的数组 | 变址+常量 | DS:[ESI+58FF58] |
| 数组的记录 | 变址+常量 | DS:[ESI+58FF58] |
| 栈分配类型  局部结构类型 | 变量 | 基地址+常量 | SS:[ESP+100] |
| 数组 | 基地址+变址+常量 | SS:[ESP+ESI\*2+100] |
| 记录 | 基地址+常量 | SS:[ESP+100] |
| 记录的数组 | 基地址+变址+常量 | SS:[ESP+ESI\*2+100] |
| 数组的记录 | 基地址+变址+常量 | SS:[ESP+ESI\*2+100] |
| 堆分配类型  动态分配结构类型 | 变量 | 基地址 | DS:[EBX] |
| 数组 | 基地址+变址 | DS:[EBX+ESI\*2] |
| 记录 | 基地址+常量 | DS:[EAX+50] |
| 记录的数组 | 基地址+变址+常量 | DS:[EBX+ESI\*2+50] |
| 数组的记录 | 基地址+变址+常量 | DS:[EBX+ESI\*2+50] |

## 汇编指令格式

80386指令

观察一下Intel 的80x86 CPU文档手册会发现，Intel的80x86 CPU支持几百条指令，如此多的指令莫说深入研究，连入门恐怕也非易事。

幸运的是，在这几百条的指令中，常用的也不过几十条而已，Intel 的80x86 CPU之所以支持几百条指令，原因在于Intel 的80x86 CPU为了保持向下兼容的问题，所以从过去到现在的所有指令都包含在CPU里面，例如有8086/8088、80186、80286和80386等。

Intel格式和AT&T格式（&在英语里读and）

编写IA-32架构的汇编语言常见有两种格式，一种是Intel格式，另一种是AT&T格式。

1. Intel 格式的指令格式。

指令名称 目标操作数DST，源操作数SRC

示例代码：

Mov eax,[edx] //将内存地址为EDX的数据放入EAX寄存器

Xchg eax,edi //交换EAX和EDI寄存器的值

Add eax,ebx // 将EAX和EBX相加，结构放回EAX中

Shl eax,4 //将EAX逻辑左移4位。

1. AT&T格式的指令格式。

指令名称 源操作数SRC，目标操作数DST

示例代码：

Mov (%EDX),%EAX //将内存地址为EDX的数据放入EAX寄存器

XCHG %EDI,%EAX //交换EAX和EDI寄存器的值

ADD %EBX,%EAX //将EAX和EBX相加，结构放回EAX中

SHL $4,%EAX //将EAX逻辑左移4位。

本教程中讲解的汇编语言统一采用Intel 格式的汇编格式。

## 数据传送指令

80386的数据传送指令是为了实现CPU和内存、输入/输出端口之间的数据传送。

1. MOV:称为数值传送指令，格式是“MOV DST,SRC”。
2. MOV指令将源操作数SRC传送到目的操作数DST中，

传送的数据格式可以是8字节、16字节和32字节。

示例代码：

MOV EAX,56 //将56H立即数传送到EAX寄存器

MOV ESI,DWROD PTR [EAX\*2+1] //将内存地址为EAX\*2+1处的4字节数据传送到ESI寄存器。

MOV AH,BYTE PTR [ESI\*2+EAX] //将内存地址为ESI\*2+EAX处的8位数据传送到AH寄存器。

MOV DWORD PTR [ESP+36],EBX //将EBX寄存器的值以4字节传送到堆栈地址为ESP+36所指向的地方。

1. XCHG:称为交换指令，XCHG实现寄存器间和内存间的数据交换。

格式是“XCHG DST,SRC”。XCHG指令交换SRC和DST之间的数据，

交换的数据可以是8字节、16字节和32字节，其中SRC和DST必须格式相同。

示例代码：

XCHG EAX,EDX //将EDX寄存器的值和EAX寄存器的值交换

XCHG [ESP-55],EDI //将EDI寄存器的值和堆栈地址为[esp-55]处的值交换。

XCHG BH,BL //将BL寄存器和BH寄存器的值交换。

## 堆栈操作指令

PUSH和POP:称为压入堆栈指令和弹出堆栈指令，格式是“PUSH SRC(源操作数)”和“POP DST（目的操作数）”。

PUSH指令和POP指令是匹配出现的，上面的代码有多少个PUSH下面的代码就有多少个POP，否则堆栈就会不平衡。

windows32位采用满栈的栈操作。

PUSH指令将源操作数SRC压入堆栈，同时ESP-4，而POP恰恰相反，POP指令从堆栈的顶部弹出4字节的数值然后放入DST。在32位的Windows操作系统上，PUSH和POP指令的操作是以4字节为单位的。

PUSH和POP指令常用于向函数传递参数。

示例代码：

PUSH EAX //将EAX寄存器的值以4字节压入堆栈，同时ESP-4

PUSH DWORD PTR [12FF8589] //将内存地址为12FF8589所指向的值以4字节压入堆栈，同时ESP-4

-------------------------

POP DWORD PTR [12FF8589] //将堆栈顶部的4字节弹出到内存地址为12FF8589所指地方，同时ESP+4

POP EAX //将堆栈顶部的4字节弹出到EAX寄存器，同时ESP+4

## 地址传送指令

80x86有3条地址传送指令，分别是LEA，LDS和LES。其实LDS和LES指令和段寄存器有关，在32位的Windows操作系统上，一般的程序员都不需要管理段寄存器，所以相对而言，LDS和LES寄存器使用得比较少，一般情况下常见的只有LEA指令。

LEA：称为地址传送指令，格式是“LEA DST,ADDR”。LEA指令将ADDR地址加载到DST，其中ADDR可以是内存，也可以是寄存器，而DST必须是一个通用寄存器。

LEA指令相当于C语言中的“&”and操作符，需要注意的是LEA和MOV是不同的，前者LEA传送的是地址，后者MOV传送的是操作数。

示例代码：

LEA EAX,[12345678]

MOV EAX,[12345678]

//指令执行后EAX寄存器的值为12345678H，而MOV EAX,[12345678] 指令执行后EAX寄存器的值为内存地址12345678指向的那个数值。

还有一点需要注意LEA指令可用于算法运算。

示例代码：

LEA ECX,[ECX+EAX\*4] //ECX=ECX+EAX\*4

## 算数运算指令

80x86提供了8条加减法指令：ADD、ADC、SUB、SBB、INC、DEC、CMP和NEG，4条乘除法指令：MUL、IMUL、DIV和IDIV。

1. 常用的加减法指令。

1.ADD：称为加法指令，格式是“add OPER1,OPER2”。（操作数（operand)）

ADD指令将OPER1+OPER2结果存放在OPER1中。

示例代码：

ADD EAX,ESI //将EAX寄存器的值加上ESI寄存器的值，得出的结果保存在EAX寄存器中。

ADD EBX,DWORD PTR [12345678] //将EBX寄存器的值加上内存地址为12345678所指的4字节值，得出的结构保存在EBX寄存器中，其中DWORD PTR的意思是显示说明按多少字节来操作，DWORD是DOUBLE WORD的缩写，也就是两个WORD的意思。

不同的平台和编译器中，DWORD占用的字节数不同，在32位的Windows中一个WORD占用16字节空间，DWORD占用32字节空间，读者可以在32位的Windows平台上使用Visual C++编译器编写C语言Printf("%d",sizeof(DWORD))来验证。

在汇编语言中常用的还有WORD PTR和BYTE PTR，表示的意思分别是按WORD来操作和按BYTE来操作。

1. SUB:称为减法指令，格式是“SUB OPER1,OPER2”。

SUB 指令将OPER1-OPER2结果存放在OPER1中。

示例代码：

SUB ECX,4H //将ECX寄存器的值减去4H，得出的结果保存在EAX寄存器中。

SUB BYTE PTR[EAX],CH //将内存地址为EAX所指向的数据按字节为单位和CH寄存器相减，得出的结果按字节为单位保存在EAX所指向的地方。

1. MUL:称为乘法指令，格式是“MUL OPER1,OPER2”。

MUL指令将OPER1\*OPER2结果存放在OPER1中。

1. DIV:称为除法指令，格式是“DIV OPER1,OPER2”。

DIV指令将OPER1/OPER2结果存放在OPER1中。

## 自加自减指令

INC:称为加1指令，格式是“INC OPER”。

INC指令将操作数OPER加1，得出的结果保存在OPER中。

示例代码：

INC EAX //将EAX寄存器的值加1，得出的结果存放在原来的地方。

INC WORD PTR [EBX+2] //将内存地址为EBX+2的数据按WORD为单位加1，得出的结果存放在原来的地方。

DEC:称为减1指令，格式是“EDC OPER”。

DEC指令将操作数OPER减1，得出的结果保存在OPER中。

示例代码：

DEC EDX //将EDX寄存器的值减1，得出的结果存放在原来的地方

DEC DWORD PTR [EBP+36] //将堆栈地址为EBP+36的数据按DWORD为单位减1，得出的结果存放在原来的地方。

## 除法指令

DIV：称为除法指令，格式是“DIV OPER”。

DIV 指令将64位（EDX和EAX）或32位（EAX）的值除以OPER，得出的商保存在EAX寄存器中，而余数则保存在EDX寄存器中，由OPER操作数决定按多少字节操作。

示例代码：

DIV ECX //将EAX寄存器的值按4字节为单位除以ECX寄存器的值，得出的结果商保存在EAX寄存器中，余数保存在EDX寄存器中。

DIV WORD PTR [ESP+36] //将EAX寄存器的值按WORD为单位除以堆栈地址为ESP+36所指向的数据，得出的结果商保存在EAX寄存器中，余数保存在EDX寄存器中。

IDIV：称为有符号除法指令，原理和操作可以参考DIV指令，IDIV和DIV的区别是IDIV将参与运算的操作数当成有符号数来处理。

mov eax,1000  
cwd  
div ecx

1000(H)/7= 4096/7 =585.1

CWD是汇编语言中的字扩展指令，它的功能是将一个字型变量扩展为双字型变量，即***C***hange ***W***ord to ***D***ouble word。

## 逻辑运算指令

80x86提供了OR、AND、NOT/XOR和TEST这5条逻辑运算指令，逻辑运算指令是数据加密/解密的基础，所以应该熟练掌握它们的用法。

1. OR:称为或操作指令，格式是“OR OPER1,OPER2”。OR指令将OPER1操作数和OPER2操作数进行或运算，得出的结果保存在OPER1中。

OR指令主要用于维持某个二进制的某些位的值不变，而另一些位设置为1的情况。把不需要改变的位用0进行或运算，把要设置为1的位用1进行或运算即可。

示例代码：

OR EAX,80008000H //将EAX寄存器和立即数80008000H进行或运算，实际上是将EAX寄存器的31位和15位置1

OR AH,BH //将AH寄存器或BH寄存器进行或运算。

EAX= 1C1EA4

OR EAX,80008000H

得到：eax=801C9EA4

AH=1 BH=2

1 二进制 0001

Or 2 二进制 0010

0011(也就是10进制3)

因为第一位是1，第二位是0，而第二个数的第一位是0第二位是1

根据or计算原理.1 or 0=1,0 or 0=0,0 or 1=1,1 or 1=1 的原理得到的就是0011

1. AND：称为与操作指令，格式是“AND OPER1,OPER2”。AND指令将OPER1操作数和OPER2操作数进行与运算，得出的结果保存在OPER1中。

AND指令主要用于维持某个二进制数的某些位的值不变，而另一些位设置为0的情况。把不需要改变的位用1进行与运算，把要设置为0的位用0进行与运算即可。

示例代码：

AND CH,80H //将CH寄存器的值和80H进行与运算，实际上是将CH寄存器的第7位保存不变，其余位置0.

AND DWORD PTR [EAX],80008000H //将内存地址为EAX所指向的数据按DWORD为单位与80008000H进行与运算，实际上是将内存地址为EAX指向的4字节数据的第31位和15位置保存不变，其余位置0.

AND CH,80H

CH=1 二进制 00000001

80 二进制 10000000

And 00000000

根据and计算原理 0 and 1 =0 ,0 and 0=0,1 and 1=1的原理得到的就是00000000

AND DWORD PTR [EAX],80008000H

EAX=001C1EA4

001C1EC8 二进制\_00000000000111000001111011001000

080008000 二进制10000000000000001000000000000000

And 00000000000000000000000000000000 (十六进制就是00000000)

根据and计算原理 0 and 1 =0 ,0 and 0=0,1 and 1=1的原理得到的就是00000000000000000000000000000000

1. NOT：称为取反指令，格式是“NOT OPER”。NOT 指令将OPER操作数取反。注意NOT和NEG不同，NOT指令是按位取反，NEG是求补，意即将0减去操作数。

例如：

15H的二进制为 0 0 0 1 0 1 0 1 B

NEG 15H 的二进制为 1 1 1 0 1 0 1 1 B

NOT 15H 的二进制为 1 1 1 0 1 0 1 0 B

根据NOT计算原理 0 not 1, 1 not 0 得 1 1 1 0 1 0 1 0

NOT ECX // ECX=1 执行后就是用FFFFFFFF-1=FFFFFFFE

1. XOR：称为异或操作指令，格式是“XOR OPER1,OPER2”。XOR指令将OPER1操作数和OPER2操作数进行异或运算，得出的结果保存在OPER1中。

XOR指令主要用于维持某个二进制数的某些位的值不变，而某些位取反的情况。把不需要改变的位用0进行异或运算，把需要取反的位用1进行异或运算即可。

示例代码：

XOR EAX,FFFF0000H //将EAX寄存器的值和立即数FFFF0000H进行异或运算，实际上将EAX寄存器的值的高16位取反，低16位保存不变。

XOR AH,F0H //将AH寄存器的值和立即数F0F0H进行异或运算，实际上是将AH寄存器的值的第15位和7位取反，其余位保持不变。

EAX=001C1EA4

001C1EA4\_0000000000111000001111010100100

FFFF0000 \_11111111111111110000000000000000

Xor \_11111111111000110001111010100100 (十六进制:FFE31EA4)

根据 XOR 计算原理：0 xor 0 =0, 1 xor 1 =0,1 xor 0 =1 得到：FFE31EA4

XOR AH,F0H

AH=50\_01010000

F0\_11110000

XOR 10100000 （A0十六进制）

根据 XOR 计算原理：0 xor 0 =0, 1 xor 1 =0,1 xor 0 =1 得到：A0

1. TEST：称为测试指令，格式是“TEST OPER1,OPER2”。TEST指令将OPER1操作数和OPER2操作数进行与运算，不保存结果，只设置标志寄存器EFLAGS相应的标志位的值。

TEST指令常用于测试一个二进制位的某些位是否为1，但不改变源操作数的情况。

示例代码：

TEST EAX,F0000000H //将EAX寄存器的值和立即数F00000000H进行与运算，实际上是测试EAX寄存器的第31、30、29、28位是否为1，并且设置标志寄存器EFLAGS相应的标志位的值。

TEST EAX,F0000000H

EAX=001C1EA4

001C1EA4\_ 000000000000000111000001111010100100

F00000000\_111100000000000000000000000000000000

And 000000000000000000000000000000000000

根据and计算原理 0 and 1 =0 ,0 and 0=0,1 and 1=1的原理得到的就是 0

## 普通移位指令

80x86有4条普通移位指令和4条循环移位指令，它们都隐含地使用CF寄存器参与运算。

1. 普通移位指令。

SAL算术左移指令、SAR算术右移指令、SHL逻辑左移指令和SHR逻辑右移指令。

这4条普通移位指令的格式都是一样的：普通移位指令名称 OPER1,OPER2，其中OPER1可以是寄存器或内存，OPER2代表的是移位的位数。其中SAL指令和SHL指令指向结果是一样的。

对于有符号和无符号数而言，SAL算术左移指令和SHL逻辑左移指令每移动一位相当于乘以2.而SAR算术右移指令和SHR逻辑右移指令有点不同。对于有符号和无符号而言，SAR算术右移指令每移动一位相当于除以2，而SHR逻辑右移指令不管操作数是有符号数还是无符号数，每向右移动一位，左边都是用0填充，所以当操作数是无符号数的时候，SHR逻辑右移指令每移动一位才等于除以2.

示例代码：

SAL EAX,2 //将EAX寄存器的值向左移动2位，得出的结果保存在EAX寄存器中，相当于EAX=EAX\*4

SAR DWORD PTR DS:[ESI],4 //将内存地址为ESI所指向的数据按DWORD为单位右移4位，相当于将内存地址为ESI所指向的数据按DWORD为单位的数据除以2\*2\*2\*2=16(十进制)

SHL DWORD PTR [EBP+2H],2 //将堆栈地址为EBP+2H所指向的数据按DWORD为单位左移2位，相当于将内存地址为EBP+2H所指向的数据按DWORD为单位的数据乘以4.

SHR EDI,4 //将EDI寄存器的值逻辑右移ECX位。EDI值除以2\*2\*2\*2=16（十进制）

## 循环移位指令

循环移位指令

ROL左循环移位指令、ROR右循环移位指令、RCL带进位左循环移位指令和RCR带进位右循环移位指令。

这4条循环移位指令的格式都是一样的：循环移位指令名称 OPER1,OPER2，其中OPER1可以是寄存器或内存，OPER2要么是CL寄存器要么是1，代表移动的次数，如果要移的次数多于1次，则需要把移位次数存放在CL寄存器中。

ROL、ROR和RCL、RCR的区别是前者没有将标志寄存器EFLAGS的CF进位标志包含参与循环移位，后者则把CF进位标志包含参与循环移位。

示例代码：

ROL AL,1 //将EAX寄存器的值向左移动一位，被移出的位送到CF，同时将被移出的位放到最低位。

ROR EAX,CL //将EAX寄存器的值向右移动CL位，被移出的位送到CF，同时将被移出的位放到最高位。

RCL EAX,1 //将EAX寄存器的值向左移动1位，被移出位送到CF，同时将CF之前的值放到最低位。

RCR EAX,CL //将EAX寄存器的值向右移动CL位，被移出位送到CF，同时将CF之前的值放到最高位。

ROL AL,1

ROR EAX,CL

把82H转成二进制数10000010(B)

循环左移1位后变成：00000101(B)，换算成十六进制数便是05(H)

循环右移1位后变成：01000001(B)，换算成十六进制数便是41(H)

RCL AL,1

RCR AL,CL

首先把82H转换成二进制数10000010B

带进位循环左移1位后变成：00000100B，CF=1换算成十六进制数便是04H

带进位循环右移1位后变成：11000001B，CF=0换算成十六进制数便是C1H

## 条件转移指令

条件转移指令

程序的结构可以分为3大部分，分别是顺序结构、分支结构、循环结构。像高级语言一样，在高级语言里有if-else条件分支、do-while循环、for 循环和goto语句来改变程序的执行流程。在汇编语言中没有高级语言里的if-else条件分支、do-while循环和for循环，汇编语言通过提供条件转移指令来实现程序执行流程的改变。

在汇编语言中条件转移指令分为无条件转移指令和条件转移指令。

1. 无条件转移指令JMP

JMP指令格式是：JMP OPER,其中OPER是目的地址。

示例代码：

JMP EAX //跳转到EAX寄存器指示的4字节地址。

JMP DWORD PTR DS:[ESI+2] //跳转到内存地址为ESI+2指示的2字节地址。

1. 条件转移指令。

汇编的条件转移指令非常多，通常可以分为3大部分：无符号数的条件转移指令、有符号数的条件转移指令和算术条件转移指令。

条件转移指令格式是：条件转移指令名称 OPER，其中OPER是目的地址。

无符号数的条件转移指令名称、转移条件、转移说明如表1-2表示。

表1-2 无符号数的条件转移说明表

|  |  |  |
| --- | --- | --- |
| 指令名称 | 转移条件 | 转移说明 |
| JA/JNBE | CF=0 且 ZF=0 | 结果低于等于转移，或者高于转移 |
| JAE/JNB | CF=0 | 结果不低于转移，或者高于转移 |
| JB/JNAE | CF=1 | 结果低于转移，或者不高于等于转移 |
| JBE/JNA | CF=1 | 结果低于等于转移，或者不高于转移 |

有符号数的条件转移指令名称、转移条件、转移说明如表1-3所示。

表1-3 有符号数的条件转移说明表

|  |  |  |
| --- | --- | --- |
| 指令名称 | 转移条件 | 转移说明 |
| JG/JNLE | ZF=0 且SF=OF | 不小于等于转移，或者大于转移 |
| JGE/JNL | SF=OF | 不小于转移，或者大于等于转移 |
| JL/JNGE | SF≠OF | 小于转移，或者大于等于转移 |
| JLE/JNG | ZF=1 且 F≠OF | 小于等于转移，或者不大于转移 |

算术条件转移指令名称、转移条件、转移说明如表1-4所示。

表1-4 算术条件转移说明表

|  |  |  |
| --- | --- | --- |
| 指令名称 | 转移条件 | 转移说明 |
| JZ/JE | ZF=1 | 等于0转移，或者相等转移 |
| JNZ/JNE | ZF=0 | 不等于0转移，或者不相等转移 |
| JS | SF=1 | 为负转移 |
| JNS | SF=0 | 为正则转移 |
| JO | OF=1 | 溢出转移 |
| JNO | OF=0 | 不溢出转移 |
| JC | CF=1 | 进位标志被置转移 |
| JNC | CF=0 | 进位标志被清转移 |
| JP/JPE | PF=1 | 偶数转移 |
| JNP/JPO | PF=0 | 奇数转移 |

## 函数调用相关指令

函数调用指令CALL

在高级语言中，对函数再熟悉不过了，一个程序的功能可以认为是一组函数互相调用的结果，函数是计算并且返回某些值的一段代码。

在汇编语言中，

使用CALL指令和RET指令或者CALL指令和ADD ESP,OPER指令

实现函数的调用与函数返回。

CALL指令的格式是：CALL OPER，其中OPER是函数地址（子程序地址）

CALL指令首先将ESP堆栈指针寄存器的值减4，然后将EIP程序指令计数器的值压入堆栈，最后计算函数地址，将当前EIP程序指令计数器的值设置为函数地址。

例如：调用下面的C语言函数，计算机会执行如下操作：

PUSH a[i]

Print("%d",a[i]) : PUSH OFFSET String "%d"

CALL print

Add esp,8

在高级语言中调用函数时不需要编程人员管理堆栈和恢复函数调用前的环境，因为高级语言的编译器在编译源代码的时候已经做好了。

在汇编语言中，编程人员使用CALL调用完函数后需要使用RET指令或ADD ESP指令恢复函数调用前的坏境，以便调用完函数后程序能继续正常执行。

RET函数返回指令的格式是：RET OPER，其中OPER是需要从堆栈中弹出的字节数。

RET指令首先将堆栈中弹出4字节数据到EIP，然后ESP=ESP+2，最后根据OPER的值修改ESP堆栈指针的值ESP=ESP+OPER。

对于遵从Cdecl调用约定（后面会讲到）的那些函数而言，函数调用完后不是使用RET指令，而是由调用者使用ADD ESP,OPER从堆栈中弹出OPER字节数据来清理堆栈。

如何判断CALL有多少个参数？

1.看CALL前面的有几个PUSH，假设有3个，就是有3个参数

2.进入CALL，看CALL尾部retn指令 是几，假设是：retn 4 说明该CALL有1个参数，假设retn 0c 说明有3个参数，0c(十六进制)=12(10进制)，之所以4为一个参数，是因为1个PUSH 占用4字节。

如果平衡堆栈？

根据上面的方法可以判断出CALL有几个参数，

平衡堆栈就是用add esp,参数个数\*4再转换成16进制

假设：有5个参数，5\*4=20(十进制) =转换成十六进制（14） 也就是： add esp,14

Push 1

Push 2

Push 3

Push 4

Push 5

Call 00401000

Add esp,14

用于：CALL调用后能达到效果，但是游戏奔溃。

## 汇编函数3种常用的函数调用约定

在早期的编程语言中具有代表性的有C语言、Pascal、Basic、不辛的是这些具有代表性的编程语言各自使用不同的函数参数传递方式和由谁负责清除传递参数时使用的堆栈，俗称堆栈平衡方式，从而导致了有几种调用约定。

调用约定是规定函数参数传递的顺序和堆栈平衡的方式。常见的函数调用约定有Pascal调用约定、Cdecl调用约定和StdCall调用约定。

使用Delphi、编写的程序都遵从Pascal调用约定，C/C++/JAVA编写的程序都遵从Cdecl调用约定，Windows的API调用遵从的是StdCall调用约定。

调用约定的参数传递顺序

假设有一高级语言函数Message(parameter1,paramerer2,parameter3)，按照Pascal调用约定、Cdecl调用约定和StdCall调用约定的参数传递顺序如表1-5所示。

表1-5 常见的函数调用约定

|  |  |  |
| --- | --- | --- |
| Pascal调用约定 | Cdecl调用约定 | StdCall调用约定 |
| PUSH parameter1 | PUSH parameter3 | PUSH parameter3 |
| PUSH parameter2 | PUSH parameter2 | PUSH parameter2 |
| PUSH parameter3 | PUSH parameter1 | PUSH parameter1 |
| CALL Message | CALL Message | CALL Message |
|  | ADD ESP,0CH |  |
| 参数从左到右传递，由被调用的函数清理堆栈 | 参数从右到左传递，调用函数清理堆栈 | 参数从右到左传递，由被调用的函数清理堆栈。 |

## 汇编指令英文全称

汇编指令英文全称  
  
**1.通用数据传送指令**  
MOV----> move  
MOV dest,src ;dest←src  
MOV指令把一个字节或字的操作数从源地址src传送至目的地址dest。  
MOVSX---->extended move with sign data  
MOVZX---->extended move with zero data  
PUSH---->push  
POP---->pop

**进栈出栈指令**  
PUSHA---->push all  
POPA---->pop all  
PUSHAD---->push all data  
POPAD---->pop all data  
BSWAP---->byte swap  
XCHG---->exchange  
交换指令用来将源操作数和目的操作数内容交换，操作数可以是字、也可以是字节，可以在通用寄  
  
存器与通用寄存器或存储器之间对换数据，但不能在存储器与存储器之间对换数据。  
mov ax,1234h ;ax=1234h  
mov bx,5678h ;bx=5678h  
xchg ax,bx ;ax=5678h，bx=1234h  
xchg ah,al ;ax=7856h  
CMPXCHG---->compare and change  
XADD---->exchange and add  
XLAT---->translate  
换码指令用于将BX指定的缓冲区中、AL指定的位移处的数据取出赋给AL。

**2.输入输出端口传送指令**  
IN---->input  
OUT---->output  
3.目的地址传送指令  
LEA---->load effective addres

**有效地址传送指令**  
mov bx,0400h  
mov si,3ch  
lea bx,[bx+si+0f62h] ;BX=139EH  
这里BX得到的是主存单元的有效地址，不是物理地址，也不是该单元的内容。  
LDS---->load DS  
LES---->load ES  
LFS---->load FS  
LGS---->load GS  
LSS---->load SS

**4.标志传送指令**  
LAHF---->load AH from flag  
SAHF---->save AH to flag  
PUSHF---->push flag  
POPF---->pop flag  
PUSHD---->push dflag  
POPD---->pop dflag

**二、算术运算指令**  
ADD---->add  
加法指令mov al,0fbh ;al=0fbh  
add al,07h ;al=02h  
ADC---->add with carry  
INC---->increase 1  
AAA---->ascii add with adjust  
DAA---->decimal add with adjust  
SUB---->substract  
SBB---->substract with borrow  
DEC---->decrease 1  
NEC---->negative  
CMP---->compare  
AAS---->ascii adjust on substract  
DAS---->decimal adjust on substract  
MUL---->multiplication  
IMUL---->integer multiplication  
AAM---->ascii adjust on multiplication  
DIV---->divide  
IDIV---->integer divide  
AAD---->ascii adjust on divide  
CBW---->change byte to word  
CWD---->change word to double word  
CWDE---->change word to double word with sign to EAX  
CDQ---->change double word to quadrate word  
  
**三、逻辑运算指令**  
AND---->and  
or---->or  
XOR---->xor  
NOT---->not  
TEST---->test  
SHL---->shift left  
SAL---->arithmatic shift left  
SHR---->shift right  
SAR---->arithmatic shift right  
ROL---->rotate left  
ROR---->rotate right  
RCL---->rotate left with carry  
RCR---->rotate right with carry

**四、串指令**  
MOVS---->move string  
CMPS---->compare string  
SCAS---->scan string  
LODS---->load string  
STOS---->store string  
REP---->repeat  
REPE---->repeat when equal  
REPZ---->repeat when zero flag  
REPNE---->repeat when not equal  
REPNZ---->repeat when zero flag  
REPC---->repeat when carry flag  
REPNC---->repeat when not carry flag

**五、程序转移指令**  
1>无条件转移指令(长转移)  
JMP---->jump  
CALL---->call  
RET---->return  
RETF---->return far  
2>条件转移指令(短转移,-128到+127的距离内)  
JA--->jump when above

JAE---->jump when above or equal  
JNB---->jump when not below  
JB---->jump when below  
JNAE---->jump when not above or equal  
JBE---->jump when below or equal  
JNA---->jump when not above  
JG---->jump when greater  
JNLE---->jump when not less or equal  
JGE---->jump when greater or equal  
JNL---->jump when not less  
JL---->jump when less  
JNGE---->jump when not greater or equal  
JLE---->jump when less or equal  
JNG---->jump when not greater  
JE---->jump when equal  
JZ---->jump when has zero flag  
JNE---->jump when not equal  
JNZ---->jump when not has zero flag  
JC---->jump when has carry flag  
JNC---->jump when not has carry flag  
JNO---->jump when not has overflow flag  
JNP---->jump when not has parity flag  
JPO---->jump when parity flag is odd  
JNS---->jump when not has sign flag  
JO---->jump when has overflow flag  
JP---->jump when has parity flag  
JPE---->jump when parity flag is even  
JS---->jump when has sign flag

**3>循环控制指令(短转移)**  
LOOP---->loop  
LOOPE---->loop equal  
LOOPZ---->loop zero  
LOOPNE---->loop not equal  
LOOPNZ---->loop not zero  
JCXZ---->jump when CX is zero  
JECXZ---->jump when ECX is zero

**4>中断指令**  
INT---->interrupt  
INTO---->overflow interrupt  
IRET---->interrupt return

**5>处理器控制指令**  
HLT---->halt  
WAIT---->wait  
ESC---->escape  
LOCK---->lock  
NOP---->no operation  
STC---->set carry  
CLC---->clear carry  
CMC---->carry make change  
STD---->set direction  
CLD---->clear direction  
STI---->set interrupt  
CLI---->clear interrupt

**六、伪指令**  
DW---->definw word  
PROC---->procedure  
ENDP---->end of procedure  
SEGMENT---->segment  
ASSUME---->assume  
ENDS---->end segment  
END---->end

# 第\*\*课-英语基础(单词量)

不是为了考四级,而是供我所需~

## A

a [ei,ə] art.一(个) 每一(个)

abandon [ə'bændən] vt. 丢弃 放弃 抛弃 n. 放纵

ability [ə'biliti] n. 能力 能耐 本领

able ['eibl] adj. 能干的 有能力的 出色的

abnormal [æb'nɔ:məl] adj. 反常的 不正常的 不规则的

aboard [ə'bɔ:d] adv. 在船(车)上 在火车上 在飞机上

about ['əbaut] prep. 关于 在...周围 adj. 准备 adv. 大约

above [ə'bʌv] prep. 在...上面 超过 高于

abroad [ə'brɔ:d] adv. (在)国外 海外(一般作表语)

absence ['æbsəns] n. 缺席 不在场 缺乏

absent ['æbsənt] adj. 缺席的 不在场的 缺乏的

absolute ['æbsəlu:t] adj. 绝对的 纯粹的

absolutely ['æbsəlu:tli] adv. 独立地 完全地 绝对地

absorb [əb'sɔ:b] vt. 吸收 吸纳 吸引...的注意 吞并 使专心

abstract ['æbstrækt] adj. 抽象的 理论的

abundant [ə'bʌndənt] adj.丰富的 充裕的 大量的

abuse [ə'bju:z] vt. 滥用 辱骂 虐待 n. 滥用 恶习

academic [.ækə'demik] adj. 学院的 理论的

academy [ə'kædəmi] n. 私立中学 专科院校 学院 学术 学会

accelerate [æk'seləreit] vt. (使)加快 促进 加速 提前 跳级

acceleration [æk.selə'reiʃən] n. 加速 促进 加速度

accent ['æksənt] n. 口音 腔调 重音 强调 vt. 重读 强调

accept [ək'sept] vt. 接受 同意 承担(责任等) vi. 接受

acceptable [ək'septəbl] adj. 可接受的 合意的 受欢迎的

acceptance [ək'septəns] n. 接受（礼物、邀请、建议等）

access ['ækses] n. 接近(的机会)通道 入口 vt. (电脑)存取 进入

accessory [æk'sesəri] n. 同谋 从犯 附件 adj. 附属的(副的、辅助的)

accident ['æksidənt] n. 意外的 事故

accidental [.æksi'dentl] a. 意外的 偶然的

accommodate [ə'kɔmədeit] vt. 容纳 供应 供给

accommodation [ə.kɔmə'deiʃən] n. 招待设备

accompany [ə'kʌmpəni] vt. 陪伴 陪同 伴随 给...伴奏

accomplish [ə'kɔmpliʃ] vt. 达到(目的) 完成

accord [ə'kɔ:d] vt. 使一致 调解 赠予 给予

accordance [ə'kɔ:dəns] n. 一致 和谐 授予 符合 同意

accordingly [ə'kɔ:diŋli] adv. 因此 所以 于是 相应地 照着

account [ə'kaunt] n. 记述 解释 根据 理由 帐目 报告 估计

accumulate [ə'kju:mjuleit] vt. 积累 累加 堆积

accuracy ['ækjurəsi] n. 准确(性) 准确度

accurate ['ækjurit] adj. 准确的 正确无误的

accuse [ə'kju:z] vt. 指责 责备 控告 归咎于

accustom [ə'kʌstəm] vt. 使...习惯

accustomed [ə'kʌstəmd] adj. 惯常的 习惯的

ache [eik] vi. 痛 想念 渴望 n. 疼痛

achieve[ə'tʃi:v] vt. 完成 实现 达到 vi. 达到目的

achievement [ə'tʃi:vmənt] n. 完成 成就 成绩 达到

acid ['æsid] n. 酸 酸味物质 adj .酸的 酸性的

acquaintance [ə'kweintəns] n. 认识 了解 熟人

acquire [ə'kwaiə] vt. 占有 取得 获得 学到

acre ['eikə] n. 英亩(=6.07亩)

across[ə'krɔs] prep. 横过 穿过 在...对面

act [ækt] vi. 行动 表演 表现 充当 见效

action ['ækʃən] n. 起诉 行动 作用 功能 情节 活动

active ['æktiv] adj. 活跃的 积极的 主动的

activity [æk'tiviti] n. 活动 活跃 活力 行动

actor['æktə] n. 男演员 演剧的人

actress ['æktris] n. 女演员

actual ['æktjuəl] adj. 实际的 真实的 现行的

actually ['æktʃuəli] adv. 实际上 事实上 居然 竟然（表示惊讶）

acute [ə'kju:t] adj. 尖的 锐的 敏锐的 敏锐的 激烈的

ad [æd] n. (缩)广告( advertisement[əd'və:tismənt] 的缩写)广告

adapt [ə'dæpt] vt. 使适应 改编 vi. 适应 适合

add [æd] vt. 添加 附加 掺加 增加

addition[ə'diʃən]n. 加 加法 附加物 增加

additional [ə'diʃənl] adj. 附加的 追加的 另外的

address [ə'dres] n. 地址 住址

adequate['ædikwit] adj. 足够的 充足的 适当的 可以胜任的

adjective ['ædʒiktiv] n. 形容词 adj. 形容词的 不独立的

adjust [ə'dʒʌst] vt. 调整 调节 校正 使...适于 vi. 适应

administration [ədminis'treiʃən] n. 管理 行政 管理部门

admire [əd'maiə] vt. 钦佩 羡慕 赞赏

admission [əd'miʃən] n. 允许进入 承认 许可 入会费

admit [əd'mit] vt. 承认 准许...进入 vi. 允许进入 允许 承认

adopt [ə'dɔpt] vt. 收养 采用 采取 接受

adult [ə'dʌlt, 'ædʌlt] n. 成年人 adj.成年的 成人的 成熟的

advance [əd'vɑ:ns] n. 前进 进展 进步 预支

advanced [əd'vɑ:nst] adj. 先进的 高级的

advantage[əd'vɑ:ntidʒ] n. 优点 优势 有利条件

adventure [əd'ventʃə] n. 冒险 奇遇 惊险活动

adverb ['ædvə:b] n. 副词 adj. 副词的

advertisement [əd'və:tismənt] n. 广告 公告 登广告

advice [əd'vais] n. 劝告 忠告 意见 建议

advisable [əd'vaizəbl] n. 明智的 可取的

advise [əd'vaiz] vt . 劝告 建议 通知 警告 vi. 劝告 与...商量

aeroplane [.eərə.plein] n. 飞机

aero ['ɛərəu] adj. 飞机的 航空的 飞行的

affair[ə'fɛə] n. 事情 事件 事务 绯闻

affect [ə'fekt] vt. 影响 感动 作用

affection [ə'fekʃən] n.慈爱 爱 爱慕 影响 喜爱 感情

afford [ə'fɔ:d] vt. 担负得起(后果 损失等) 提供 给予

afraid [ə'freid] adj. 害怕的 担心的

Africa ['æfrikə] n. 非洲

African['æfrikən] adj. 非洲的 非洲人的 n.非洲人

after ['ɑ:ftə] adj .以后的

afternoon ['ɑ:ftə'nu:n] n. 下午 午后 int. 下午好

afterward ['ɑ:ftəwəd] adv. 后来 以后

again [ə'gein] adv. 又一次 而且 又 再 另一方面

against [ə'genst, ə'geinst] prep. 倚在 倚靠 逆 对着 反对

age [eidʒ] n. 年龄 时代 时期 很长时间

agency ['eidʒənsi] n. 经办 代理 代理处 政府机构

agent['eidʒənt] n. 代理人 代理商 特工人员 药剂

aggressive [ə'gresiv] adj. 侵略的

ago [ə'gəu] adv. 以前

agony ['ægəni] n. 极度痛苦 挣扎

agree [ə'gri:] vt. 同意 赞成 承认 vi .同意 持相同意见 一致 符合

agreement [ə'gri:mənt] n. 协定 协议 同意 一致

agriculture ['ægrikʌltʃə] n. 农业 农艺 农学

ahead [ə'hed] adv. 在前 向前 提前 将来地 占优势

AI [ai] abbr. 人工智能= ( Artificial Intelligence )

aid [eid] n. 帮助 救护 助手 辅助物 v. 援助 帮助 救护

aim [eim] n. 目标 对准 枪法

air [ɛə] n. 空气 空中 外观

aircraft ['eəkrɑ:ft] n. 飞机 飞行器

airline ['eəlain] n. 航空公司 航线

airplane ['eəplein] n. 飞机=aeroplane(英）

airport ['eəpɔ:t] n. 机场 航空站

alarm [ə'lɑ:m] n. 惊恐 惊慌 忧虑 警报 警告 报警器

alcohol ['ælkəhɔl] n. 酒精 酒 醇 乙醇

alike [ə'laik] adj. 同样的 相同的 相似的 adv. 一样 以同样的方式

alive [ə'laiv]adj. 活着的 有活力 活跃的

all [ɔ:l] adj. 全部的 所有的

allow [ə'lau] vi. (for)考虑到 使可能

alloy['ælɔi] n. 合金 (金属的)成色

almost ['ɔ:lməust] adv. 几乎 差不多 adj. 几乎

alone [ə'ləun] adj. 单独的 仅仅 独自一人

along[ə'lɔŋ] adv. 向前 (与某人)一道 prep. 沿着

aloud [ə'laud] adv. 出声地 大声地

alphabet ['ælfəbet] n. 字母表 字母系统 符号系统 基本原理(元素)

already [ɔ:l'redi] adv. 早已 已经

also ['ɔ:lsəu] adv. 亦 也 而且 还 同样地

alter ['ɔ:ltə] vt. 改变 变更 改做

alternative [ɔ:l'tə:nətiv] adj. 两者择一的 供选择的

although [ɔ:l'ðəu] conj. 尽管 虽然

altitude ['æltitju:d] n.高 高度 海拔 高处 高地

altogether [.ɔ:ltə'geðə] adv.完全 总共 总而言之

aluminium [.ælju'miniəm] n.铝

always ['ɔ:lweiz] adv.总是 一直 永远

a.m n.(缩)上午 午前

amaze [ə'meiz] vt.使惊奇 使惊愕

ambition [æm'biʃən] n.雄心 抱负 野心 精力 vt.有...野心 追求

ambulance ['æmbjuləns] n.救护车 野战医院

America [ə'merikə] n.美洲 美国

American[ə'merikən] adj.美洲的 美国的 美式的 n.美国人 美式英语

among [ə'mʌŋ] prep.在...之中

amongst [ə'mʌŋst] prep在...之中 在...之间(=among)

amount [ə'maunt] n.总数 数量 总额 和 vi.总计 等于

ampere ['æmpeə] n.安培

amplify ['æmpli.fai] vt.放大 增强 扩大 详述 使...增幅

amuse [ə'mju:z] vt.逗乐 给...娱乐 消遣

analyse ['ænəlaiz] vt.分析 分解 解析 检讨 细察

analysis [ə'næləsis] n.分析 分解 解析

ancestor ['ænsəstə] n.祖宗 祖先 原种

anchor ['æŋkə] n.铁锚 vi.抛锚 停泊

ancient ['einʃənt] adj.古代的 古老的 n.古人 古货币

and [ənd, ænd] conj.和 又 并 则 逻辑与

angel ['eindʒəl] n.天使 天使般的人 神差 (名)安琪儿

anger ['æŋgə] n.怒 愤怒 vt.使发怒 激怒

angle ['æŋgl] n.角 角度 角落 vt.斜移 转变角度 vi.钓鱼 谋取 转变角度

angry ['æŋgri] adj.愤怒的 生气的 伤口红肿的

animal ['æniməl] n.动物 兽 兽性 与众不同的人 adj.动物的 野兽的 动物性的

ankle ['æŋkəl] n.踝 踝关节

announce [ə'nauns] vi.当电台的播音员 宣称 vt.宣布 宣告 发表 通知

announcer [ə'naunsə] n.宣告者 播音员 广播员 告知者

annoy [ə'nɔi] vt.使恼怒 使生气 打搅 骚扰 使...苦恼

annual ['ænjuəl] adj.每年的 年度的 一年生的

another [ə'nʌðə] adj.再一个的 别的 另一的 不同的 pron.另一个

answer ['ɑ:ŋsɚ] n.答案 回答 反驳 回应 vi.回答 符合 vt.回答 响应 适应

ant [ænt] n.蚂蚁

anticipate [æn'tisipeit] vt.预料 预期 期望 抢...前 语言 提前使用 vi.预期

anxiety [æŋ'zaiəti] n.焦虑 忧虑 渴望 担心

anxious ['æŋkʃəs] adj.焦急的 忧虑的 渴望的

any ['eni] adj.什么 一些 任何的 adv.稍 一丁点 pron.任何

anybody ['eni.bɔdi, 'enibədi] n.重要人物 pron.任何人

anyhow ['eni:.hau] adv.无论如何 不管怎样

anyone ['eniwʌn] pron.任何人

anything ['eni.θiŋ] pron.任何事物 一切

anyway ['eniwei] adv.无论如何 不管怎样

anywhere ['eniweə] adv.在什么地方 任何地方 n.任何(一个)地方

apart [ə'pɑ:t] adv.相隔 分开 除去 分别地 有区别地 adj.分开的 分离的

apartment[ə'pɑ:tmənt] n.一套公寓房间

apologize[ə'pɔlədʒaiz] vi.道歉 谢罪 认错

apology [ə'pɔlədʒi] n.道歉 认错 谢罪 勉强的替代物

apparatus [.æpə'reitəs] n.装置 器具 器械 仪器 器官 机构 组织

apparent [ə'pærənt] adj.表面上的 明显的

appeal [ə'pi:l] n.呼吁 申述 恳求

appear [ə'piə] vi.出现 来到 抵达 似乎 显得 出版

appearance [ə'piərəns] n.出现 出场 来到 外观

appetite ['æpitait] n.食欲 胃口 欲望 嗜好

apple ['æpəl] n.苹果 苹果树

appliance [ə'plaiəns] n.用具 器具 器械 装置 应用

applicable ['æplikəbəl] adj.能应用的 适当的 合适的

application[.æpli'keiʃən] n.应用 请求 申请 专心 施用 应用软件程序

apply [ə'plai] vt.应用 实施 使用 涂 使专心从事 vi.申请 有关联

appoint [ə'pɔint] vt.任命 委任 约定 指定 预约 装饰 vi.行使任命权

appointment [ə'pɔintmənt] n.任命 约定 约会 预约 委派 官职 设备

appreciate[ə'pri:ʃieit] vt.欣赏 感激 赏识 领会 充分意识 感谢 vi.增值

approach [ə'prəutʃ] n.接近 途径 方法 vt.向...靠近 靠近 接近 动手处理

appropriate[ə'prəupri:ət] adj.适当的 恰当的 相称的 vt.拨出(款项) 占用

approval [ə'pru:vəl] n.赞成 同意 批准 认可

approve [ə'pru:v] vt.赞成 称许 批准 核准 证实 vi.赞同

approximate [ə'prɔksimət] adj.近似的 大约的 vt.近似 接近 约等于

approximately [ə'prɔksimətli] adv.近似地 大约

April ['eiprəl] n.四月

arbitrary ['ɑ:bi.trəri] adj.随心所欲的 专断的 任意的 专制的 武断的 霸道的

architecture ['ɑ:kitektʃə] n.建筑学

area ['eəriə] n.面积 地区 区域 领域 方面

argue ['ɑ:gju:] vt.争论 辩论 争吵 劝说 表明 vi.提出理由 争论 争辩 辩论

argument ['ɑ:gjumənt] n.争论 辩论 理由 论点(据)

arise [ə'raiz] vi.出现 由...引起 上升

arithmetic[ə'riθmətik] n.算术 四则运算

arm[ɑ:m] n.手臂 臂状物 武器

army ['ɑ:mi] n.军队 陆军 大群

around [ə'raund] adj.大约 在现存范围内

arouse [ə=rauz] vt.引起 唤起 唤醒 激发 激起 vi.唤醒

arrange [ə'reindʒ] vt.筹备 整理 调解 安排 计划 改编(乐曲) vi.协商 计划

arrangement [ə'reindʒmənt] n.整理 排列 安排 商议

arrest [ə'rest] n.逮捕 监禁 vt.逮捕 拘留 阻止 妨碍 吸引

arrival[ə'raivəl] n.到达 到来 到达者

arrive [ə'raiv] vi.到达 来临 达到 成功

arrow ['ærəu] n.箭 箭头 箭状物 vi.快且笔直的前进有如穿梭的箭

art [ɑ:t] adj.美术(品)的 艺术(品)的 n.艺术 美术 技术 阴谋

article ['ɑ:tikəl] n.文章 条款 物品 冠词 vt.定契约

artificial [.ɑ:ti'fiʃəl] adj.人工的 人造的 虚伪的 武断的 娇揉造作的

artist ['ɑ:tist] n.艺术家 美术家 艺人

artistic [ɑ:'tistik] adj.艺术的 艺术家的

as [æs] adv.同样地 像 prep.当做 conj.当...的时候 像...一样 因为

ash [æʃ] n.灰 灰烬 灰末 骨灰

ashamed [ə'ʃeimd] adj.惭愧(的) 羞耻(的)

Asia ['eiʃə] n.亚洲

Asian ['eiʃən] adj.亚洲的 n.亚洲人

aside [ə'said] n.旁白 顺便说的话 adv.在旁边 到旁边 在一边 另外 离开

ask [ɑ:sk] vt.问 询问 要求 邀请 请求 开价 vi.询问 恳求

asleep [ə'sli:p] adj.睡着的 睡熟的 已死的

aspect ['æspekt] n.方面 样子 外表 外貌 外观 方位 方向

assemble [ə'sembəl] vt.集合 召集 装配 vi.集合 聚集

assembly [ə'sembli] n.集合 集会 装配

assess [ə'ses] vt.对(财产等)估价 估定 评定

assign [ə'sain] vt.指派 分配 指定 转让 n.受让人

assignment [ə'sainmənt] n.分配 功课 任务 被指定的(课外)作业 委派

assist [ə'sist] vt.援助 帮助 协助的器械 搀扶

assistant [ə'sistənt] n.助手 助理 助教 adj.副的

associate [ə'səuʃieit] vt. 联合 联想 vi.交往

association [ə.səusi'eiʃən] n.协会 团体 社团 联合 结合 交往 联想

assume[ə'sju:m] vt.假定 设想 承担 呈现 (想当然的)认为

assure [ə'ʃuə] vt.使确信 向...保证 保险 保证 确信 担保

astonish [əs'tɔniʃ] vt.使惊讶 使吃惊

astronaut ['æstrənɔ:t] n.宇宙航行员 宇航员

at [æt] prep.在...里 在...时

athlete ['æθlit, 'æθli:t] n.运动员 田径运动员

Atlantic [ət'læntik] adj.大西洋的 n.大西洋

atmosphere ['ætməsfiə] n.大气 空气 气氛 格调 情趣 气压

atmospheric [.ætməs'ferik] adj.大气的 大气层的 制造气氛的

atom ['ætəm] n.原子 微粒 微量

atomic [ə'tɔmik] adj.原子的 原子能的

attach [ə'tætʃ] vt.缚 系 贴 附加 使依恋

attack [ə'tæk] n.攻击 评击

attain [ə'tein] vt.达到 获得 完成

attempt [ə'tempt] vt.尝试 试图 n.企图

attend [ə'tend] vt.出席 照顾 护理 照料 参加 注意 vi.专心 留意 待命

attention [ə'tenʃən] n.注意(力) 留心 关心 立正

attentive [ə'tentiv] adj.注意的 留意的 有礼貌的

attitude ['ætitju:d] n.态度 看法 姿势

attract[ə'trækt] vt.吸引 引起 诱惑 有吸引力

attraction [ə'trækʃən] n.吸引 吸引力 有吸引力的人或事物 引力

attractive [ə'træktiv] adj.有吸引力的 引起注意的

attribute ['ætribju:t, ə'tribju:t] vt.把...归因于 属于 n.属性 特征 标志 象征

audience ['ɔ:diəns] n.听众 观众 读者 拥护者 倾听 正式会见

August ['ɔ:gʌst] n.八月(大写时） adj.威严的 尊严的(小写时）

aunt [ɑ:nt] n.伯母 婶母 姑母 姨

aural ['ɔ:rəl] adj.耳的 听觉的

Australia [ɔ'streiliə] n.澳大利亚

Australian [ɔ'streiliən] adj.澳大利亚的 n.澳大利亚人

author ['ɔ:θə] n.作者 作家 创造者 vt.创造 写作

authority [ə'θɔriti] n.当局 官方 权力 权威 专家 权威人士 依据

auto ['ɔ:təu] n.(口语)汽车autos=automobile 自动汽车 pref.自动的 自己的

automatic [.ɔ:tə'mætik] adj.自动的 机械的 半自动化的 n.自动装置 半自动武器

automation [.ɔ:tə'meiʃən] n.自动 自动化

automobile ['ɔ:təməbi:l, ɔ:tə'məubil] n.汽车 机动车 adj.汽车的

autumn ['ɔ:təm] n.秋 秋季 秋天

auxiliary [ɔ:g'ziljəri] n.帮助者 辅助物 助动词 adj.辅助的 附属的 附加的

available [ə'veiləbəl] adj.可利用的 可得到的 有用的 有效的 通用的

avenue ['ævənju:, 'ævinju:] n.林荫道 道路 大街 途径手段

average ['ævəridʒ] n.平均数 平均水平 adj.平均的 一般的 通常的 v.取平均值 达到平均水平

aviation [.eivi'eiʃn, 'evi'eʃən] n.航空 航空学 飞机制造业

avoid [ə'vɔid] vt.避免 躲开 逃避 撤消

await [ə'weit] vt.等候 期待 准备...以待 储存 vi.等待

awake [ə'weik] adj.醒着的 警惕的 vt.唤醒 唤起 激发 vi.醒来 恢复意识

award [ə'wɔ:d] n.奖 奖品 判定 vi.授予 给予 vt.授予

aware [ə'weə] adj.知道的 意识到的

away [ə'wei] adv.离开 远离 出去 连续的 遥远地 ...去 adj.远离

awful ['ɔ:fl] adj.令人不愉快的 可怕的 畏惧的 充满敬畏的 艰难的 adv.极其

awfully ['ɔ:fuli] adv.令人畏惧的 很 可怕地 非常地 极端地

awkward ['ɔ:kwəd] adj.笨拙的 尴尬的

ax [æks] n.斧子 （吉他或者萨克斯风类的）乐器 vt.用斧砍 削减 abbr.=(axiom axis)

axis ['æksis] n.轴 轴线 中心线 中枢

## B

baby ['beibi] adj.(形容用法)婴儿的 过小的 n.婴儿

back [bæk] adj. 后面的 偏远的 过时的 积欠的 向后的

background ['bækgraund] n.背景 后景 经历 幕后 配音 vt.提供背景

backward ['bækwəd] adj.向后的 倒的 adv.倒 向后

bacteria [bæk'tiəriə] n.(复数)细菌

bad [bæd] adj.坏的 恶的

badly ['bædli] adv.坏 差 严重地 恶劣地 极度地

badminton ['bædmintən] n.羽毛球

bag [bæg] n.袋 包 钱包 背包 猎获物

baggage ['bægidʒ] n.行李

bake [beik] vt.烤 烘 焙 烧硬 vi.烘焙 烤 变得炎热 n.烘焙 烤

balance ['bæləns] vt.使平衡 称 称重量 权衡 抵消

ball [bɔ:l] n.球 球状物 舞会 v.(把...捏)成球状

balloon [bə'lu:n] n.气球 玩具气球

banana [bə'nɑ:nə] n.香蕉 芭蕉属植物

band [bænd] n.乐队 带 波段 箍 队 v.联合 结合

bang [bæŋ] n.(突然)巨响 枪声 猛击 重击

bank [bæŋk] n.银行 库 岩 堤 vt.堆积 vi.开户 把...存入银行

banner ['bænə] n.旗 旗帜 横幅 大标题

bar [bɑ:] n.酒吧间 条 杆 栅 条 棒 横木 vt.禁止 阻挠

barber ['bɑ:bə] n.理发师

bare [bɛə] adj.赤裸的 显露的 极少的 仅仅的 vt.使赤裸 露出

bargain ['bɑ:gən] n.交易 买卖 物美价廉的东西 vi.议价 成交 vt.议价交易

bark [bɑ:k] n.吠叫声 狗叫 咆哮 厉声说话

barn [bɑ:n] n.谷仓 牲口棚

barrel['bærəl] n.桶 圆筒 枪管 炮管 笔管 一桶的量 大量 汽油桶 vi.快速移动 vt.把...装桶

barrier ['bæriə] n.栅栏 屏障 障碍 障碍物 界线

base [beis] n.基础 底层 基地 （支持、收入、力量等的）基础 vt.以...作基础 adj.卑鄙的 不道德的

basic ['beisik] n.基本 实质性的东西 基础训练 [计] BASIC语言 基本高级语言 adj.基本的 基础的 含硅量少的

basically ['beisikəli] adv.基本上 主要地

basin ['beisən] n.盆 洗脸盆 盆地 流域 水池

basis ['beisis] n.基础 根据 主要成分

basket ['bɑ:skit] n.篮 篓 筐 <篮>篮球筐 得分

basketball ['bɑ:skitbɔ:l] n.篮球 篮球运动

bat [bæt] n.球拍 短棍 棒 击打 蝙蝠 v.用棒击打 [计算机] DOS文件名：批文件

bath [bɑ:θ] n.浴 洗澡 洗澡水 浴缸 浴室 财政亏损 (复)游泳池 n.(Bath)巴斯(英国城市) v.洗澡

bathe [beið] vt.给...洗澡 弄湿 沐浴 用水洗 沉浸

bathroom ['bɑ:θrum, -ru:m] n.浴室 盥洗室

battery ['bætəri] n.电池 一套 一组 一系列 排炮

battle ['bætl] n.战役 斗争 争论 vi.与...作战 与...斗争

bay [bei] adj. 红棕色 n. 湾 海湾 山脉中的凹处 红棕色 隔间 配电间 犬吠 围困 月桂树 名望

vi. 嗥叫 vt. 吼叫着追赶或进攻 哀号 把...围困住

B.C.['bi: si:] (缩)公元前

be [bi:] aux.v. &vi.是 在 做 有

beach [bi:tʃ] n.海滩 湖滩 河滩 vt.拖(船)上岸

beam [bi:m] n.梁 横梁 束 柱 光线 电波 容光焕发 vt.闪亮 为...上梁 vi.微笑 发光

bean [bi:n] n.豆 蚕豆

bear [beə(r)] n.熊 粗鲁的人 vt.忍受 负荷 结果实 生子女

beard [biəd] n.胡须 络腮胡子 vt.抓住胡须 公开反对

beast [bi:st] n.兽 野兽 牲畜 凶残的人 令人厌憎的人

beat [bi:t]**n**.打 敲打声 拍子 独家报道 一会儿 vt. &vi.打 敲 打败

beautiful ['bju:tifəl] adj.美的 美丽的 漂亮的

beauty ['bju:ti] n.美 美丽 美人 美的东西

because [bi'kɔz] conj.由于 因为

become[bi'kʌm] vt.变得 vi.存在 变成 成为 变得 经历改变或发展

bed [bed] n.床 床位 圃 河床 底层 基座 vt.铺床 去睡觉 安置 发生性关系 vi.铺床 去睡觉 铺层 躺平

bee [bi:] n.蜂 密蜂 忙碌的人

beef [bi:f] n.牛肉 菜牛 v.抱怨 诉苦

beer [biə] n.啤酒

before [bi'fɔ:] adj. & adv.以前 在...之前 prep. & conj.在...以前 向...

beg[beg] vt.乞讨 恳求 合理需要 vi.行乞 乞求 abbr.(begin)的缩写 开始

beggar ['begə] n.乞丐 穷人 <非正式>人 家伙 vt.使贫穷 使不足 使不能

begin [bi'gin] vi.开始 起初是 vt.开始 意欲作出某事

beginner [bi'ginə] n.初学者 生手

beginning [bi'giniŋ] n.开始 开端 起源 早期 adj.刚开始的 动词begin的现在分词

behalf [bi'hɑ:f] n.利益 维护 支持

behave [bi'heiv] vi.表现 举止 行为 举止端正 运转

behavior [bi'heivjə] n.行为 举止 态度

behind [bi'haind] adv.在后面 落后 迟地 adj.后面的 落后的 迟的 慢的 prep.在...后面 落后于 掩盖 基于 支持 n.屁股

being ['bi:iŋ] n.存在 生物 生命 人 vi.存在 处于 举行 去或来 是 adj.现在的 目前的

belief [bi'li:f] n.信任 相信 信念 信仰

believe [bi'li:v] vt.相信 认为

bell [bel] n.钟 铃 门铃 钟声 花冠 吼叫声 vt.敲钟 使象钟状地张开 vi.使呈钟状 张开

belong [bi'lɔŋ] vi.属于 附属 适合于

beloved [bi'lʌvid] adj.为...的爱的 心爱的 n.爱人

below [bi'ləu] adv.在下面 向下 下级 prep.在...下面(以下) (级别、重要性)低于 adj.在...下面 n.下面的东西

belt [belt] n.带(状物) 腰带 皮带 区 vi.飞奔 大声唱 vt.系上腰带 环绕 击打 大声唱

bench [bentʃ] n.长凳 条凳 工作台 梯田 vt.坐在长凳上 放置长凳 在展览会上展览 vi.形成梯田

bend [bend] vt.使弯曲 vi.弯曲 屈服 n.弯曲(物) 弯道

beneath [bi'ni:θ] adv.在下方 prep.在...下方 在...之下

beneficial [.beni'fiʃəl] adj.有利的 有益的

benefit ['benifit] n.利益 恩惠 津贴 义卖 义演 vt.有益于 vi.得益

berry ['beri] n.浆果(如草莓等)

beside [bi'said] prep.在...旁边 与…相比

besides [bi'saidz] adv.而且 也 此外 prep.除...之外 adj.别的

best [best] adj.最好的 最大的 adv.最 最好地 n.最好的人(或物) vt.胜于

bet [bet] n.打赌 赌注 最大的可能 vi.下注 vt.打赌

betray [bi'trei] vt.误导 出卖 背叛 辜负 泄漏 vi.证明...错误

better ['betə] adj.较好的 更好的 adv.更好地 v.使...更好 超越 n.更好的事物

between [bi'twi:n] adv.在中间 prep.在...中间

beyond [bi'jɔnd] adv.在更远处 另外 n.远处 来世 prep.在...的那边 超出 越过 另外

Bible ['baibl] n.基督教《圣经》 有权威的书

bicycle ['baisik(ə)l] n.自行车 脚踏车 v.骑自行车

big [big] adj.大的 巨大的 重要的 adv.以郑重的方式 大数量 n.杰出的组织和个人

bike [baik] n.自行车 vi.骑自行车

bill [bil] n. 比尔(男名) 账单 钞票 招贴 票据 清单 议案 法案 广告 鸟嘴 喙 vt. 开帐单 用海报宣传 把...列成表

billion ['biljən] num.(英)万亿 (美)十亿 n.大量 adj.大量的

bind [baind] n.窘境 vt.捆绑 包扎 装钉 约束 约束

biology [bai'ɔlədʒi] n.生物学 生态学

bird [bə:d] n.鸟 家禽 陶土飞靶 伙伴 vi.观察或辨认鸟

birth [bə:θ] n.分娩 出生 诞生 出身 血统 出现 起源

birthday ['bə:θdei] n.生日 诞生的日期

biscuit ['biskit] n.(英)饼干 (美)软饼 姜黄褐色

bit[bit] n.一点 少量 一些 小片 钻头 马嚼子 辅币 位 比特(二进位制信息单位) vt.控制

bite [bait] n.咬 一口 少量食物 咬伤的伤口 vt.咬 叮 螫 剌穿 vi.咬(饵) 使剧痛 紧抓

bitter ['bitə] adj.苦的 痛苦的 剧烈的 严寒的 n.苦久 苦涩 苦的东西 vt.使...变苦 adv.激烈地 极其

bitterly ['bitəli] adv.苦苦地 悲痛地 怨恨地 残酷地

black [blæk] adj.黑的 暗的 穿黑衣服的 脏的 邪恶的 阴沉的 机密的 n.黑色东西 黑色 黑人 (棋类)黑子 v.(使)变黑 抹黑

blackboard ['blækbɔ:d] n.黑板

blade [bleid] n.刀刃 刀片 叶片 刀锋

blame [bleim] n.过失 责备 vt.责备 把...归咎于

blank [blæŋk] adj. 空白的 茫然的 全部的 面无表情的 未完工的 n. 空白(处) 空隙 空虚 (有弹药而无弹头的)空弹 vi. 消失 走神 vt. 使模糊不清 删除 封闭 (比赛时连续得分)使对手得零分

blanket ['blæŋkit] n.毛毯 羊毛毯 覆盖物 排字版 vt.用毯子裹 扑灭 挡风 覆盖 使包含 adj.包含所有的 适用于所有情形的

blast [blɑ:st] n.爆炸 冲击波 气流 一阵 汽笛声 vi.爆破 突然发出声音 vt.喷射 炸掉

blaze [bleiz] n.火焰 烈火 闪光 vi.燃烧 发光 vt.燃烧 宣布 在树皮上刻痕 领导 开拓

bleed [bli:d] vi. 出血 流血 泌脂 ( 为事业 祖国 ) 负伤或牺性 ( 花草﹑ 树木等 ) 流出汁液 出大笔钱

vt. 放血 勒索钱财 大大削弱 使...流血 n. 印刷到纸边上去 漏出

blend [blend] vt. &vi.混合 n.混合物

bless [bles] vt.为...祝福 赐福 祈佑 称颂上帝 使神圣 赋予

blind [blaind] adj.瞎的 盲目的 未察觉的 n.百叶窗 幌子 vt.使失明 隐藏 adv.盲目地

block [blɔk] n.街区 木块 石块 阻塞(物) 障碍(物) vt.堵塞 拦阻

blood [blʌd] n.血 血液 血统 仇恨 vt.流血

bloom [blu:m] n.花 开花 开花期 钢块 兴旺 vt.使...开花 vi.开花 变得健康

blossom ['blɔsəm] n.花 开花 全盛期 vi.开花 成长

blow [bləu] vi.吹 吹动 吹响 打击 殴打 v.吹 风吹 擤（鼻子） 【口】浪费（机会）

blue [blu:] adj.蓝色的 沮丧的 忧郁的 (女人)有学问的 n.蓝色 vt.使...变蓝 vi.变蓝

board [bɔ:d] n. 板 木板 甲板 董事会 伙食 舞台 演员的职业 船舷 vt. 上(飞机、船、车等) 有偿提供食宿 用木板覆盖 vi.搭伙

boast [bəust] n.吹牛 vi.吹牛 自夸 vt.吹牛 吹嘘

boat [bəut] n.小船 艇 渔船 船型物体 vt.放于船上 vi.划船

body ['bɔdi] n.身体 人 主体 尸体 正文 vt.赋予形体 体现

boil [bɔil] n.皮下脓肿 沸腾 vi.沸腾 汽化 vt.煮沸

bold [bəuld] adj.大胆的 冒失的 粗体的 醒目的 无礼的 陡峭的

bolt [bəult] n. 螺栓 螺钉 插销 门闩 突发 剑弩 闪电 vt. 闩门 闩住 用螺栓固定在一起 狼吞虎咽

vi. 冲出去 急逃 adv. 背部挺直地坐或站

bomb [bɔm] n.炸弹 高压罐 火山口喷出的熔岩 vt.轰炸 大败 长传 vi.失败

bond [bɔnd] n.联结 联系 结合 公债 债券 粘结剂 粘合剂

bone [bəun] n.骨 骨骼 核心 vt.除去骨头 往衣服里加撑条 磨光表面 vi.努力学习 adv.非常地 极其地

book [buk] n.书 书籍 vt.预定 登记 vi.通过...预订

boot [bu:t] n. 靴子 长统靴 (英)汽车行李箱 猛踢 锁扣 vt. 踢 穿靴子 锁车 vi. (计算机) 启动 在(通常为非法停放的汽车)车轮上装制动装置 利用 帮助

booth [bu:θ] n.货摊 公用电话亭 小间

border ['bɔ:də] n.边 边缘 边界 vt.作...之疆界 加边

bore [bɔ:] vbl.忍受(动词bear 过去式) n.令人讨厌的人 枪膛 孔 井眼 枪管口径 涌潮 vt.使厌烦 钻 挖 vi.挖掘 开凿

born [bɔ:n] 动词bear的过去分词 adj.天生的 出生的 与生俱来的

borrow ['bɔrəu] vt.借 借用 借入

bosom ['buzəm] n.胸 胸部 内心

boss [bɔs] n.老板 上司 vt.指挥 控制

both [bəuθ] adj.两者的 conj.不但…而且… pron.两者(都)

bother ['bɔðə] vt.烦扰 迷惑 vi.烦扰 操心 n.麻烦 烦扰

bottle ['bɔtl] n.瓶 酒瓶 一瓶 vt.装瓶 抑制

bottom ['bɔtəm] n. 底 底部 根基 臀部 船身 水底 服装下面的部分 低洼地 底色 尽头 adj. 底部的 vt. 给...装底 建立基础 到达底部 vi. 到达底部 垫底 降到最低点

bough [bau] n.大树枝

bounce [bauns] n.跳 反跃 (活)弹力 vt.弹跳 向...透露注意 vi.反跳 弹起 跳起 重新恢复

bound [baund] n.跳跃 界限 范围 adj.一定的 必定的 有义务的 受约束的 装订的 vt.跳跃前进 形成界限

boundary ['baundri] n.分界线 边界

bow [bəu,bau] n.弓 蝴蝶结 鞠躬 船头 两手准备 vi.鞠躬 成弓形 弯腰 首次上演 屈服 听从 尊重 vt.重压 弯曲

bowl [bəul] n.碗 钵 碗状物 物体的碗状部分 季后赛 圆形露天剧场 vi.打保龄球 快速移动 vt.投球 投球得分

box [bɔks] n.箱 盒 包箱 专席 亭子 窘境 vt. 装...入盒中 限制住 调和 改变形状 用拳击… vi.参与拳击 拳击 打拳

boy [bɔi] n.男孩 儿子 少年 家伙

brain [brein] n.脑 脑髓 脑力 智力 vt.打...的脑袋

brake [breik] n.蕨菜 闸 刹车 制动器 vt. & vi.制动 煞车 压榨

branch [brɑ:ntʃ] n.分支 树枝 分部 分科 vt & vi.分支 分岔

brand [brænd] n.燃烧的木条 剑 商品 商标 牌子 烙印 类型 vt.铭刻 打烙印 污蔑

brandy ['brændi] n.白兰地酒 vt.以白兰地酒调制

brass [brɑ:s] n.黄铜(制品) 黄铜器 铜管(乐器) adj.黄铜的

brave [breiv] adj.勇敢的 华丽的 缤纷的 vt.勇敢地面对(危险等) 抵抗 n.勇敢的人

Brazil [brə'zil] n.巴西

Brazilian [brə'ziljən] n.巴西人 adj.巴西的

bread [bred] n.面包 食物 粮食 生计 vt.撒面包屑

breadth [bredθ] n.宽度 幅度 幅面

break [breik] n.休息 中断 破裂 vt.打破 损坏 弄破 弄坏 破坏

breakfast ['brekfəst] n.早饭 早餐 vt.吃早餐 vi.提供早餐

breast [brest] n.乳房 胸脯 胸膛 vt.面对 以胸对着

breath [breθ] n.气息 呼吸 气味 微风 迹象 精神 一种说话的声音

breathe [bri:ð] vi.呼吸 vt.呼吸 轻声说 流露感情 注入

breed [bri:d] n.品种 族类 vt.使繁殖 养育 繁殖 引起 vi.繁殖

breeze [bri:z] n.微风 和风 轻而易举的事 vi.飘然而行 轻松地得胜或进步

brick [brik] n.砖 砖块 砖状物 积木 可靠的朋友 vt.(up)用砖砌补 用砖堵住

bridge [bridʒ] n.桥 桥梁 桥牌 鼻梁 vt.渡过 架桥

brief [bri:f] n.摘要 简报 adj.简短的 简洁的 短暂的 vt.作简报 简单讨论

bright [brait] adj.明亮的 阳光的 生动的 聪明的 adv.亮 n.亮色

brighten ['braitn] vt.使发光 使快活 使变亮 使生辉 vi.发光 发亮

brilliant ['briljənt] adj.光辉的 卓越的 灿烂的 有才气的 杰出的 n.宝石

brim [brim] n.边 边缘 帽沿 v.(装)满 (使)溢出

bring [briŋ] vt.带来 引出 促使 劝诱 vi.产生

brisk [brisk] adj.活泼的 活跃的 快的 清新的 敏锐的 凛冽的

bristle ['brisl] n.短而硬的毛 鬃毛

Britain ['britən] n.不列颠 英国

British ['britiʃ] adj.不列颠的 英国的 英联邦的 n.英国人 英国英语

brittle ['britl] adj.脆的 易损坏的 易碎的

broad [brɔ:d] adj.宽广的 阔的 广泛的 清楚无误的 明显的 粗俗的 adv.完全地 n.(复）河流的延伸

broadcast ['brɔ:dkɑ:st] n.广播 广播节目 播音 adj.广播的 vt. & vi.广播 adv.经广播 四散地

broken ['brəukən] adj.被打碎的 骨折的 坏掉的 坏掉的 v.打破 弄坏(break的过去分词)

bronze [brɔnz] adj.青铜色的 n.青铜 青铜制品 青铜色 铜牌(第三名) vt.使成青铜色

brood [bru:d] n.窝 同窝幼鸟 vt.孵(蛋) 沉思

brook [bruk] n.小河 溪流 vt.(常用于否定句或疑问句)容忍 忍受

broom [bru:m] n.扫帚 灌木 vt.扫

brother ['brʌðə] n.兄弟 同事 同胞 教友

brow [brau] n.额 眉 眉毛

brown [braun] (姓)布朗 adj.褐色的 棕色的 n.褐色 棕色

bruise [bru:z] n.青肿 伤痕 擦伤 瘀青 挫伤 vi.受伤 擦伤 vt.使挫伤

brush [brʌʃ] n.刷子 毛刷 画笔 轻擦 小冲突 灌木丛 vt.用刷子清洗 画 拭去 打发 vi.轻擦 拂拭

brute [bru:t] n.禽兽 畜生 畜生 残忍的人 adj.残忍的 无理性的 粗暴的

bubble ['bʌbl] n.泡 泡沫 气泡 幻想 骗局 欢乐 vi.冒泡 沸腾 起泡 出现 洋溢 vt.使冒泡

bucket ['bʌkit] n.水桶 吊桶 铲斗 vt.装在桶里 vi.急急忙忙

buckle ['bʌkl] n.皮带扣 vt. & vi.扣住 弯曲 专心做事 屈服 adj.带扣 弯曲的

bud [bʌd] n.芽 萌芽 蓓蕾 vi.发芽 萌芽 vt.使发芽

build [bild] vt. & vi.建筑 建立 创立 建造 创建 开发 逐渐增强 n.体格 身材

building ['bildiŋ] n.建筑物 大楼 建筑

bulb [bʌlb] n.电灯泡 球状物

bulk [bʌlk] n.物体 容积 大批 大部分 大多数 大块 大批 容积 体积 vi.变大 增加 vt.膨胀 出现 adj.大量的

bull [bul] adj.大型的 公牛的 雄性的 n. 公牛 雄的象 买空着 像牛的东西 斗牛犬 教宗训谕 法令 vi. 猛冲 vt.暴力 强制

bullet ['bulit] n.枪弹 子弹 弹丸

bunch [bʌntʃ] n.束 球 串 一群 一组 凸起 大量 vi.凸起 使成一束(或一群等) vt.使成一束

bundle ['bʌndl] n.捆 包 束 包袱 一批 一大笔钱 vt.捆 匆匆送走 附赠 vi.匆忙

burden ['bə:dn] n.担子 负担 重担 装载量 v.使负重 装载 烦扰

bureau ['bjuərəu] n.局 司 处 社 所 办公处

burn [bə:n] vt. & vi.烧 燃烧 烧着 烧毁 灼伤 n.烧伤 灼伤

burst [bə:st] n.破裂 阵 爆发 vt. & vi.爆裂 突发

bury ['beri] vt.埋葬 葬 埋藏

bus [bʌs] n.公共汽车

bush [buʃ] n.灌木 灌木丛 矮树 荒野 狐狸尾巴 vt. 用灌木保护或支撑 加衬套于 vi.成灌木形 丛生 adj.长得低矮的 二流的

business ['biznis] n.商业 生意 事务

busy ['bizi] adj.忙的 繁忙的 杂乱的 vt.使忙于 vi.忙

but [bʌt] adv.仅仅 只 conj.但是 可是 prep.除...以外 pron.只有

butcher ['butʃə] n.屠夫 屠杀者 刽子手 肉商 小贩 vt.屠宰 滥杀 将某事弄得一团糟

butter ['bʌtə] n.黄油 阿谀奉承 vt.涂黄油

butterfly ['bʌtəflai] n.蝴蝶 蝶状物 蝶泳 vt.(烹饪时把鱼肉等)切开摊平

button ['bʌtn] n.扣子 按钮 像纽扣的东西 未成熟的蘑菇 按钮 vt.扣紧 用纽扣装饰 钉纽扣 闭嘴 vi.扣上

buy [bai] n.合理的价格 购买 vt.买 购买 换得 收买 接受 vi.购买

by [bai] prep.在...旁 被 由 经由 在...之前 根据 adv.靠近 经过 放在一旁地 adj.在旁边的 附带的 n.次要的事件

## C

cabbage ['kæbidʒ] n.洋白菜 卷心菜 vt.偷窃

cabin ['kæbin] n.小木屋 船舱 机舱 vt.抑制 vi.住在小屋

cabinet ['kæbinit] n.橱 柜 内阁 adj.私人的

cable ['keibl] n.缆 索 电缆 海底电报 vt.打海底电报 把...拧成缆 用缆捆住 vi.打海底电报

cafe [kə'fei] n.咖啡馆 小餐厅

cafeteria [.kæfi'tiəriə] n.自助食堂 adj.有多种选择的

cage [keidʒ] n.笼 鸟笼 囚笼 监狱 笼状结构 vt.关入笼中

cake [keik] n.饼 糕 蛋糕 块状物 小菜一碟(容易做成的事) vt.涂厚厚的一层 覆盖 填满 vi.结成块

calculate ['kælkjuleit] vt.计算 估计 考虑 推测 计划 vi.计算 预测 依赖

calculation [.kælkju'leiʃən] n.计算 计算结果

calculator ['kælkju.leitə] n.计算器 计算者

calendar ['kælində] n.日历 月历 日程表 历书 历法 vt.进入日历

call [kɔ:l] n.呼叫 访问 打电话 请求 把…看作 vt.把...叫做 叫 喊 呼叫 访问 打电话 vi.(短暂的)拜访 呼叫

calm [kɑ:m] n.平稳 风平浪静 adj.静的 平静的 冷静的 vt. & vi.平静下来 镇静

camel ['kæməl] n.骆驼 浮船箱 浅黄棕色

camera ['kæmərə] n.照相机 摄影机 罗马教廷的财政部

camp [kæmp] n.野营 营地 兵营 露营 帐棚 训练营 做作 vt. & vi.露营 扎营 做作 adj.做作的

campaign [kæm'pein] n.战役 运动 活动 vt.参加竞赛 vi.(从事、组织)活动

campus ['kæmpəs] n.校园 学校场地

can [kæn] aux.v.能 会 可能 n.罐头 听头 容器 vt.罐装 vt.炒某人鱿鱼

Canada ['kænədə] n.加拿大

Canadian [kə'neidjən] adj.加拿大的 加拿大人的 n.加拿大人

canal [kə'næl] n.运河 沟渠 管 气管 食管 vt.建运河 疏导

cancel ['kænsl] n.取消 撤消 vt.取消 撤消 删去 约分 vi.抵消

cancer ['kænsə] n.癌 癌症 肿瘤 巨蟹座

candidate ['kændidit] n.候选人 投考者 求职者

candle ['kændl] n.蜡烛 烛形物 烛光

candy ['kændi] n.糖果 砂糖结晶 vt.用糖煮 使结晶为砂糖 vi.结晶为砂糖

cannon ['kænən] n.大炮 火炮 榴弹炮 v.开炮

canoe [kə'nu:] n.独木舟 皮艇 划子 轻舟 v.乘独木舟

canon ['kænən] n. 教会法规 标准、准则 正典 正式名单 教堂里的教士 佳能 (财富500强公司之一 总部所在地日本 主要经营办公设备)

canteen [kæn'ti:n] n.小卖部 临时餐室 食堂

canvas ['kænvəs] n.粗帆布 一块油画布 (帆布)画布 油画

cap [kæp] n.帽子 便帽 帽状物 盖子 顶 vt.给戴帽 覆盖顶端 超过 形成化学键于 抑制 vi.形成化学键

capable ['keipəbl] adj.有能力的 有才能的 足以胜任的 有...倾向的

capacity[kə'pæsiti] n.容量 能力 能量 容积 资格 职位 adj.(达到最大容量)满的

capital ['kæpitl] n.资本 资金 首都 大写字母 adj.大写的 资本的 首要的

captain ['kæptin] n.陆军上尉 队长 船长 首领 vt.率领 指挥

captive ['kæptiv] n.俘虏 被监禁的人 迷恋者 adj.被俘的 被迷住的

capture ['kæptʃə] n.抓取 战利品 捕获之物 vt.捕获 俘获 夺得 抓取 获得 迷住

car [kɑ:] n.汽车 小汽车 轿车 车厢 吊舱

carbon ['kɑ:bən] n.碳 灯芯 复写纸

card [kɑ:d] n.卡 卡片 名片 纸牌 打牌 节目单 vt.附上或提供卡片 印卡 在卡片上列出 检查资格

care [kɛə] n.小心 照料 忧虑 慎思 焦虑因素 vi.关心 介意 照顾 喜爱 vt.介意 希望 abbr.=Cooperative for American Remittances to Europe 汇款到欧洲合作社

career [kə'riə] n.生涯 职业 事业 经历 vi.猛冲 飞跑 vt.在...中急行

careful ['kɛəfəl] adj.小心的 仔细的 细致的

careless ['kɛəlis] adj.粗心的 漫不经心的 疏忽的 n.不关心的 粗心的 [反]attentive

cargo ['kɑ:gəu] n.船货 货物

carpenter ['kɑ:pintə] n.木工 木匠 v.做木工活

carpet ['kɑ:pit] n.地毯 毡毯 毛毯 vt.铺以地毯 铺盖

carriage ['kæridʒ] n.客车厢 四轮马车

carrier ['kæriə] n.运输工具 运载工具 运送者 行李架 客运公司 航空母舰 带菌者 电信公司

carrot ['kærət] n.胡罗卜 报酬 好处

carry ['kæri] n. 搬运 射程 投射 联运 vt. 负载 承担 携带 运载 运输 传送 传播 表达 带有 容纳 怀孕 延续 推动 赢得 记录 vi. 负载 被传达 容许被运输 被通过

cart [kɑ:t] n.(二轮或四轮)运货马车 手推车 vt.用马车装载 用手提(笨重物品) 强行带走

carve [kɑ:v] vt.刻 雕刻 切开 vi.雕刻 切割

case [keis] n.情况 事实 实情 病例 箱(子) 盒(子) 套 案例 vt.装箱 踩点

cash [kæʃ] n.现金 现款 vt.兑现 付现款 adj.现金的

cassette [kə'set] n.盒式录音带 盒子 匣子 盒式录音机

cast [kɑ:st] n.演员阵容 投掷 vt.投 扔 抛 浇铸

castle ['kɑ:sl] n.城堡 巨大建筑物 vt.置于城堡中 (棋)移动王车易位 vi.(棋)移动王车易位

casual ['kæʒjuəl] adj.偶然的 随便的 非正式 漫不经心的

cat [kæt] n.猫 猫科 猫皮 邪恶的女人 吊锚机 单帆小船 vt.吊锚 vi.寻找性交伴侣 (CAT)电子计算机横断层扫描

catalog ['kætəlɔ:g] n. 目录 目录册 大学概况手册 商品型录 vt. 做目录 分类说明 名入列表 vi. 研究(手册或目录) (商品)在型录中明码标价

catch [kætʃ] n.捕捉 陷阱 捕获物 接(球等) 扣栓物 隐患 vt. 捉住 赶上 领会 了解 感染 被钩住 击中 察觉 vi. 抓住 燃着

cathedral [kə'θi:drəl] n.总教堂 大教堂

cattle ['kætl] n.牛 牲口 家畜 畜牲(复数)

cause [kɔ:z] n.原因 理由 事业 vt.引起 使发生

cave [keiv] n.山洞 洞穴 窑洞 vt.挖空 塌落 屈服 vi.塌落

cease [si:s] vi. &vi. &n.停止 终了 停息

ceiling ['si:liŋ] n.天花板 顶蓬 上限

celebrate ['selibreit] vt.庆祝 歌颂 赞美 vi.庆祝 祝贺 颂扬

cell [sel] n.细胞 小房间 电池 小组 单人牢房 (蜂房的)巢室 手机

cellar['selə] n.地窑 地下室 vt.把...藏入地窑(或酒窑)

cement [si'ment] n.水泥 胶泥 纽带 接合剂 牙骨质 补牙物 基石 vt.粘结 接合 用水泥涂 vi.接合起来

cent [sent] n.分 分币 百(分之)

center ['sentə] n.中心 中间 主角 中锋 重点 vi.居中 聚焦于... vt.使集中 放在中央

centigrade ['sentigreid] adj.百分度的 摄氏的

centimetre ['senti.mi:tə] n.公分 厘米=centimeter(美）

central ['sentrəl] adj.中心的 主要的 基本的 折中的 中枢的 n.电话交流 中央办公室 中心

centre ['sentə] n.中心 中枢 聚集点 vt.集中 置于中央 关注于 vi.居中

century ['sentʃuri] n.世纪 一百年 成百的东西

ceremony ['seriməni] n.典礼 仪式 礼节 礼仪

certain ['sə:tn] adj.确实的 肯定的 必然的 特定的 pron.某几个 某些

certainly ['sə:tənli] adv.一定 必定 当然

certainty ['sə:tnti] n.必然 肯定 确定 确实的事情

certificate [sə'tifikit] n.证(明)书 证件 执照 vt.批准 认可 发证书给...

chain [tʃein] n.链 链条 项圈 连锁 束缚 vt.用铁练锁住 束缚 囚禁

chair [tʃɛə] n.椅子 主席 席位 讲座 要职 vt.上任 使担任(某事务)的主席 主持

chairman ['tʃɛəmən] n.主席 议长 会长

chalk [tʃɔ:k] n.白垩 粉笔 v.用粉笔写 记录

challenge ['tʃælindʒ] n.挑战 要求 需要 v.向...挑战

chamber ['tʃeimbə] n.会议室 房间 会所 室 腔 vt.装入室中(枪膛) adj.室内演奏的

champion ['tʃæmpjən] n.冠军 得胜者 拥护者 勇士 vt.保卫 拥护

chance [tʃɑ:ns] n.机会 意外 运气 风险 可能性 vi.偶然发生 vt.偶然发生 冒险

change [tʃeindʒ] n.改变 变化 零钱 vt. & vi.改变

channel ['tʃænl] n.海峡 渠道 通道 频道 方法 vt.引导 开导 形成河道 输送

chapter ['tʃæptə] n.章 回 篇 牧师的例行会议 地方分会

character ['kæriktə] n.性格 个性 品质 特性 角色 字符 人物 名誉 地位 adj.[剧](角色)代表某一特性的

characteristic [.kæriktə'ristik] adj.特有的 典型的 n.特性 特征 特色

charge [tʃɑ:dʒ] vt. & vi.索价 控告 加罪于 要价 赊帐 充电 管理 n.电荷 指控 费用 照顾 责任

charity ['tʃæriti] n.施舍 慈善事业 慈善 慈善机关(团体) 仁慈 宽厚

charming ['tʃɑ:miŋ] adj迷人的 可爱的

chart [tʃɑ:t] n.图 图表 海图 vt.制成图表

chase [tʃeis] n.追逐 追赶 追求 狩猎 争取 vt.追捕 狩猎

cheap [tʃi:p] adj.廉价的 便宜的 劣质的 品质低的 虚伪的 adv.便宜地

cheat [tʃi:t] n.骗子 欺骗行为 vt.骗取 哄 作弊 vi.行骗 作弊

check [tʃek] vt.检查 制止 阻止 核对 (国际象棋)将军 n.检查 支票 账单 制止 阻止物 检验标准 (国际象棋)将军

cheek [tʃi:k] n.面颊 脸蛋 vt.粗鲁地向...讲

cheer [tʃiə] n.愉快 激励 欢呼 vt.欢呼 加油 鼓舞 快活起来 vi.使振奋 欢呼

cheerful ['tʃiəfəl] adj.快乐的 愉快的

cheese [tʃi:z] n.乳酪 干酪 要求照相的人说的口形词 vt.停止

chemical ['kemikəl] adj.化学的 n.化学制品

chemist ['kemist] n.化学家 药剂师

chemistry ['kemistri] n.化学 化学性质 相互作用

cheque [tʃek] n.支票

cherry ['tʃeri] n.樱桃 樱桃树 樱桃色

chess [tʃes] n.棋 国际象棋

chest [tʃest] n.胸腔 胸膛 (大)箱子 金库 资金 一箱 密封室：衣橱

chew [tʃu:] vt.咀嚼 嚼碎 损坏 vi.咀嚼 n.咀嚼 咀嚼物

chicken ['tʃikin] n.小鸡 小鸟 鸡肉 胆小 懦夫 adj.懦弱的 胆小的

chief [tʃi:f] adj.主要的 首席的 n.首领

child [tʃaild] n.小孩 儿童 儿子 产物

childhood ['tʃaildhud] n.童年 幼年 早期

childish ['tʃaildiʃ] adj.孩子气的 幼稚的 简单的 年老糊涂的

chill [tʃil] vt. 使变冷 使...寒心 冷冻 使...惊恐 使泄气 vi. 变冷 发抖 凝固 感到寒冷 n. 寒冷 寒意 失意 恐惧 adj. 寒冷的 冷漠的 感冒的 扫兴的

chimney ['tʃimni] n.烟囱 烟筒 玻璃罩

chin [tʃin] n.颏 下巴 vt.做单杠 vi.作引体向上 聊天

China ['tʃainə] n.中国

china ['tʃainə] n.瓷器 瓷料

Chinese ['tʃai'ni:z] adj.中国的 n.中国人 汉语

chocolate ['tʃɔkəlit] n.巧克力 巧克力糖 巧克力饮品 adj.巧克力的 有巧克力糖衣的 巧克力色的

choice [tʃɔis] n.选择 挑选 抉择 adj.上等的 精选的

choke [tʃəuk] vt.(掐住或阻塞气管)使(某人)停止呼吸 塞满 vi.窒息 阻塞 n.窒息 阻风门开关 阻塞

choose [tʃu:z] vt.选择 挑选 情愿 vi.选择

chop [tʃɔp] vt.砍(价) 劈 切(细) 剁碎 vi.砍 n.厚肉片 排骨 砍

Christian ['kristʃən] n.基督教徒 信徒 adj.基督教(徒)的 信奉基督教的 仁慈的

Christmas ['krisməs] n.圣诞节

church [tʃə:tʃ] n.教堂 礼拜堂 礼拜仪式 基督教徒 教会 adj.教堂的 vt.把...带到教堂接受某种宗教仪式

cigarette [.sigə'ret] n.香烟 纸烟 卷烟

cinema ['sinimə] n.电影院 电影 影片

circle ['sə:kl] n.圆 圆周 圈子 社交圈 循环 vt. & vi.包围 盘旋 环绕

circuit ['sə:kit] n.电路 环行 一圈 巡行 巡回 vt. & vi.巡回

circular ['sə:kjulə] adj.圆的 循环的 n.传单 通报

circulate ['sə:kjuleit] vt.使循环 使流通 vi.流通 循环 传播

circumference [sə'kʌmfərəns] n.圆周 周长 周围 圆周线 胸围

circumstance ['sə:kəmstəns] n.情况 条件 境遇 环境 (复数)境况 事件 详情

citizen ['sitizn] n.公民 市民 居民

city ['siti] n.城市 都市

civil ['sivil] adj.国内的 公民的 文职的 文明的 有礼貌的 民用的

civilization [.sivilai'zeiʃən] n.文明 文化 开化

civilize ['sivilaiz] vt.使文明 教育

claim [kleim] vt.要求 请求 声称 主张 断言 索取 n.要求 要求权 主张 断言 声称 要求物

clap [klæp] vi.拍手 砰的一声 突然离开 vt.拍 轻拍 轻敲 鼓掌 击掌 急速放置 n.拍手 拍手声 一阵强风

clarify ['klærifai] vt.澄清 阐明 使...明晰

clasp [klɑ:sp] n.扣子 钩子 别针 紧握 v.扣紧 紧握 密切合作

class [klɑ:s] n.班 班级 (等)阶级 种类 vt.分类

classical ['klæsikəl] adj.古典的 经典的 典雅的

classification [.klæsifi'keiʃən] n.分类 分级 分类法

classify ['klæsifai] vt.把...分类 分类 归类

classmate ['klɑ:smeit] n.同班同学

classroom ['klɑ:srum] n.教室 课堂

claw [klɔ:] n.爪 钳 脚爪 螯 爪状物 vt. & vi.抓 撕

clay [klei] n.粘土 泥土 肉体

clean [kli:n] adj.清洁的 干净的 纯洁的 正当的 完全的 精准的 空的 adv.完全地 n.清洁 vt. & vi.打扫 清扫

clear [kliə] adj. 清晰的 清楚的 明确的 澄清的 头脑清醒的 vt. 清除 清除障碍 澄清 使明亮 明确

vi. 变清晰 消失 放晴 adv. 清楚地 一直 n. 间隙 空地 开旷的地方

clearly ['kliəli] adv.明白地 清晰地 adj.清楚的

clerk [klɑ:k] n.店员 办事员 职员

clever ['klevə] adj.聪明的 机敏的

cliff [klif] n.悬崖 峭壁

climate ['klaimit] n.气候 风土 地带 风气 气氛

climb [klaim] vt. & vi.攀登 爬 上升 n.攀登(处)

cloak [kləuk] n.斗篷 覆盖(物) 宽大外衣 掩护 v.遮掩 隐匿

clock [klɔk] n.钟 时钟 计时器 仪表

close [kləus,kləuz] vt. 关 闭 结束 靠近 vi. 关 停业 搏斗 n. 完结 结论 adj. 靠近的 亲近的 几乎 密切的 势均力敌的adv. 紧挨着 紧紧地

closely ['kləusli] adv.紧密地 接近地 严密地

cloth [klɔθ] n.布 衣料 桌布

clothe [kləuð] vt.给...穿衣服 穿上 覆上 授以 赋予 表达

clothes [kləuðz] n.衣服 服装 被褥

clothing ['kləuðiŋ] n.(总称)衣服 被褥

cloud [klaud] n.云 云状物 云状的烟 阴影 vt.使…模糊 以云遮敝 笼罩 使…混淆 毁坏 vi.脸色显得阴沉

cloudy ['klaudi] adj.多云的 云一般的

club [klʌb] n.俱乐部 夜总会 击棍 扑克牌中的梅花 vt.用棍棒打 集合 vi.集合 分担花费 泡夜总会

clue [klu:] n.线索 暗示 提示 vt.提示

clumsy ['klʌmzi] adj.笨拙的 笨重的 愚笨的 不得体的

coach [kəutʃ] n.长途公共汽车 大巴 教练 (火车)客车车厢 四轮马车 经济舱 vt. & vi.训练 指导

coal [kəul] n.煤 煤块 木炭 运煤工 vt.烧成炭 供应煤 vi.装煤 加煤

coarse [kɔ:s] adj.粗的 粗糙的 非精制的 粗俗的

coast [kəust] n.海岸 海滨(地区) 滑坡 vt.沿海岸而行 vi.下滑

coat [kəut] n.外套 上衣 表皮 动物皮毛 一层 vt.外面覆盖 盖上毛衣

cock [kɔk] n.公鸡 雄禽 旋塞 (男子间友好的互称用语)老兄 阴茎 vt.立起 竖起 vi.昂首阔步地走

code [kəud] n.码 密码 法规 准则 法典 代码 vt.把...编码 制成法典

coffee ['kɔfi] n.咖啡 咖啡茶

coil [kɔil] n.(一)卷 线圈 骚动 【电】线圈 绕组 vt. & vi.盘绕 卷

coin [kɔin] n.硬币 金钱 货币 铸造(硬币) vt.制造钱币 创造 adj.与硬币有关的

cold [kəuld] adj.冷的 冷淡的 漠然的 客观的 n.冷 感冒 低温 adv.唐突地 无回旋余地地 即兴地

collapse [kə'læps] vi.倒坍 崩溃 瓦解 折叠 vt.倒塌 缩减 n.崩溃 倒塌 暴跌

collar ['kɔlə] n.衣领 项圈 vt.抓住 为...戴上项圈

colleague ['kɔli:g] n.同事 同僚

collect ['kɔlekt,kə'lekt] vt.收集 推论 接走 vi.收款 聚集 积累 adj. & adv.对方付费(的)

collection [kə'lekʃən] n.搜集 收集 收藏品 小组 集合

collective [kə'lektiv] adj.集体的 共同的 集合性的 n.集体

college ['kɔlidʒ] n.学院 学校 社团 枢机主教团 大学

collision [kə'liʒən] n.碰撞 冲突

colonel ['kə:nl] n.陆军上校 中校

colony ['kɔləni] n.侨民 殖民地 侨居地 聚居(地)

color ['kʌlə] n.颜色 彩色 颜料 气色 风格 vt.把...涂颜色 粉饰 vi.脸红

column ['kɔləm] n.柱 支柱 圆柱 柱形物 专栏 [计算机]列

comb [kəum] n.梳子 梳状物 鸡冠 蜂巢 vt.梳理 清理 搜查 用梳子 vi.搜寻

combination [.kɔmbi'neiʃən] n.结合 联合 化合 联合体

combine [kəm'bain] vt.使结合 兼有 融合 联结 收割 混合谷物 vi.结合 联合 n.集团 联合收割机

come [kʌm] vi.来 来到 达到 出现 发生 进入 vt.接近 扮演

comfort['kʌmfət] n.舒适 安慰 安慰者 vt.安慰

comfortable ['kʌmfətəbl] adj.舒适的 安慰的 充裕的

command [kə'mɑ:nd] vt.命令 指挥 控制 支配 n.命令 指挥 控制 [计算机]DOS命令：引用辅助命令处理器

commander [kə'mɑ:ndə] n.司令官 指挥员

comment ['kɔment] n.评论 意见 注释 闲话 v.注释 评论

commerce ['kɔmə:s] n.商业 贸易 社交

commercial [kə'mə:ʃəl] adj.商业的 商品化的 n.商业广告

commission [kə'miʃən] n.委任(状) 任官令 委员会 佣金 犯罪行为 vt.委任 委托 使服役

commit [kə'mit] vt.犯(错误) 干(坏事) 委托(托付) 作...事 承诺 [计算机]委托

committee [kə'miti] n.委员会 全体委员

common ['kɔmən] adj.普通的 共同的 平常的 n.平民 普通 公地 公园

commonly['kɔmənli] adv.普通地 一般地 通常地

communicate [kə'mju:nikeit] vi.通讯 传达 传播 vi.交流 沟通 联通

communication [kə.mju:ni'keiʃən] n.通讯 传达 交通 信息 [计算机]通信

communism ['kɔmjunizəm] n.共产主义

Communism ['kɔmjunizəm] n.共产主义学说 共产主义制度

communist ['kɔmjunist] n.共产党员 共产主义者 adj.共产主义的

community [kə'mju:niti] n.社区 团体 社会 公社 群落(生)

companion [kəm'pænjən] n.同伴 同事 共事者 伴侣 成对物品之一 (船的)甲板间扶梯(或扶梯的顶篷) vt. & vi.陪伴

company ['kʌmpəni] n.公司 商号 同伴 客人 一群 连队 vt.陪伴 vi.联合

comparative[kəm'pærətiv] adj.比较的 相对的 n.对手

compare [kəm'pɛə] vt. & vi.比较 对照 比作 比喻 n.比较 对照

comparison [kəm'pærisn] n.比较 对照 比似

compass ['kʌmpəs] n.罗盘 指南针 圆规 vt.图谋 包围 达成

compel [kəm'pel] vt.强迫 迫使屈服

compete [kəm'pi:t] vi.比赛 竞争 对抗

competent ['kɔmpitənt] adj.有能力的 胜任的 足够的 应该做的

competition [.kɔmpi'tiʃən] n.竞争 比赛

compile [kəm'pail] vt.编辑 编制 编译 编纂 搜集

complain [kəm'plein] vi.抱怨 拆苦 悲叹 控告 控诉

complaint [kəm'pleint] n.抱怨 怨言 控告

complete [kəm'pli:t] adj.完整的 完成的 彻底的 vt.完成 使圆满 使完美

completely [kəm'pli:tli] adv.十分 完全地 全然

complex ['kɔmpleks] adj.结合的 复杂的 n.复合体

complicate ['kɔmplikeit] vt.使复杂 使陷入 使错综 使起纠纷

complicated ['kɔmplikeitid] adj.复杂的 难懂的

component [kəm'pəunənt] n.组成部分 分 组件 元件 成份 adj.组成的 构成的

compose [kəm'pəuz] vt.组成 构成 创作 写作 作曲 使镇静 vi.创作

composition [.kɔmpə'ziʃən] n.构成 作品 写作 作文 著作 组织 合成物 成份

compound ['kɔmpaund,kɔm'paund] n. 化合物 混合物 复合词 adj. 复(混)合的 合成的 vt. 混合 调合 妥协 vi. 混合 妥协

comprehension [.kɔmpri'henʃən] n.理解 理解力 领悟

comprehensive [.kɔmpri'hensiv] adj.广泛的 理解的 综合的

compress ['kɔmpres,kəm'pres] vt.压紧 压缩 [计算机] 压缩

comprise [kəm'praiz] vt.包含 包括 构成

compromise ['kɔmprəmaiz] n.妥协 让步 和解 折衷 折衷案 vt.妥协处理 危害 vi.妥协 让步

compute [kəm'pju:t] vt.计算 估计 估算

computer[kəm'pju:tə] n.计算机 电脑

comrade ['kɑ:mræd] n.同志 亲密的同伴

conceal [kən'si:l] vt.把...隐藏起来 掩盖

concentrate ['kɔnsentreit] vt. & vi.集中 专心 聚集 浓缩 n.浓缩 精选

concentration[.kɔnsen'treiʃən] n.集中 专心 专注 浓缩 浓度

concept ['kɔnsept] n.概念 观念 设想

concern [kən'sə:n] n.关心(人或事) 忧虑 挂念 关系 公司 企业 vt.涉及 影响 关心

concerning [kən'sə:niŋ] prep.关于

concert ['kɔnsət] n.音乐会 演奏会 一致 和谐 vt.制定计划 通过协商达成一致 vi.合作

conclude [kən'klu:d] vt.推断出 使结束 缔结 断定 得出结论 vi.结束 作出决定

conclusion [kən'klu:ʒən] n.结论 推论 结尾

concrete ['kɔnkri:t] n.水泥 混凝土 具体物 adj.具体的 实在的 水泥的 vt. & vi.凝结 结合

condemn [kən'dem] vt.谴责 指责 判刑 官方宣称(某事物)有缺陷或不宜使用

condense [kən'dens] vt.压缩 使缩短 浓缩 摘要

condition [kən'diʃən] n.状况 状态 环境 条件 v.以...为条件 决定 支配 训练

conduct [kən'dʌkt] n.举止 行为 品行 指导 vi.引导 指挥 管理 vt.导电 传热

conductor [kən'dʌktə] n.售票员 (乐队)指挥 导体

conference ['kɔnfərəns] n.会议 讨论会

confess [kən'fes] vt.供认 承认 坦白 告白 忏悔

confidence ['kɔnfidəns] n.信任 信赖 信心 把握 adj.骗得信任的

confident ['kɔnfidənt] adj.确信的 自信的

confine [kən'fain] vt.限制 禁闭 n.边界 约束 范围 限制

confirm [kən'fə:m] vt.证实 肯定 批准 确定

conflict ['kɔnflikt] n.争论 冲突 斗争 战斗 矛盾 vi.争执 撞斗 冲突 [计算机] 冲突

confuse [kən'fju:z] vt.使混乱 混淆 狼狈 困惑

confusion [kən'fju:ʒən] n.混乱 骚乱 混淆 不确定状态

congratulate [kən'grætju.leit] vt.祝贺 向...道喜

congratulation [kən.grætju'leiʃən] n.祝贺 祝贺词

congress ['kɔŋgres] n.大会 国会 议会 集会

conjunction [kən'dʒʌŋkʃən] n.接合 连接 连接词 关联 (事件等的)同时发生

connect [kə'nekt] vt.连接 连结 联通 联系 使有联系

connection [kə'nekʃən] n.连接 联系 关系 亲戚 连贯性

conquer [.kɔŋkə] vt.征服 战胜 克服 破除 vi.得胜

conquest ['kɔŋkwest] n.攻取 征服 克服

conscience ['kɔnʃəns] n.良心 道德心 责任心 顾忌

conscious ['kɔnʃəs] adj.意识到的 神志清醒的 意识到的 自觉的 有意的 n.意识

consciousness ['kɔnʃəsnis] n.意识 觉悟 知觉 自觉

consent [kən'sent] n.同意 赞成 许可 vi.同意 承诺

consequence ['kɔnsikwəns] n.结果 后果

consequently ['kɔnsikwəntli] adv.因此 因而 所以

conservation [.kɔnsə:'veiʃən] n.保存 保护 守恒 防止流失 守恒 保护自然资源

conservative [kən'sə:vətiv] adj.保守的 守旧的 n.保守的人 保守派(党)

consider [kən'sidə] vt.认为 考虑 思考 关心

considerable [kən'sidərəbl] adj.相当大的 重要的 可观的

considerate [kən'sidərit] adj.考虑周到的 体谅的

consideration [kənsidə'reiʃən] n.考虑 思考 体贴 考虑因素 敬重 意见

consist [kən'sist] vi.由...组成 存在 一致 在于 n.组成

consistent [kən'sistənt] adj.坚持的 一贯的 始终如一的 一致的

constant ['kɔnstənt] adj.经常的 永恒的 不变的 n.常数 恒量

constitution [.kɔnsti'tju:ʃən] n.章程 体质 构造 组织 宪法 体格

construct ['kɔnstrʌkt,kən'strʌkt] vt.建造 建设 构筑 绘制 想出 n.构想 建造物

construction [kən'strʌkʃən] n.建造 建设 建筑 建筑物 结构 构造

consult [kən'sʌlt] vt.向...请教 查阅 商讨

consume [kən'sju:m] vt.消耗 消费 消灭 毁灭 吃、喝、挥霍 vi.耗尽生命 被烧毁

consumption [kən'sʌmpʃən] n.消耗量 消耗 消费

contact ['kɔntækt] vt.使接触 与...联系 n.接触 联系 联系人 adj.接触的

contain [kən'tein] vt.包含 容纳 等于 克制 抑制 vi.自制

container [kən'teinə] n.容器 集装箱

contemporary [kən'tempərəri] adj.当代的 同时代的 同时的 现代的 n.同时代的人

contempt [kən'tempt] n.轻蔑 藐视 受辱

content ['kɔntent,kən'tent] n.内容 目录 容量 adj.满足的 满意的 vt.使...满足 使...安心

contest ['kɔntest,kən'test] vt.争夺 争取 辩驳 竞赛 vi.奋斗 n.竞赛 比赛

continent ['kɔntinənt] n.大陆 陆地 洲 (the Continent)欧洲大陆

continual [kən'tinjuəl] adj.不断的 连续的 频繁的

continue [kən'tinju:] vt. & vi.继续 连续 维持 延伸

continuous [kən'tinjuəs] adj.连续不断的 连续的 继续的 连绵不断的 持续的

contract ['kɔntrækt,kən'trækt] n.契约 合同书 合同 婚约 v.缩短 缩小 vt.订约

contradiction [.kɔntrə'dikʃən] n.矛盾 不一致 否认 反驳

contrary ['kɔntrəri] adj.相反的 截然不同的 n.相反 反面 adv.相反(地)

contrast ['kɔntræst,kən'træst] n.对比 对照 悬殊 差别 v.对比 成对照 [计算机] 反差

contribute [kən'tribju:t] vt.捐献 捐助 投稿 vi.投稿 贡献 是原因之一

control [kən'trəul] vt.控制 克制 掌管 支配 n.克制 控制 管制 操作装置

convenience [kən'vi:njəns] n.便利 方便 便利设施 厕所 适宜 方便的时间 舒适

convenient [kən'vi:njənt] adj.便利的 近便的 方便的

convention [kən'venʃən] n.习俗 惯例 公约 大会 协定 [计算机] 约定

conventional [kən'venʃənl] adj.普通的 习惯的 传统的 惯例的 常规的

conversation [.kɔnvə'seiʃən] n.会话 谈话 非正式会谈

conversely [kən'və:sli] adv.相反地

conversion [kən'və:ʃən] n.转变 转化 改变 改变信仰 换位

convert ['kɔnvə:t,kən'və:t] vt.使转变 使改变 倒置 n.皈依者 改宗者

convey [kən'vei] vt.传送 运送 传播 转移 [计算机] 输送

convince [kən'vins] vt.使确信 使信服 说服 使...相信

cook [kuk] vt.烹调 煮 编造 虚构 烧(菜) vi.烧菜 发生 做好 n.厨师 烧菜

cool [ku:l] adj.凉的 冷静的 冷色系的 adv.冷静地 n.凉快 vt. & vi.(使)变凉 (使)冷静

cooperate [kəu'ɔpəreit] vi.合作 协作 配合

coordinate [kəu'ɔ:dneit,kəu'ɔ:dnit] vt. 使协调 调节 整合 使一致 [计算机] 坐标 n. 同等的人物 (色调 图案 样式等)配套服装 adj. 同等的 等位的 (大学)男女分院制的

cope [kəup] vi.对付 应付 对抗 n.长袍 斗篷状物 vt.笼罩

copper ['kɔpə] n.铜 铜币 铜制器 有紫铜色翅膀的小蝴蝶 警官 vt.镀铜

copy ['kɔpi] n.抄件 副本 模仿 一册 稿件 vt.抄写 抄袭 复制 复印 vi.复制 抄写 n.[计算机] DOS命令：复制组合文件 拷贝

cord [kɔ:d] n.细绳 粗线 索 束缚 vt.用绳索绑

cordial ['kɔ:djəl] adj.真诚的 诚恳的 热忱的 兴奋的 n.兴奋剂 补品

core [kɔ:] n.果实的心 核心 要点 vt.挖去果核

corn[kɔ:n] n.谷物 谷粒 陈词滥调 鸡眼 (英)小麦 vt.使成颗粒状 用盐水腌制 用谷物喂 vi.变成颗粒

corner ['kɔ:nə] n. 角 角落 转角 窘境 犄角 边远地区 偏僻住所 [计算机] 角点 vt. 使...走投无路 使...陷入绝境 垄断 转弯adj.位于角落

corporation [.kɔ:pə'reiʃən] n.公司 法人 集团 企业 社团

correct [kə'rekt] adj.正确的 vt. & vi.改正 纠正

correction [kə'rekʃən] n.改正 纠正 修改 改正的地方

correspond [.kɔris'pɔnd] vi.相符合 相当 通信

correspondent [.kɔri'spɔndənt] n.通信者 通讯记者 通讯员 adj.与...一致的 相应的

corresponding [.kɔri'spɔndiŋ] adj.相应的 符合的 一致的 相同的 相当的

corridor ['kɔridɔ:] n.走廊 回廊 通路

cost [kɔst] n.价格 代价 成本 费用 vt.花费 使付出 vi.值 付出 [计算机] 费用

costly ['kɔstli] adj.昂贵的 价值高的 代价高的

cottage ['kɔtidʒ] n.村舍 小屋

cotton ['kɔtn] n.棉 棉花 棉线 棉布 vi.向…讨好 明白 领悟

cough [kɔf] vt. & vi.咳 咳嗽 n.咳嗽

could [kud] aux.v.(can的过去式)

council ['kaunsil] n.理事会 委员会 商议

count [kaunt] vt.计算 视为 依赖 vi.数 计数 n.计数

counter ['kauntə] n.柜台 计数器 计算者 [计算机] 计数器 adj.相反的 adv.与…相反地 vt.反对 反击 vi.反对 反击

country ['kʌntri] n.国家 国土 农村 adj.国家(的) 农村(的)

countryside ['kʌntri'said] n.乡下 农村

county ['kaunti] n.英国的郡 美国的县

couple ['kʌpl] n.夫妇 (一)对 几个 数个 v.加倍 成双 连结

courage ['kʌridʒ] n.勇气 胆量 胆识

course[kɔ:s] n.课程 讲座 过程 路线 一道菜

court [kɔ:t] n.法院 法庭 庭院 奉承 球场 v.献殷勤 追求 招致危险

cousin ['kʌzn] n.堂(或表)兄弟(姐妹)

cover ['kʌvə] vt.盖 覆盖 包括 涉及 包含 n.盖子 封面 表面

cow [kau] n.母牛 奶牛 母兽

coward ['kauəd] n.懦夫 胆怯者 adj.胆小的

crack [kræk] n.裂缝 裂纹 声变 vi.爆裂 破裂 声变 打开

craft [krɑ:ft] n.工艺 手艺 行业 航空器

crane [krein] n.起重机 摄影升降机 鹤 v.引颈 伸长(脖子) vt.伸长(脖子等)

crash [kræʃ] vi.碰撞 坠落 猛使...破碎 迫降 n.碰撞 猛撞 轰隆声 崩溃 破产 垮台

crawl [krɔ:l] vi.爬 爬行 n.爬行 匍匐而行

crazy ['kreizi] adj.疯狂的 荒唐的

cream [kri:m] n.奶油 乳酪 乳脂 奶油色 面霜 精华

create [kri'eit] vt.创造 引起 产生

creative [kri'eitiv] adj.创造性的 创作的

creature ['kri:tʃə] n.生物 动物 家畜 人

credit ['kredit] n.信用贷款 信用 荣誉 学分 v.归功于 赞颂 信任

creep [kri:p] vi.爬行 缓慢地行进 蔓延 n.爬 徐行 蠕动

crew [kru:] n.全体船员 全体乘务员 vi.一起工作

cricket ['krikit] n.板球 蟋蟀 vi.打板球

crime [kraim] n.罪 罪行 犯罪 羞耻

criminal ['kriminəl] n.犯人 罪犯 刑事犯 adj.犯罪的 刑事的 无耻的

cripple ['kripl] n.跛子 残废的人 v.使...跛 使...成残废 削弱

crisis ['kraisis] n.危机 存亡之际

critic ['kritik] n.批评家 评论家 爱挑剔的人

critical ['kritikəl] adj.决定性的 批评的 危险的 挑剔的 临界的

criticism ['kriti.sizəm] n.批评 批判 评论

criticize ['kritisaiz] vt.批评 评论 非难 吹毛求疵 vi.批评

crop [krɔp] n.农作物 庄稼 收成 一熟 一群 平头 vt.割掉 修剪 剪短 收割 使谷物生长 vi.啃青草 种植谷物

cross [krɔs] vt.穿过 划叉剔除 画横线于 vi.穿越 prep.穿过 n.十字架 十字架形物件 交叉 交叉路 adj.生气的 交叉的

crow [krəu] n.鸦 乌鸦 啼叫 欢叫 vi.啼 啼叫 报晓 欢叫 vt.洋洋夸口 自鸣得意

crowd [kraud] n.群 大众 一伙人 一堆 vi.拥挤 挤满 挤进 vt.挤满 将...塞进 催促

crown [kraun] n.王冠 冕 王权 顶点 花冠 vt.使...成王 加冕 居...之顶 为牙齿镶上假齿冠 vi.火迅速烧到树的顶端

crude [kru:d] adj.简陋的 天然的 未加工的 粗鲁的 n.原油

cruel ['kru:əl] adj.残忍的 残酷的 引起痛苦的

crush [krʌʃ] vt.压碎 碾碎 镇压 拥挤的人群 迷恋 关键时刻 果汁 榨汁酿酒 vt.压破 征服 挤入 镇压 vi.被压碎 挤

crust [krʌst] n.面包皮 硬外皮 外壳 坚硬的外壳 vt.盖以硬皮 vi.结硬皮

cry [krai] vi.哭 哭泣 叫喊 强烈要求 vt.大声叫喊 宣传 哭得... n.叫 哭 喊 口号 叫卖声 普遍意见

crystal ['kristl] n.水晶 结晶体 晶粒 adj.清澈透明的 晶体的

cube [kju:b] n.立方体 立方 vt.求...的立方 adj.立方的

cubic ['kju:bik] adj.立方体的 立方的

cucumber ['kju:kəmbə] n.黄瓜

cultivate ['kʌltiveit] vt.耕 种植 培养 栽培 结交(朋友) 促进增长 教养

culture ['kʌltʃə] n.文化 文明 教养 种植 vt.培养

cunning ['kʌniŋ] adj.狡猾的 狡诈的 有眼光的 精巧的 可爱的 n.狡诈 灵巧

cup [kʌp] n.杯子 (一)杯 奖杯 v.(使)成杯形 vt.使成杯形

cupboard ['kʌbəd] n.碗柜 碗碟橱 食橱

cure [kjuə] vt.医治 消除 vt. &vi.治疗 治愈 改正 n.治疗 治愈 治疗法

curiosity [.kjuəri'ɔsiti] n.好奇 好奇心 珍品

curious ['kjuəriəs] adj.好奇的 奇特的 稀奇古怪的

curl [kə:l] n.卷毛 螺旋 vi.卷曲 弄卷 弯曲

current ['kʌrənt] adj.当前的 现在的 最近的 通用的 流通的 n.(水、气、电)流 趋势

curse [kə:s] n.诅咒 咒骂 天谴 祸端 行经 vt.咒骂 诅咒 引起灾祸 vi.诅咒

curtain ['kə:tn] n.帘 窗帘 幕(布) 遮蔽物 vt.用窗帘装饰 遮蔽

curve [kə:v] n.曲线 弯 弯曲 曲球 vt.弄弯 使...弯曲 vi.弯曲

cushion ['kuʃən] n.垫子 坐垫 橡皮软垫 缓冲 vt.加垫褥 放在垫子上 保护 减缓 为...装垫子

custom ['kʌstəm] n.习惯 风俗 海关

customer ['kʌstəmə] n.顾客 主顾 家伙

cut [kʌt] vt. 切 割 剪 减少 停止 修剪 vi. 切割 快速移动 [计算机] 剪掉 n. 切口 割伤 降低 份额 删节 割下的一块肉

cycle ['saikl] n自行车 循环 周期 长期 圈 套曲 vt.环行 骑自行车 vi.使环行

## D

daily ['deili] adj.每日的 按天计算的 日常的 adv.每日 天天 n.日报 (复)工作样片

dairy ['dɛəri] n.牛奶场 乳制品 乳品店 adj.牛奶的

dam [dæm] n.水坝 水堤 障碍物 vt.筑坝 抑制(情感）等

damage ['dæmidʒ] vt.损害 毁坏 n.(pl.)赔偿 损坏 毁坏 损害

damp [dæmp] adj.潮湿的 有湿气的 n.毒气 湿气 丧气 vt.呛 抑制 使潮湿 vi.衰减

dance [dɑ:ns] vi.跳舞 摇晃 雀跃 vt.使跳跃 使某人以跳舞的方式进入 n.舞 跳舞 舞蹈 舞会 舞步

danger ['deindʒə] n.危险 危险事物

dangerous ['deindʒərəs] adj.危险的 不安全的 引起危险的

dare [dɛə] vt. & vi. & aux.v.敢 竟敢 敢于 勇于面对 n.敢 挑战

daring ['dɛəriŋ] adj.大胆的 勇敢的 n.冒险精神

dark [dɑ:k] adj.暗的 黑色的 n.黑暗 暗处

darling ['dɑ:liŋ] n.亲爱的人 可爱的人 可爱的物 宠儿 adj.可爱的 亲爱的

dash [dæʃ] vt.使猛撞 猛掷 泼溅 冲撞 匆忙 溅 vi.猛冲 猛撞 n.猛冲 冲撞 破折号 少量 短跑 冲劲

data ['deitə] n.(datum的复数)数据 资料

date [deit] n.日期 约会 枣椰子 vt.约会 定...日期 vi.追溯到 n.[计算机] DOS命令：显示或设置系统日期

daughter['dɔ:tə] n.女儿 adj.与女儿有关的 女性直系后代的

dawn [dɔ:n] n.黎明 破晓 开端 vi.破晓 逐渐明白

day [dei] n.(一)天 白昼 白天

daylight ['deilait] n.白昼 日光 黎明

dead [ded] adj.死的 无生命的 萧条的 精疲力竭的 用完的 无趣的 adv.完全地 突然地 直接地 n.死者 死

deadly ['dedli] adj.致命的 致死的 死一般的 adv.非常地 如死一般地

deaf [def] adj.聋的 不愿听的

deal [di:l] n.买卖 待遇 交易 协定 份量 vt.(dealt dealt[delt])处理 应付 分配 给予

dear [diə] adj.严重的 珍贵的 亲爱的 昂贵的 衷心的 adv.高价地 热爱地 n.亲爱的人 可爱的人 int.啊 哎呀

death [deθ] n.死 死亡 灭亡 死神

debate [di'beit] n. &vi.争论 辩论 商讨 vt.争论 思考

debt [det] n.债 债务 欠债 负债状态 义务 罪 过失

decade ['dekeid] n.十年 十年期

decay [di'kei] vi.腐烂 衰败 n.腐烂 衰退

deceit [di'si:t] n.欺骗 欺诈 诡计 不诚实

deceive [di'si:v] vt.欺骗 蒙蔽 行骗 vi.行骗

December [di'sembə] n.十二月

decent ['di:snt] adj.正派的 体面的 有分寸的 得体的 相当好的

decide [di'said] vt. & vi.决定 决心 解决 作出抉择

decision [di'siʒən] n.决定 决心 果断 决策

deck [dek] n.甲板 舱面 层面 桥面 楼层 vt.装饰

declare [di'klɛə] vt. & vi.断言 声明 表明 宣布 申报

decorate ['dekəreit] vt.装饰 装璜 修饰 授予某人奖章或其他奖状

decrease ['di:kri:s,di:'kri:s] vi. &n.减少 减少

deduce [di'dju:s] vt.演绎 推论 推断 溯源

deed [di:d] n.行为 事迹 功绩 (尤指房产)契约 证书 vt.通过行为转化

deep [di:p] adj.深的 有...深的 深奥的 纵深的adv.深深地 晚地 n.深处

deepen ['di:pən] vt.加深 使...加深 使...强烈 vi.深化 加深 变浓

deer [diə] n.鹿

defeat [di'fi:t] vt.战胜 击败 挫败 n.败北 失败

defect [di'fekt] n.缺点 缺陷 欠缺 vi.背叛

defence [di'fens] n.防御 防务 辩护 答辩

defend [di'fend] vt. & vi.防护 辩护 防卫 保卫 防守

define [di'fain] vt.给...下定义 限定 解释

definite ['definit] adj.明确的 肯定的 一定的

definitely ['definitli] adv.一定地 明确地 确切地

definition [.defi'niʃən] n.定义 释义 定界 阐释 清晰度

degree [di'gri:] n.程度 度 度数 学位

delay [di'lei] vt.推迟 耽搁 延误 vi.耽搁 n.耽搁 迟滞

delete [di'li:t] vt.删除 擦掉

delegation [.deli'geiʃən] n.代表团 委派

delicate ['delikit] adj.纤细的 易碎的 精美的 微妙的 美味的 脆弱的 敏锐的 n.精美的东西

delicious [di'liʃəs] adj.美味的 可口的 怡人的

delight [di'lait] n.快乐 快乐 vt.使高兴 vi.给与乐趣

deliver [di'livə] vt.投递 递送 送交 发表 表达 释放 交付 接生 履行 vi.达成

delivery [di'livəri] n.投递 交付 分娩

demand [di'mɑ:nd] vt.要求 需要 询问 n.要求 需求

democracy [di'mɔkrəsi] n.民主 民主制 民主国家

democratic [.demə'krætik] adj.民主的 民主政体的 大众的 平等的

demonstrate ['demənstreit] vt.说明 论证 证明 表露 示范 演示 vi.示威

dense[dens] adj.密集的 浓厚的 愚钝的 不易看透的

density ['densiti] n.密集 稠密 密度 透明度

deny [di'nai] vt.否定 拒绝相信 拒绝 反对 克制

depart [di'pɑ:t] vi.离开 起程 出发 死亡 脱轨 vt.离开

department [di'pɑ:tmənt] n.部 司 局 处 系 部门 省

departure [di'pɑ:tʃə] n.离开 出发 起程 分歧

depend [di'pend] vi.依靠 依赖 相信 指望

dependent [di'pendənt] adj.依靠的 依赖的 附属的 悬垂的 n.受经济援助者

deposit [di'pɔzit] vt.使沉淀 存放 堆积 vi.沉淀 n.存款 定金 堆积物

depress [di'pres] vt.使沮丧 按下 降低 [计算机] 降低

depth [depθ] n.深度 深厚 深处 强度 诚挚

derive [di'raiv] vt.取得 得自 起源 引申于 vi.起源

descend [di'send] vt. & vi.降 传 降临 下来 下降 下倾

describe [di'skraib] vt.形容 描写 描绘 描述 画(尤指几何图形)

description [di'skripʃən] n.描写 描述 说明书 作图 形容 种类

desert ['dezət,di'zə:t] n.沙漠 荒地 应得(的惩罚或奖励) vt.离弃 擅离 遗弃 vi.擅离职守 adj.沙漠的 荒凉的

deserve [di'zə:v] vt.应受 值得 vi.应该得到

design [di'zain] vt.设计 想像 计划 vi.计划 设计 n.设计 图样

desirable [di'zaiərəbl] adj.值得相望的 可取的 值得有的 令人满意的 有吸引力的 n.有吸引力的人

desire [di'zaiə] vt.要求 向往 请求 渴望 n.愿望 欲望 情欲

desk [desk] n.书桌 办公桌

despair [di'spɛə] n.绝望 失望 vi.绝望 失望

desperate ['despərit] adj.拼死的 绝望的 不顾一切的 极想得到的

despise [di'spaiz] vt.鄙视 蔑视

despite [di'spait] prep.不管 尽管 不顾 v.憎恨 轻视

destination [.desti'neiʃən] n.目的地 终点 目标 景点

destroy [dis'trɔi] vt. & vi.破坏 消灭 打破 [计算机] 撤消

destruction [di'strʌkʃən] n.破坏 毁灭 消灭 破坏者

detail ['di:teil] n.细节 详情 部分 分派 枝节 零件 vt.详述 选派 用小元素装饰 vi.画细节

detect [di'tekt] vt.察觉 发觉 侦察 发现 探测信息 vi.当侦探

detection [di'tekʃən] n.察觉 发觉 侦察 探测

determination [di.tə:mi'neiʃən] n.决心 (正式)决定 确定 规定 测定 定位

determine [di'tə:min] vt.决定 查明 决心 [计算机] 确定

develop[di'veləp] vt.发展 形成 开发 冲洗照片 vi.进步

development [di'veləpmənt] n.发展 开发 生长 显影 开发区

device [di'vais] n.器械 装置 设备 设计 策略

devil ['devl] n.魔鬼 恶魔 坏人 难对付的东西 困难 小尘暴 vt.加料 惹恼

devise [di'vaiz] vt.设计 发明 遗赠 n.遗赠(的财产)

devote [di'vəut] vt.将...奉献 致力于 投入于 献身

dew [dju:] n.露 露水

diagram ['daiəgræm] n.图解 图表 简图 vt.用图解法表示

dial ['daiəl] n.钟面 拨号盘 刻度盘 针面 转盘 vt.拨 用仪表测量 操作仪表 vi.拨号 用仪表测量 [计算机] 拨号

dialect ['daiəlekt] n.方言 土语 地方话

dialog ['daiəlɔg] n.对话 对白 (美)=dialogue

diameter [dai'æmitə] n.直径

diamond ['daiəmənd] n.金钢石 钻石 菱形 像钻石的物质 纸牌的方块 棒球内场 vt.用钻石装饰 adj.60或75周年纪念的

diary ['daiəri] n.日记 日记簿

dictate ['dikteit,dik'teit] vt. & vi.口授 命令 听写 n.命令 指挥 指令

dictation [dik'teiʃən] n.口授笔录 听写

dictionary ['dikʃəneri] n.词典 字典 辞典

die [dai] vi.死 死亡 灭亡 n.金属模子 印模 骰子

differ ['difə] vi.不同(意见) 相异 不一致

difference ['difərəns] n.差别 差 分歧 vt.引起区别

different ['difərənt] adj.差异的 各种的 不同的 与众不同的

difficult ['difikəlt] adj.困难的 难对付的 (人)难取悦的 难相处的

difficulty ['difikəlti] n.困难 争议 反对 麻烦 难事 困境

dig [dig] vt.掘 挖 翻土 采掘

digest [dai'dʒest] vt.消化 领会 理解 vi.消化 n.文摘 摘要

digital ['didʒitəl] adj.数字的 计数的 手指的 数码的 电子的

diligent ['dilidʒənt] adj.勤勉的 勤奋的 用功的

dim [dim] adj.昏暗的 暗淡的 朦胧的 模糊的 笨的 vt.使暗淡 使失去光泽 vi.变暗 n.车头灯的短焦距光

dimension [di'menʃən] n.尺寸 尺度 面积 次元 容积 维度 范围 方面 vt.标尺寸 形成...尺寸

dinner ['dinə] n.正餐 主餐 宴会 晚宴

dip [dip] vt.浸 蘸 给...洗药浴 vi.浸一浸 下降 向下倾斜 翻阅 提取(款项) n.游 下降 减少 低洼处 调味酱 药浴液

direct [di'rekt] adj.直接的 直率的 笔直的 坦白的 vt.指挥 命令 负责 vi.指导 指出

direction [di'rekʃən] n.方向 方位 指导 用法说明 趋势

directly [di'rektli] adv.直接地 立即 完全

director [di'rektə, dai'rektə] n.指导者 理事 董事 经理 主管 导演

dirt [də:t] n.尘 土 污物 污垢 泥土 污秽的言行 卑鄙的人 堕落 矿渣

dirty ['də:ti] adj.脏的 下流的 肮脏的 色情的 卑鄙的 adv.卑鄙地 vt.使...变脏 vi.弄脏

disable [dis'eibl] vt.使无能 使伤残 使...失去能力

disadvantage [.disəd'væntidʒ] n.不利 不利地位 不利条件 损害 损失

disagree [.disə'gri:] vi.有分歧 不一致 不适应 不适宜

disappear [.disə'piə] vi.不见 失踪 消失 灭绝 vt.引起消失

disappoint [.disə'pɔint] vt. & vi.使失望 使受挫折

disaster [di'zɑ:stə] n.灾难 灾祸 天灾

disk [disk] n.圆盘 唱片 磁盘 光盘 花盘 盘形物 [计算机] 磁盘 vt.用圆盘耙耙地

discard ['diskɑ:d,dis'kɑ:d] vt.丢弃 抛弃 遗弃 vi.丢牌 n.丢牌 废牌

discharge [dis'tʃɑ:dʒ] vt. vi.释放 排出 放出 解雇 解除 放电 清偿债务 n.释放 卸货 放电 一股 一阵 开火 发射 准许离开

discipline ['disiplin] n.纪律 训练 惩罚 学科 vt.训练 惩罚

disclose [dis'kləuz] vt.揭开 揭发 透露

discourage [dis'kʌridʒ] vt.使泄气 使灰心 阻碍

discover [dis'kʌvə] vt. & vi.发现 暴露 显示

discovery [dis'kʌvəri] n.发现 被发现的事物

discuss [di'skʌs] vt.讨论 谈论 论述

discussion [di'skʌʃən] n.讨论 谈论 辩论 论述

disease [di'zi:z] n.病 疾病 病害 弊端

disguise [dis'gaiz] vi.隐瞒 掩埋 vt.假装 假扮 掩饰 n.假装 假面目 伪装物

disgust [dis'gʌst] n.厌恶 憎恶 嫌恶 vt. & vi.令人厌恶

dish [diʃ] n.碟 盘子 菜肴 凹形 vt.装盘 提出 揭露 挖空 vi.闲谈

dishonour [dis'ɔnə] n.不光彩 丢脸的人 耻辱 拒付 vt.使丢脸 侮辱 拒付 =dishonor

dislike [dis'laik] vt. &n.不喜爱 厌恶 反感

dismiss [dis'mis] vt.不再考虑 解雇 解散 开除 逃避 (法律)驳回

disorder [dis'ɔ:də] n.混乱 杂乱 骚乱 vt.扰乱

display [di'splei] vt.陈列 展览 显示 炫耀 vt.显示 表现 夸示

disposal [di'spəuzəl] n.丢掉 处理 销毁 菜碾碎器

dispose [di'spəuz] vi.去掉 丢掉 销毁 vt.倾向于 处置

displease [dis'pli:z] vt.使不愉快 使生气

dispute [di'spju:t] vi.争论 争执 vt.争论 质疑 争夺 抵抗 n.争论 争吵

dissatisfy [dis'sætisfai] vi.使不满 使不平

dissolve [di'zɔlv] vt.使溶解 解散 消除 解决(问题) 使沮丧 vi.消散 溶解 解散 液化 变得模糊 n.叠化画面

distance ['distəns] n. 距离 间距 远处 (人际关系的) 疏远 vt. 使疏远 把 (对手) 甩在后面 与...保持距离

adj. ( 通过电子手段 ) 远程 ( 教育 )

distant ['distənt] adj.在远处的 疏远的 疏远的 远房的 陌生的 远行的

distinct [di'stiŋkt] adj.与其他不同的 不同的 明显的 (from)独特的 清楚的

distinction [di'stiŋkʃən] n.差别 不同 区分 对比 杰出

distinguish [di'stiŋgwiʃ] vt.区别 辨别 认别 特别关注 vi.区别

distress [di'stres] n.忧虑 悲伤 不幸 苦恼 贫困 忧郁 恶化 扣押财物 vt.使...苦恼 仿造 adj.亏本出售的

distribute [di'stribjut] vt. & vi.分发 分送 分布 分配

distribution [.distri'bju:ʃən] n.分发 分配 分布 散布

district ['distrikt] n.区 地区 区域 行政区 vt.划分区域 划分地区

disturb [dis'tə:b] vt. & vi.打扰 扰乱 弄乱 妨碍 使...不安

ditch [ditʃ] n.沟 沟渠 渠道 vt.坠入沟中 摆脱 飞机(在海上)迫降 vi.挖一条沟渠 飞机紧急降落

dive [daiv] vi.跳水 潜水 俯冲 vt.跳水 俯冲 vi.把...伸入

diverse [dai'və:s] adj.不一样的 相异的 不同的 多种多样的

divide [di'vaid] vt.分 分配 分开(组) vi.除 分开

division [di'viʒən] n.分 分配 除法 划分 分开 部门 师(军队)

divorce [di'vɔ:s] n.离婚 离异 vi.离婚 与...办离婚

do [du:] aux.v.vt. & vi.做 干 办

dock [dɔk] n.船坞 码头 (造)修船厂 尾巴的骨肉部分 被告席 vt.进港 (减)剪短 对接 vi.进港 对接

doctor ['dɔktə] n.医生 医师 博士 vt.医治 篡改 对...做手脚 vi.用药

document ['dɔkjumənt,'dɔkjument] n.公文 文件 证件 [计算机] 文档 vt.记载 用文件等证明

dog [dɔg] n.狗 犬 犬科动物 卑鄙的人 vt.尾随 跟踪

dollar ['dɔlə] n.美元(货币单位)

domestic [də'mestik] adj.本国的 国内的 家庭的 驯养的 n.家仆 佣人

donkey ['dɔŋki] n.驴 笨蛋

door [dɔ:] n.门 通道 一家

dorm [dɔ:m] n.(集体)宿舍

dormitory ['dɔ:mitri] n.集体寝室 宿舍

dose [dəus] n.剂量 用量 一剂 vt.配

dot [dɔt] n.点 圆点 少量 小东西 vt.作小点记号 点缀 vi.作小点记号 [计算机] 点

double ['dʌbl] adj.两倍的 复制品 替身 双重角色的演员 反转 双的 双人房 adj.两倍的 双重的 骗人的 加量的 adv.两倍地 双人地 翻来覆去地 对折地 vt.使加倍 对折 握紧 做替身 vi.加倍 转身 兼任 替代演出

doubt [daut] n.怀疑 疑虑 不信任 vt.怀疑 不信任 vi.不确定

doubtful ['dautfəl] adj.难以预测的 怀疑的 疑心的 不确定的

doubtless ['dautlis] adv.无疑地 很可能 确定地 adj.无疑的 确定的

down [daun] adv. 向下 在下面 下跌 在南方 adj. 向下的 降低的 情绪低落 停止运行 n. 绒毛 汗毛 轻视 prep.沿着...而下

downstairs[.daun'stɛəz] adv.在楼下 向楼下 adj.楼下的 n.楼下

downward ['daunwəd] adj.向下的 adv.向下地

dozen ['dʌzn] n.一打 十二个 adj.一打的 十二个的

draft [dræft] n.草稿 草图 汇票 徵兵 vt.起草 征兵 选秀 adj.酝酿中的

drag [dræg] vt.拖 拉 拖曳 拖累 拖拉(动) vi.拖拉(动) 缓慢地行走 n.用来拖拉的东西 拖 拖累

dragon ['drægən] n.龙 凶暴的人 严厉的人

drain [drein] vt.排去 放水 耗尽 排出 排干 喝光 vi.变干 排出 排水 n.下水道 排水沟 消耗 耗竭

drama ['drɑ:mə] n.一出戏剧 剧本 戏剧 戏剧性情节 刺激

dramatic [drə'mætik] adj.引人注目的 戏剧的 给人深刻印象的 n.(pl.)戏剧作品 戏曲

dramatization [.dræməti'zeiʃən] n.编剧 改编成戏剧

draw [drɔ:] vt.(drew drawn) 拉 拖 挨近 提取 画 绘制 划 拨出 n.平局 抽签 拉 拽 吸引人的事物

drawer ['drɔ:ə] n.抽屉

drawing ['drɔ:iŋ] n.图画 素描 绘图 绘图技巧

dread [dred] n.畏惧 恐怖 可怕的人 可怕的事 vt.惧怕 恐惧 adj.可怕的 可怖的

dream [dri:m] n.梦 梦想 vt. & vi.做梦 想象

dress [dres] n.女服 童装 服装 vt. & vi.穿著 adj.正式的(衣服)

drift [drift] vi.漂流 漂泊 吹积 n.漂流 漂移 漂流物 观望 吹积物 趋势 vt.使漂流 把...吹积

drill [dril] n.钻头 钻孔机 钻子 播种机 操练 vi.钻孔 vt.训练 钻孔

drink [driŋk] vt. & vi.饮 喝 喝酒 n.饮料

drip [drip] vi.滴下 漏水 n.水滴 滴 点滴 乏味的人

drive [draiv] vt. & vi.驾驶 驱 驱使 驾进力 打入 n.驾车 快车道 推进力 驱使 路 街 [计算机] 驱动器

driver ['draivə] n.驾驶员 司机

drop [drɔp] vt.使落下 降低 减少 vi.落下 减少 减弱 n.滴 微量 落下

drought [draut] n.旱灾 干旱

drown [draun] vi.淹死 溺死 vt.淹死 沉浸于

drug [drʌg] n.药 药物 药材 毒品 vt.下药 使...服麻醉药 vi.吸毒

drum [drʌm] n.鼓 鼓声 鼓状物 圆桶 冰丘 vi.击鼓 作鼓声 引起兴趣 vt.召集入伍 逐出 (经过努力)推动... 连续击打

drunk [drʌŋk] adj.醉的 陶醉的 vbl.喝 喝酒

dry [drai] adj.干的 干燥的 干旱的 不行于色的 枯燥无味的 无酒的 口渴的 vt.把...弄干 vi.(使)变干

duck [dʌk] n. 鸭 雌鸭 鸭肉 板球零分 vt. 把…按入水中 低头弯身(以免被打中或看见) 躲避

vi. 迅速行进 飞快行走(以免被看见) 推委责任

due [dju:] adj.预期的 应给的 到期的 应有的 预定的 n.应得物

dull [dʌl] adj.枯燥的 不鲜明的 呆滞的 迟钝的 无趣的 钝的 暗的 vt.使变钝 vi.变钝

dumb [dʌm] adj.哑的 无言的 沉默的 愚蠢的 vt.使人沉默

dump [dʌmp] vt.倾卸 倾倒 倾销 丢弃 抛售 导出 vi.骤降 扔垃圾 n.垃圾场 堆放处

durable['djuərəbl] adj.耐久的 耐用的 n.(复)耐用品

duration [dju'reiʃən] n.持续 持久 持续时间 期间

during ['djuəriŋ] prep.在...期间

dusk [dʌsk] n.薄暮 黄昏 幽暗 vi.(天)变黑 变暗 vt.把…变黑变暗 adj.颜色暗沉的 昏暗的

dust [dʌst] n.尘土 灰尘 无价值之物 屈辱状态 vt.拂去灰尘 撒上粉状物体 大胜 vi.擦拭灰尘

duty ['dju:ti] n.职责 责任 义务 税 adj.值班的 作为一种义务的

dwelling ['dweliŋ] n.住处 寓所

dye [dai] vt.染 vi.染色 n.染料 染色

dying ['daiiŋ] adj.垂死的 临终的 n.死 死亡

dynamic [dai'næmik] adj.有活力的 动态的 (有)动力的 (有)力的 n.动力 力学

## E

each [i:tʃ] pron.各 各自 adj.每个 每 每一 adv.每个 int.每个 个人 个自

eager ['i:gə] adj.渴望的 热切的

eagle ['i:gl] n.鹰 vt.(高尔夫)鹰击

ear [iə] n.耳朵 听力 听觉 倾听 麦穗 vi.长出麦穗

early ['ə:li] adv.早 adj.早的 早期的

earn [ə:n] vt.赚得 挣得 获得 赚得 赢得 生利

earnest ['ə:nist] adj.认真的 诚恳的 重要的 n.真诚 重要的东西 定金 保证

earth [ə:θ] n.地球 陆地 地面 泥土 世间 大地 地球人 vt.赶入洞中 提取土壤 vi.躲入洞穴

earthquake ['ə:θkweik] n.地震 大震荡

ease [i:z] n.容易 舒适 安乐 n.安逸 悠闲 vt.缓和 使...安乐 使...安心 v.减轻 放松

easily ['i:zili] adv.容易地 舒适的 无疑地

east [i:st] n.东 东部 adv.在东方 向东方 adj.东方的

eastern ['i:stən] adj.东部的 东方的 朝东的

easy ['i:zi] adj.容易的 安逸的 不费力的 adv.慢慢地 当心地

eat [i:t] vt.吃 喝 消耗 vi.吃饭 侵蚀 n.食物

echo ['ekəu] n. 回声 回音 回波 反响 共鸣 vi. 重复 发回声 回响 仿效 vi. 回响 回荡 n. [计算机] DOS命令：禁止(或允许)批处理命令行显示屏幕上

economic [.i:kə'nɔmik] adj.经济的 经济学的 有利可图的 n.[-s]经济学(状况等)

economical [.i:kə'nɔmikəl] adj.节约的 节俭的 经济的 合算的 经济学的

economy [i'kɔnəmi] n.经济 节约 节省 adj.经济型的

edge [edʒ] n.边缘 边 刀口 vt.给...加上边 缓慢横过 使逐渐丧失 vi.徐徐前进

edition [i'diʃən] n.版 版本 版次

editor ['editə] n.编辑 编者 校订者

educate ['edjukeit] vt.教育 培养 训练

education [.edjukeiʃn] n.教育 培养 训导 教育学

effect [i'fekt] n.结果 效果 效力 影响 要旨 vt.招致 引起 实现 达到 n.(pl.)财产 动产

effective [i'fektiv] adj.有效的 有影响的

efficiency [i'fiʃənsi] n.效率 功效 效能

efficient [i'fiʃənt] adj.效率高的 有能力的 胜任的

effort ['efət] n.努力 努力的成果

egg [eg] n.蛋 鸡蛋 卵 (俚语)人

eight [eit] num.八 八个 第八

eighteen ['ei'ti:n] num.十八 十八个

eighth [eitθ] num.第八 n.八分之一

eighty ['eiti] num.八十 八十个

either ['i:ðə] pron.(两者)任何一个 adj.任一 两方的 adv.也 conj.或 也

elaborate [i'læbəreit] adj.复杂的 精心制作的 精细的 详尽的 精心的 v.详细地说明 用心地作 推敲

elastic [i'læstik] n.橡皮圈 有松紧的东西 松紧带 adj.有弹性的 灵活的 可变的

elbow ['elbəu] n.肘 肘部 弯管 急弯 扶手 v.用手肘推开 推挤

elder ['eldə] adj.年龄较大的 年长的 n.长者 长辈

elect [i'lekt] vt.选举 推选 选择 n.当选人 被选的人 上帝的选民 adj.被选的 选出的

election [i'lekʃən] n.选举 选择权 当选

electric [i'lektrik] adj.电的 电动的 令人激动的 鲜亮的 n.供电

electrical [i'lektrikəl] adj.电的 电气科学的 与电有关的

electricity [.ilek'trisiti] n.电 电学 电流

electron [i'lektrɔn] n.电子

electronic a.电子的

electronics [ilek'trɔniks] n.电子学 电子器件

element ['elimənt] n.成分 要素 元素 基本原理 (复数)恶劣天气

elementary [.elə'mentəri] adj.基本的 初级的 元素的

elephant ['elifənt] n.象 庞大笨拙难处理的事物

elevator ['eliveitə] n.电梯 升降机 飞机升降舵 斗式皮带输送机

eleven [i'levn] num.十一 十一个

eleventh [i'levnθ] num.第十一(个)

eliminate [i'limineit] vt.消灭 消除 排除 除去 剔除 忽略

elimination [i.limi'neiʃən] n.消灭 排除 除去 消除

else [els] adv.其它 另外 否则 adj.别的 其他的

elsewhere ['els'wɛə] ad.在别处 向别处

embarrass [im'bærəs] vt.使窘迫 使为难 阻碍 vi.变得困窘

embrace [im'breis] vt.拥抱 包括 包围 信奉 包含 n.拥抱

emerge [i'mə:dʒ] vi.出现 涌现 冒出 (由某种状态)脱出 (事实)显现出来

emergency [i'mə:dʒənsi] n.紧急情况 突然事件

emit [i'mit] vt.散发 发射 发表 发出 放射 吐露

emotion [i'məuʃən] n.情感 感情 情绪 激动

emotional [i'məuʃənl] adj.感情的 情绪的

emperor ['empərə] n.皇帝

emphasis ['emfəsis] n.强调 重点 重要性

emphasize ['emfəsaiz vt.强调 着重

empire ['empaiə] n.帝国

employ [im'plɔi] vi.雇用 使用 使忙于

employee [.emplɔi'i:] n.受雇者 雇员 雇工

employer [im'plɔiə] n.雇佣者 雇主

employment [im'plɔimənt] n.工业 雇用 使用 职业

empty ['empti] a.空的 空洞的

enable [i'neibl] vt.使能够 使可能 [计算机] 使能

enclose [in'kləuz] vt.围住 圈起 附上 放入封套 [计算机] 括入

encounter [in'kauntə] vt. & vi.遭遇 遇到 偶然碰到 n.遭遇 意外的相见

encourage [in'kʌridʒ] vt.鼓励 支持 助长 激励 支持

end [end] n.末端 目标 结束 终点 vt. & vi.结束 终结 adj.最终的

ending ['endiŋ] n.结尾 结局 死亡

endless ['endlis] adj.无止境的 没完没了的

endure [in'djuə] vt.忍受 容忍 vi.持久 持续

enemy ['enimi] n.敌人 仇敌 敌兵 敌对势力 有害物

energy ['enədʒi] n.活力 精力 能力 能 能量

enforce [in'fɔ:s] vt.实施 执行 强制 强迫 加强

engage [in'geidʒ] vt.使从事于 聘用 答应 预定 使忙碌 订婚

engine ['endʒin] n.发动机 引擎 机车 火车头 vt.装引擎

engineer [.endʒi'niə] n.工程师 技师 机械师 vt.操纵 设计 建造

engineering [.endʒi'niəriŋ] n.工程 工程学

England ['inglənd] n.英格兰 英国

English ['iŋgliʃ] n.英语 adj.英国人的 英国的 英语的 vt.译成英语 把...(外来语言)加到英语中

Englishman ['iŋgliʃmən] n.英国男子

enjoy [in'dʒɔi] vt. & vi.享受 欣赏 喜爱

enlarge [in'lɑ:dʒ] vt.扩大 扩展 增大 放大

enormous [i'nɔ:məs] adj.巨大的 庞大的

enough [i'nʌf] adj.足够的 adv.足够地 完全地 pron.足够 受够

ensure [in'ʃuə] vt.保证 保护 赋予

enter ['entə] vt. 走进 进入 参加 报名 开始 加入 报价 提出 vi. 进入 参加 开始 上台 进入角色 [计算机] 送入

entertain [.entə'tein] vt. & vi.使欢乐 招待 怀抱

enthusiasm [in'θju:ziæzəm] n.热情 热心 热忱 热衷的事物

enthusiastic [in.θju:zi'æstik] adj.热情的 热心的

entire [in'taiə] adj.全部的 整个的 完整的 同性质的 纯正的 全面的

entitle [in'taitl] vt.给...权利(或资格) 取名为 使有权利

entrance ['entrəns,in'trɑ:ns] n.入口 门口 进入 登场 vt.使神志恍惚

entry ['entri] n.入口处 登记 进入 入口 项目

envelope ['enviləup] n.信封 封套 封皮 壳层

environment [in'vaiərənmənt] n.环境 外界 围绕

envy ['envi] vt. & vi. & n.妒忌 嫉妒 羡慕

equal ['i:kwəl] adj.相等的 平等的 胜任的 n.对手 匹敌 同辈 vt.等于 比得上 [计算机] 相等

equality [i:'kwɔliti] n.等同 平等 相等

equation [i'kweiʃən] n.方程(式) 等式 均衡 相等

equip [i'kwip] vt.装备 配备 设 穿…

equipment [i'kwipmənt] n.装备 设备 配备

equivalent [i'kwivələnt] adj.相等的 等量的 n.相等物

era ['iərə] n.时代 年代 纪元

erect [i'rekt] vt.建造 建立 使竖立 adj.直立的 竖立的 笔直的

error ['erə] n.错误 谬误 差错

esp.[is'peʃəl] adj.(=especial)特别的 特殊的 尤其 adv.(=especially)特别 特殊

escape [is'keip] vt. & vi.逃跑 逃脱 避开 逸出 n.逃跑 逃亡 避难设备 adj.逃避的

especially [is'peʃəli] adv.特别 尤其 格外

essay ['esei,e'sei] n.短文 散文 论文 小品文 尝试 vt.尝试

essential [i'senʃəl] adj.必要的 重要的 本质的 n.要素 要点

establish [i'stæbliʃ] vt.建立 设立 确立 创办

establishment [is'tæbliʃmənt] n.建立 设立 确立 设施 机构 权威

estimate ['estimeit] vt.估计 估价 评价 n.估计 估价

etc.[et'setərə] n.附加的人 附加物 以及其它 等等 = etcetera

Europe ['juərəp] n.欧洲

European [.juərə'pi(:)ən] adj.欧洲的 n.欧洲人

evaluate [i'væljueit] vt.评估 评价 估...的价

evaporate [i'væpəreit] vt.使蒸发 vi.蒸发 失去水分 消失

eve [i:v] n.前夜 前夕 前一刻 [Eve] 夏娃

even ['i:vən] adv.甚至 甚至更 还 恰好 正当 adj.平坦的 偶数的 相等的 均匀的 v.使平坦 相等

evening ['i:vniŋ] n.傍晚 黄昏 晚上

event [i'vent] n.事件 大事 事变 项目 活动

eventually [i'ventjuəli] adv.终于 最后

ever ['evə] adv.在任何时候 曾经 永远 究竟

every ['evri] adj.每一的 每隔...的

everybody ['evribɔdi] pron.每人 人人 各人

everyday ['evri'dei] adj.每天的 日常的 平常的

everyone ['evriwʌn] pron.每人 人人 n.每个人

everything ['evriθiŋ] pron.每件事 事事 一切

everywhere ['evriwɛə] adv.到处 处处 无论何处

evidence ['evidəns] n.根据 证据 证人 vt. & vi.证实 证明

evident ['evidənt] adj.明显的 明白的

evil ['i:vl] n.邪恶 罪恶 祸害 adj.邪恶的 坏的

evolution [.i:və'lu:ʃən] n.进化 演化 发展

evolve [i'vɔlv] vt. & vi.使进化 使发展 展开

exact [ig'zækt] adj.确切的 精确的 vt.强求 要求 索取

exactly [ig'zæktli] adv.确切地 恰恰正是

exaggerate [ig'zædʒəreit] vt. & vi.夸大 夸张

exam [ig'zæm] n.考试 检查 细查=examination 考试

examination [ig.zæmi'neiʃən] n.考试 检查 细查 审讯

examine [ig'zæmin] vt.检查 仔细观察 调查 考试 vi.调查 [计算机] 检测

example [ig'zɑ:mpl] n.例子 实例 模范 榜样 vt.把...作为例子

exceed [ik'si:d] vt.超过 胜过 超出 超出界限 vi.领先

exceedingly [ik'si:diŋli] adv.极端地 非常 很

excellent ['eksələnt] adj.优秀的 杰出的

except [ik'sept] prep. & conj.除了...之外 若不是 除非 vt.除 除外

exception [ik'sepʃən] n.例外 除外

excess [ik'ses] n.超越 过量 过度 adj.过量的 额外的

excessive [ik'sesiv] adj.过多的 过分的 极度的

exchange [iks'tʃeindʒ] vt. & vi.交换 交流 交易 兑换 n.交换 交换物 汇兑 交易所

excite [ik'sait] vt.使激动 引起 刺激 激励

exciting [ik'saitiŋ] adj.令人兴奋的

exclaim [iks'kleim] vi.呼喊 惊叫 大声叫

exclude [iks'klu:d] vt.把...排除在外 拒绝 [计算机] 排除

exclusively [iks'klu:sivli] adv.专门地 排他地(独占地 专门地 仅仅 只)

excursion [iks'kə:ʃən] n.远足 短途旅行

excuse [iks'kju:z] vt.原谅 免除 辩解 准许离开 同意赦免 n.借口 理由 低劣的样品 假条

execute ['eksikju:t] vt.将...处死 实施 执行 完成 [计算机] 执行

executive [ig'zekjutiv] adj.执行的 行政的 决策的 经营的 [计算机] 执行指令 n.执行者 主管 行政部门

exercise ['eksəsaiz] n.锻炼 训练 练习 作业 运动 vi.练习 锻炼 vt.运用

exert [ig'zə:t] vt.尽(力) 运用 施加(压力 影响等)

exhaust [ig'zɔ:st] vt.使筋疲力尽 用尽 排空 vi.耗尽 n.排气 排气装置

exhibit [ig'zibit] vt.显示(感情或品质) 陈列 展览 展现 提出证据 vi.展览 n.展览品 展览 物证

exhibition [.eksi'biʃən] n.展览 陈列 展示 展览会

exist [ig'zist] vi.存在 生存 生活

existence [ig'zistəns] n.存在 实在 生存

exit ['eksit] n.出口 退场 离去 vi.退出 死亡 vt.离开

expand [iks'pænd] vt.扩大 增加 详述 扩展 使膨胀 vi.扩展 增加 详述

expansion [iks'pænʃən] n.扩大 扩充 扩张 膨胀

expect [iks'pekt] vt.预料 预期 等待 盼望

expectation [.ekspek'teiʃən] n.期待 期望 预期

expense [iks'pens] n.花费 消费 费用 代价

expensive [iks'pensiv] adj.昂贵的 花钱多的 豪华的

experience [iks'piəriəns] n.经验 感受 经历 vt.经历 体验

experiment [iks'perimənt] n.实验 试验 尝试 vi.做实验 尝试

experimental [iks.peri'mentl] adj实验(性)的 试验(性)的

expert ['ekspə:t] n.专家 能手 adj.熟练的 内行的 vi.当专家

explain [iks'plein] vt. & vi.解释 说明 为...辩解

explanation [.eksplə'neiʃən] n.解释 说明 辩解

explode [iks'pləud] vt.使爆炸 驳倒 vi.爆炸 爆发(感情) 激增 [计算机] 爆炸

exploit ['eksplɔit,iks'plɔit] vt. & vi.剥削 利用 开拓 开发 n.功绩 勋绩

explore [iks'plɔ:] vt. & vi.探险 探索 探测 探究 [计算机] 探讨

explosion [iks'pləuʒən] n.爆炸 爆发 炸裂 激增

explosive [iks'pləusiv] n.炸药 adj.爆炸(性)的

export ['ekspɔ:t,eks'pɔ:t] vt.输出 出口 运走 n.出口 输出品 vi.传播 输出 adj.出口的

expose [ik'spəuz] vt.使暴露 揭露 揭穿 使遭受(危险或不快)

exposure [iks'pəuʒə] n.暴露 揭露 显露 曝光 面临(困难)

express [iks'pres] vt.表达 表示 挤压出 快递 n.快车 快递 专使 adj.急速的 明确的 直接的 adv.用快递 特意地

expression [iks'preʃən] n.词句 措辞 表达 表示 表现 表情

extend [iks'tend] vt.延长 延伸 伸展 扩大 扩充 扩展 致

extension [iks'tenʃən] n.延长部分 伸展 扩充 范围 分机号码

extensive [iks'tensiv] adj.广阔的 广泛的

extent [iks'tent] n.广度 范围 程度 宽度 长度 大小 [律] <英>扣押 <美>临时所有权令

exterior [eks'tiəriə] adj.外部的 外在的 表面的 对外的 [计算机] 外部的 n.外部 表面 外型

external [eks'tə:nl] adj.外部的 外面的 外用的 客观的 表面的 n.外部

extra ['ekstrə] adj.额外的 ad.特别地 n.额外的事物 另外收费的

extraordinary [iks'trɔ:dnri] adj.非同寻常的 特别的 非凡的 特派的

extreme [iks'tri:m] adj.极度的 极端的 尽头的 严重的 末端的 n.极端 极限

extremely [iks'tri:mli] adv.极端 极其 非常

eye [ai] n.眼睛 眼力 鉴赏力 v.看 注视

eyesight ['aisait] n.视力 目力

## F

facility [fə'siliti] n.设备 容易 便利 灵巧 容易 熟练

fact [fækt] n.事实 实际 实情

factor ['fæktə] n.因素 因子 系数 vt.把…因素包括进去 vi.作为因素

factory ['fæktəri] n.工厂 制造厂

faculty ['fækəlti] n.才能 能力 (大学的)系 科 全体教员

fade [feid] vi.褪色 逐渐消失 凋谢 vt.使褪色 n.淡入 淡出 adj.平凡的 乏味的

Fahrenheit ['færənhait, 'fɑ:r-] n.华氏温度计 adj.华氏的

fail [feil] vi.失败 失灵 不能 衰退 不及格 vt.辜负 缺少 未能做 n.不及格

failure ['feiljə] n.失败 失败的人或事

faint [feint] adj.微弱的 虚弱的 无力的 模糊的 v.昏倒 n.昏厥 昏倒

fair [fɛə] adj.公平的 晴朗的 美丽的 相当的 (皮肤 毛发等)浅色的 白皙的 n. 展览会 市集 定期集市 adv. 公平地 直接地

fairly ['fɛəli] adv.相当 公平地 公正地 合法地 适当地

faith [feiθ] n.信任 信心 信仰

faithful ['feiθfəl] adj.忠诚的 如实的

fall [fɔ:l] vi.落下 跌倒 倒下 陷落 来临 n.秋天 落下 瀑布

false [fɔ:ls] adj.不真实的 伪造的 人造的 假的 错误的 虚伪的 adv.不真诚地 虚伪地

fame [feim] n.名声 名望 vt.使有名望

familiar [fə'miljə] adj.熟悉的 熟知的 不拘礼节的 冒昧的 n.熟人 熟悉某种事物的人

family ['fæmili] n.家 家庭 家族 adj.家庭的

famine ['fæmin] n.饥荒 严重的缺乏

famous ['feiməs] adj.著名的 出名的 一流的

fan [fæn] n.(运动等)狂热爱好者 风扇 扇形物 vt.风选 煽 吹 煽动 向扇子一样展开 vi.像扇子一样行动 鱼贯而出

fancy ['fænsi] n. 想象力 设想 幻想 爱好 喜好 爱 adj. 想像的 时髦的 华丽装饰的 奢侈的 技巧的 vt. 想象 自认为 喜好

far [fɑ:] adj.远的 adv.远 遥远 n.远处

fare [fɛə] n.车费 船费 票价 费用 食物 vi.进展 进食

farewell ['fɛə'wel] int.再会 别了 n.告别 adj.告别的

farm [fɑ:m] n.农场 农庄 农田 饲养场 vt.收费 务农 vi.务农

farmer ['fɑ:mə] n.农民 农夫 农场主

farther ['fɑ:ðə] adv.更远地 进一步地 adj.更远的 进一步的

fashion ['fæʃən] n.样子 方式 风尚 流行 时装 vt.形成 适合

fashionable ['fæʃənəbl] adj.流行的 时髦的

fast [fɑ:st] adj. 快速的 偏快的 固定的 关紧的 忠诚的 聪明的 耐久的 野性的 adv. 快 很快地 固定地 彻底地 鲁莽的 超前的 vi. 绝食 斋戒 n. 禁食(期) 系泊用具

fasten ['fæsn] vt.扎牢 扣住 拴紧 使固定 系 强加于 vi.固定 系紧 抓紧

fatal ['feitl] adj.致命的 毁灭性的 决定性的 命运的

fate [feit] n.命运 天数 天意 vt.注定

father ['fɑ:ðə] n.父亲 神父 创始人 鼻祖 vt.作为父亲生育子女 vi.像父亲般照顾

father-in-law ['fɑ:ðərinlɔ:] n.岳父 公公

fatigue [fə'ti:g] n.疲劳 劳累 累活 adj.疲劳的 vt.使...疲劳 使...心智衰弱 vi.疲劳

fault [fɔ:lt] n.缺点 过失 故障 vt. & vi.挑剔 指责 发现错误 断裂

faulty ['fɔ:lti] adj.有错误的 有缺点的

favour ['feivə] n.好感 赞同 恩惠 好意 喜爱 偏爱 特权 vt.给予 关心 偏爱 支持 利于 长得像 =favor

favourable ['feivərəbl] adj.有利的 有用的 良好的 赞成的

favourite ['feivərit] adj.特别受喜爱的 特别喜爱的 n.特别喜爱的人(或物)

fear [fiə] n.害怕 担心 敬畏 不安的原因 vt.害怕 敬畏

fearful ['fiəfəl] adj.害怕的 可怕的 担心的

feasible ['fi:zəbl] adj.可行的 可能的

feast [fi:st] n.盛宴 宴会 酒席 筵席 节日 节会 vt. & vi.款宴 享乐 请客

feather ['feðə] n.羽毛 翎毛 羽状物 心情 种类 服饰 vt.用羽毛装饰 vi.长出羽毛

feature [fi:tʃə] n.特征 特色 面貌 vt.以...为特色 vi.起重要作用

February ['februəri] n.二月

federal ['fedərəl] adj.联邦(制)的 联邦政府的 联盟的 同盟的 n.联邦主义者 同盟盟友

fee [fi:] n.费 酬金 赏金 封地

feeble [fi:bl] adj.虚弱的 微弱的 无力的

feed [fi:d] vt.喂(养) 饲养 靠...为生 vi.吃饲料 进食 n.饲养 饲料 一餐

feedback ['fi:dbæk] n.回授 反馈 反应 成果

feel [fi:l] vi.有知觉 似乎 同情 vt.触 摸 感觉 觉得 n.感觉 触觉 直觉 氛围

feeling ['fi:liŋ] n.感情 感觉 知觉 意识 气氛 adj.有感情的 富于同情心的 含情脉脉的

fellow ['feləu] n.人 家伙 伙伴 朋友 同类 校董 研究生 adj.同类的 同事的 同伴的

female ['fi:meil] n.雌性的动植物 女子 adj.雌的 女(性)的 凹的

fence [fens] n.栅栏 围墙 击剑术 买卖赃物的人 v.用篱笆围住 练习剑术 防护 买卖赃物

fertile ['fə:tail] adj.肥沃的 多产的 能繁殖的、

fertilize ['fə:tilaiz] v.施肥 使丰饶

fertilizer ['fə:tilaizə] n.肥料

festival ['festəvəl] n.节日 音乐节 庆祝及祭祀 欢宴 adj.节日的 喜庆的

fetch [fetʃ] vt.拿来 请来 接去 航行到达 vi.取回 兜圈子 n.取得

fever ['fi:və] n.发热 发烧 狂热 vt.激怒 vi.发烧

few [fju:] adj.很少的 少数的 pron.少数 n.少量

fibre ['faibə] n.纤维 纤维质 纤维组织 织地

fiction ['fikʃən] n.小说 虚构 杜撰

field [fi:ld] n.田野 田地 原野 牧场 场地 领域 运动场 adj.田间的 野外的 vi.担任场外队员 vt.把(谷物等)暴晒于场上 上场

fierce [fiəs] adj.凶猛的 猛烈的 残暴的 强烈的 狂热的

fifteen ['fif'ti:n] num.十五 十五个 pron.十五(个 只...) adj.十五的 十五个的

fifth [fifθ] num.第五 n.五分之一

fifty ['fifti] num.五十 五十个

fight [fait] vi.打架(仗) 斗争 战斗 斗志 vt.对抗 打架 vi.斗争 奋斗

figure ['figə] n.数字 外形 人物 图形 形状 外形 体型 v.演算 认为 领会到

file [fail] n.档案 公文箱 锉刀 [计算机] 文件 vt.把...归档 列队行进 申请

fill [fil] vt.装满 盛满 占满 填充 弥漫 担任 n.足量

film [film] n.影片 胶卷 薄层 薄膜 vt.把...拍成电影 给...覆上一薄层 vi.从事电影拍摄

filter ['filtə] vt. & vi.过滤 渗透 n.滤纸 筛选 滤波器 过滤器 滤色镜 [计算机] 过滤

final ['fainl] adj.最后的 决定性的 n.决赛 期末考试

finally ['fainəli] adv.最后 最终 不可更改的

finance [fai'næns] n.财政 金融 财源 vt.给...提供资金 负担经费

financial [fai'nænʃəl] adj.财政的 金融的

find [faind] vt. & vi.找到 发觉 找出 感觉到 认为 n.发现物

finding ['faindiŋ] n. 发现 发现物 调查(或研究)的结果 [律] 裁判 裁决 (复数）findings：(珠宝商等使用的)零碎的工具或材料 find的现在分词

fine [fain] a. 美好的 优质的 精致的 晴朗的 健康的 纤细的 恭维的 辞藻华丽的 adv. 恰好地 合适地

n. 罚款 罚金 结尾 vt. 罚款 vi. 净化 细化

finger ['fiŋgə] n.手指 指针 指状物 vt.用手指触碰(演奏) (用手)指 vi.触摸 用手指演奏乐器

finish ['finiʃ] vt.完成 结束 获...第几名 vi.终结 完成 n.结束 完成 最后加工 结局 完美

fire ['faiə] n.火 火灾 射击 vi.点燃 解职 开枪 开火

fireman ['faiəmən] n.消防队员 司炉工

firm [fə:m] n. 商行 商号 公司 vt. (使)变得坚实 定案 vi. 变硬 恢复 adj. 坚定的 坚固的 结实的 adv. 稳固地 坚定地

first [fə:st] num. 第一 adv. 最初 首先 第一 优先 adj. 第一的 最早的 一流的 n. (序数词)第一 第一个人或事物 (大学学位的) 最优等 (汽车 自行车等的) 最低挡

fish [fiʃ] n.鱼 鱼肉 vi.钓鱼 捕鱼

fisherman ['fiʃəmən] n.渔民 渔夫 打鱼人 渔船

fist [fist] n.拳(头) 抓住 vt.紧握 用拳头打

fit [fit] vt.适合 安装 vi.适合 n.适宜 发作 adj.适宜的 健壮的

five [faiv] num.五 五个 第五

fix [fiks] vt. & vi.使固定 决定 修理 准备 n.困境 (船只、飞机等的)定方位 贿赂

flag [flæg] n.旗 旗帜 旗标 信号旗 旗舰旗 vt.打旗号 vi.无力地下垂 减退

flame [fleim] n. 火焰 火舌 鲜红色 光辉 恋人 怀念 辱骂信息 热情 vi. 焚烧 爆发 面红 发辱骂信息

vt. 用烟火发信号 用火对待 发辱骂信息

flare [flɛə] vi.闪耀 闪光 vt.使闪亮 n.闪光 闪耀 [天] 耀斑

flash [flæʃ] n. 闪光 闪现 闪烁物 花哨 一瞬间 电脑动画技术 vi. 闪 闪现 闪烁 vt. 传送 用光联络 流露出 adj. 花哨的 炫耀的 昙花一现的

flat [flæt] adj.平的 扁平的 平坦的 单调的 adv.平直地 干脆地 n.一套房间 单元住宅 [英] 公寓 vt.变平 vi.降调

flavour ['fleivə] n.味 味道 风味 香味 vt给...加味

fleet [fli:t] n.舰队 船队 团队 机群 vt. & vi.飞过 掠过 adj.跑得快的

flesh [fleʃ] n.肉 肌肉 肉体 人性 vt.用肉喂 赋予血肉 使长肉 vi.变胖

flexible ['fleksəbl] adj.易弯曲的 灵活的 柔韧的 可变通的

flight [flait] n.航班 飞行 逃跑 航程 (鸟等)群 奇思妙想 一段楼梯 vi.滑行 vt.冲

float [fləut] vi. 漂浮 游荡 飘 vt. 使漂浮 提出 开办新公司 洽谈 n. 漂流物 装有轮子的展台或平板车 企业的日常零钱

flock [flɔk] n.羊群 群 大量 信众 软填料 短绒 棉絮 毛屑 vt.塞填充料 vi.成群而行 聚集

flood [flʌd] n.洪水 水灾 vt. & vi.灌溉 淹没 泛滥

floor [flɔ:] n.地板 楼层 底部 地面 议员 底线 vt.铺地板 击倒 油门踩到底

flour [flauə] n.面粉 粉 粉状物质 vt. & vi.研成粉末

flourish ['flʌriʃ] vi.繁荣 茂盛 兴旺 活跃 手舞足蹈 vt.挥舞 n.挥舞 华丽词藻 茂盛 兴旺 炫耀

flow [fləu] vi.流动 飘垂 涨潮 顺利进行 来 vt.使流动 淹没 n.流 水流 循环 顺畅 涨潮

flower ['flauə] n.花 花卉 开花 华丽 盛年 精英 自然发展的结果 vi.成熟。开花 vt.使开花 用花装饰

flu [flu:] n.流行性感冒

fluent ['flu:ənt] adj.流利的 流畅的

fluid ['flu:id] n.流体 液体 adj.流畅优美的 流体的 流动的 不稳定的

flush [flʌʃ] vi.奔流 (脸)发红 冲刷 旺盛成长 vt.用水冲洗 惊起(一群鸟) 抄...老窝 使兴奋 adj.丰足的 齐平的 满盈的 adv.正直地 羞涩地 n.流溢 面红 旺盛 冲水(尤指冲马桶) 突然萌发 (纸牌)同花顺 清一色 [计算机] 冲洗

fly [flai] n.蝇 苍蝇 两翼昆虫 飞行 升高 vt. & vi.飞行 飞翔 逃走 飞逝 猛然移动

focus ['fəukəs] vi.聚焦 注视 集中 聚集 使集中 n.焦点 焦距

fog [fɔg] n.雾 烟雾 尘雾 朦胧 喷雾 迷惑 vt. & vi.以雾笼罩 使模糊 迷惑

fold [fəuld] vt.折叠 合拢 包 交叉 拥抱 vi.翻倍 彻底失败 倒闭 n.褶 折层 折痕 羊栏 羊群

folk [fəuk] n.人们 家属 亲属 adj.民间的

follow ['fɔləu] vt.跟随 沿行 遵循 听得懂 结果是 vi.跟着做 之后是 n.跟随

following ['fɔləuiŋ] adj.接着的 下列的 其次的 prep.在(某事)以后 n.下列各项 部下 党羽

fond [fɔnd] adj.喜爱的 溺爱的 愚蠢的 渴望的 n.基础

food [fu:d] n.食物 食品 养料

fool [fu:l] n.蠢人 傻子 受骗者 vt.欺骗 愚弄 vi.干傻事 开玩笑 adj.愚蠢的

foolish ['fu:liʃ] adj.愚蠢的 鲁莽的 荒谬的 可笑的

foot [fut] n.脚 最下部 英尺 山脚 最底部 一步长 音步 进程 vi.跳舞 步行 vt.表演舞步 付账 踩踏 结算

football ['futbɔ:l] n.足球比赛 足球 橄榄球

footstep ['futstep] n.脚步 脚步声 足迹

for [fɔ:] prep.为 给 因为 关于 代表 支持 往 conj.因为

forbid [fə'bid] vt.禁止 不许 阻止 妨碍

forbid [fə'bidn] adj.被禁止的

force [fɔ:s] vt.强迫 施暴 强夺 加压力 [计算机] 强制 n.力 力量 武力 暴力 影响力

forecast ['fɔ:kɑ:st] n.预测 预报 vt. & vi.预示 预测

forehead ['fɔ:hed] n.额头 前部

foreign ['fɔrin] adj.外国的 外来的 外交的 不相关的

foreigner ['fɔ:rinə] n.外国人

foremost ['fɔ:məust] adj.最初的 第一流的 adv.在最前面

forest ['fɔrist] n.森林 森林地带 园林 林火 vt.用树林覆盖 植树

forever [fə'revə] adv.永远 总是 老是 n.漫长时间

forget [fə'get] vt. & vi.忘记 遗忘 忽略

forgive [fə'giv] vt. & vi.原谅 饶恕 宽恕

fork [fɔ:k] n.餐叉 叉 叉状物 vt. & vi.叉起 耙 成叉状

form [fɔ:m] n.形式 形状 表格 vt.形成 组成 建立

formal ['fɔ:məl] adj.正式的 礼仪上的 形式的 拘谨的 n.正式(活动)

formation [fɔ:'meiʃən] n.形成 构成 形成物 编队

former ['fɔ:mə] adj.在前的 以前的 前任的 n.前者 形成的原因

formula ['fɔ:mjulə] n.公式 式 配方 规则 代乳品 adj.(赛车的)级的 方程式的

forth [fɔ:θ] adv.向前 向外 往外 露出

fortnight ['fɔ:tnait] n.两星期 十四天

fortunate ['fɔ:tʃənit] adj.幸运的 侥幸的

fortunately ['fɔitʃənitli] adv.幸运地 幸亏

fortune ['fɔ:tʃən] n.命运 运气 财产

forty ['fɔ:ti] num. & adj.四十 第四十 pron.四十(个 只...)

forward ['fɔ:wəd] adv.向前 今后 往后 adj.向前的 早的 迅速的 前锋 v.转交 促进 运送

found [faund] vt.创立 创办 建立 打地基 铸造 vbl.寻找(find的过去分词) adj.装备完善的 n.食宿

foundation [faun'deiʃən] n.基础 根据 建立 地基 基金

fountain ['fauntin] n.泉水 喷泉 源泉 储水容器 vt. & vi.使像喷泉一样流

four [fɔ:] num.四 四个 第四

fourteen num.十四 第十四

fourth [fɔ:θ] num.第四 四分之一 n.四分之一

fox [fɔks] n.狐狸 狡猾的人 v.奸狡地行动

fraction ['frækʃən] n.小部分 片断 分数 破片

fragment ['frægmənt] n.碎片 破片 碎块 vt. & vi.变成碎片 [计算机] 碎片

frame [freim] n.框架 框子 构架

framework n.框架 构架 结构 骨架 v.构成 把...框起来 陷害 adj.木结构的

France [frɑ:ns] n.法国 法兰西

frank [fræŋk] adj.坦白的 直率的 无误的 vt.免费邮寄 使自由出入 n.免费邮寄

free [fri:] adj.自由的 空闲的 大方的 免费的 vt.释放 解放 adv.自由地 免费地

freedom ['fri:dəm] n.自由 自主

freely ['fri:li] adv.自由地 免费地 直率地 大量地

freeze [fri:z] vi.冻 结冻 冷动 僵硬 [计算机] 冻结 vt.使结冰

freight [freit] n.货运 货物 运费 船货 vt.装货 运送 使充满(某种心情或口气)

French [frentʃ] adj.法国(人)的 法语的 n.法国人 法语 vt.剁肉 蔬菜切成长条

frequency ['fri:kwənsi] n.屡次 次数 频率 频繁

frequent ['fri:kwənt] adj.时常发生的 经常的 频繁的 vt.常到 常去

frequently ['fri:kwəntli] adv.时常 常常 经常地 频繁地

fresh [freʃ] adj.新的 新鲜的 无经验的 淡(水)的 精神饱满的 纯洁无瑕的 生动的 adv.最近地 n.涌流

friction ['frikʃən] n.摩擦 摩擦力 分歧

Friday ['fraidi] n.星期五

fridge [fridʒ] n.电冰箱

friend [frend] n.朋友 友人 赞助者 支持者 vt.做朋友 对人友好

friendly ['frendli] adj.友好的 友谊的 adv.友好地 n.友好者

friendship ['frendʃip] n.友谊 友好

frighten ['fraitən] vt.使惊恐 吓唬 惊恐

frog [frɔg] n.蛙 马的蹄叉 刀剑的挂环

from [frɔm] prep.从...来 离开

front [frʌnt] adj.前面的 n.前部 前面 开头 前线 vt.面对 朝向 对付 vi.面朝 做掩护

frontier ['frʌntjə] n.边境 边界 边疆 新领域 尖端 边缘

frost [frɔst] n.冰冻 严寒 霜 冷漠 vt. & vi.覆着霜 冻结 结霜

frown [fraun] vt. & vi.皱眉 蹙额 不同意 n.皱眉 不悦

fruit [fru:t] n.水果 果实 成果 vt. & vi.结果实

fruitful ['fru:tfəl] adj.多产的 肥沃的 富有成效的

fry [frai] vt.油煎 油炸 油炒 炸焦 vi.油炸 烧焦 n.(美)法式炸薯条

fuel ['fjuəl] n.燃料 木炭 vt.给...加燃料 vi.加燃料

fulfil [ful.fil] vt.履行 满足 完成 实践 使应验

full [ful] adj. 满的 完全的 充满的 丰富的 adv. 完全地 整整 vt. 使...充满 通过缩水、捶打或熨烫增加(布料的)重量和厚度 vi. 增厚 n. 全部 十分 极点 鼎盛

fun [fʌn] n.乐趣 娱乐 玩笑 有趣的人或事物 vi.开玩笑 adj.充满乐趣的 令人愉悦的

function ['fʌŋkʃən] n.功能 职务 函数 重大聚会 vi.运行 起作用

fund [fʌnd] n.资金 基金 存款 储备 vt.为...提供资金

fundamental [.fʌndə'mentl] adj.基础的 基本的 根本的 重要的 n.基本原理 基础

funeral ['fju:nərəl] n.葬礼 丧礼 丧葬 adj.葬礼(的)

funny ['fʌni] adj.古怪的 滑稽的 有趣的 莫名其妙的 n.笑话 (the funny)滑稽戏

fur [fə:] n.软毛 毛皮 裘皮 毛皮服饰 毛皮制品 vt.用毛皮制作 用毛皮盖于 vi.用毛皮制的

furious ['fjuəriəs] adj.狂怒的 狂暴的 猛烈的

furnace ['fə:nis] n.炉子 熔炉 鼓风炉 闷热地带 严峻考验

furnish ['fə:niʃ] vt.供应 提供 装备 布置

furniture ['fə:nitʃə] n.家具 装置 设备

further ['fə:ðə] adv.更远地 此外 进一步地 adj.更远的 v.促进 推动 vt.促进

furthermore ['fə:ðə'mɔ:] adv.而且 此外

future ['fju:tʃə] n.将来 未来 前途 前程 期货 adj.将来的

## G

gain [gein] vt.获得 得到 增进 增加 vi.增加 得到 (钟 表等)走得快 n.增进 增益 获得 利润

gallery ['gæləri] n.长廊 游廊 画廊 美术馆 顶层楼座 狭长的房间

gallon ['gælən] n.加仑(容量单位)

game [geim] n.游戏 比赛 猎物 行业 非法手段 策略 野味 戏弄 vi.赌博 vt.欺骗 adj.乐意做的 勇敢的 跛的

gang [gæŋ] n.一帮 一伙 队 群 帮 vt. & vi.结一组 联合在一起

gap [gæp] n. 缺口 间隔 差距 (意见、个性等的)差异 缝隙 漏洞 vt. 制造缺口 嘉普 ( 财富500强公司之一 总部所在地美国 主要经营服装零售)

garage ['gærɑ:ʒ] n.车库 加油站 汽车修理厂 vt.把...放入车库

garbage ['gɑ:bidʒ] n.垃圾 污物 废料 废物 食物残渣

garden ['gɑ:dn] n. 花园 菜园 果园 菜园 草地 公园 vi. 栽培花木 从事园艺工作 vt. 造园 以花园装饰 adj. 花园的 园艺的 绿化的 平凡的

gardener ['gɑ:dnə] n.园丁 花匠 园艺家

gas [gæs] n.煤气 气体 汽油 空话 能量 vt.使吸入毒气 死 给...加汽油 vi.空谈 加汽油

gaseous ['gæsiəs] adj.气体的 气态的

gasoline ['gæsəli:n] n.(美)汽油

gasp [gɑ:sp] vi.气喘 喘息 v.喘气 喘息 渴望

gate [geit] n.大门 入口 篱笆门 vt.<英>惩罚(禁止学生离校) 解雇 n.<古>道路 <方>方式 suf.常表示政治丑闻

gather ['gæðə] vi.聚集 集合 渐增 vt.收集 使聚集 推断 鼓起 n.集合 聚集

gauge [geidʒ] vt.量 测量 估计 判断 n.测量标准 轨距 口径 直径 测量仪器

gay [gei] adj.快乐的 欢快的 鲜明的 色彩鲜艳的 同性恋的 n.同性恋

gaze [geiz] vi.凝视 盯 注视 n.凝视

general ['dʒenərəl] adj.总的 一般的 普遍的 全体的 n.将军 一般 大体

generally ['dʒenərəli] adv.一般地 通常地

generate ['dʒenəreit] vt.发生 产生 引起 生殖 [计算机] 产生

generation [.dʒenə'reiʃən] n.一代 一代人 产生

generator ['dʒenəreitə] n.发电机 发生器 发生者

generous ['dʒenərəs] adj.慷慨的 宽厚的 宽宏大量的 丰盛的 味浓的

genius ['dʒi:njəs] n.天才 天赋 天资 高智力

gentle ['dʒentl] adj. 和蔼的 轻柔的 显赫的 温和的 文雅的 有教养的 n. 贵族 vt. 使显赫 使温顺 抚摸 使缓和 vi. 变温顺

gentleman ['dʒentlmən] n.绅士 先生 有教养的人

gently ['dʒentli] ad.有礼貌地 柔和地 轻轻地 温柔地

genuine ['dʒenjuin] adj.真的 真正的 真实的 诚恳的

geography [dʒi'ɔgrəfi] n.地理 地理学

geometry [dʒi'ɔmitri] n.几何 几何学

germ [dʒə:m] n.微生物 细菌 幼芽

German ['dʒə:mən] adj.德国(人)的 德语的 日耳曼的 n.德国人 德语 日耳曼人

Germany ['dʒə:məni] n.德意志 德国

gesture ['dʒestʃə] n.姿势 手势 姿态 vt. & vi.作手势表达

get [get] vi. 变得 成为 到达 致富 能够 离开 vt. 得到 准备 抓住 克服 激怒 使得 记忆 沟通 n. 生育 后代 救球

ghost [gəust] n.鬼 灵魂 幽灵 鬼魂 一丝 vt. & vi.像(鬼一样)地行进

giant ['dʒaiənt] n.巨人 伟人 巨大的东西 adj.巨大的

gift [gift] n.礼物 赠品 天赋 赋予 vt.赠送 赋予

girl [gə:l] n.女孩子 姑娘 女儿

give [giv] vt.做 作 送给 vi.让步 塌下 折断 弯曲

glad [glæd] adj.高兴的 乐意的 v.<古>使高兴 n.剑兰

glance [glɑ:ns] vi.看一下 瞥闪 瞥见 反光 浏览 vt.瞥见 反光 n.一瞥 闪光 一滑

glare [glɛə] vi.瞪眼 发眩光 n.瞪眼 闪耀光 刺眼

glass [glɑ:s] n. 玻璃 玻璃杯 玻璃器皿 窗户 (复)眼镜 透镜 vt. 用玻璃把…盖(或罩、围)住 给...装上玻璃 使成玻璃状 反映 反射 审视 vi. 成为玻璃状 用光学仪器眺望

glide [glaid] vi.滑动 滑翔 溜走 消逝 n.滑行 滑动 滑过 流水

glimpse [glimps] vt. & vi.瞥见 一瞥 闪烁 n.一瞥 一看 一闪

glitter ['glitə] vi.闪闪发光 灿烂 闪烁 辉耀 n.闪光 灿烂 闪烁 辉耀

globe [gləub] n.地球 世界 地界仪 球体

gloomy ['glu:mi] adj.黑暗的 阴暗的 令人沮丧的 抑沉的 忧闷的

glorious ['glɔ:riəs] adj.光荣的 壮丽的 辉煌的

glory ['glɔ:ri] n.光荣 荣誉 壮丽 赞颂 荣誉的事 vi.为...而骄傲自豪 interj.用于表示惊叹

glove [glʌv] n.手套 vt.给...戴手套

glow [gləu] n.白热光 赤热 光辉 热情 vi.发白热光 发红光 红光焕发

glue [glu:] n.胶 胶水 粘性之物 vt.胶合 粘 紧附于...

go [gəu] vi.去 走 变为 进行 放 通往 消失 vt.走过 称重 [计算机] 执行

goal [gəul] n.球门 得分 目的 终点

goat [gəut] n.山羊 好色之徒 替罪羔羊

God [gɔd] n.神 神像 上帝 偶像 vt.像对神一样崇拜

gold [gəuld] n.金 金子 金币 金黄色 金牌 钱财 adj.金制的 金色的

golden ['gəuldən] adj.金(黄)色的 极好的

golf [gɔlf] n.高尔夫球 v.打高尔夫球 vi.(打)高尔夫球

good [gud] adj.好的 优良的 上等的 有本事的 n.善行 好处 [pl.] 商品

goodbye [.gud'bai] int.再见 再会 n.说再见

goodness ['gudnis] n.优良 德性 仁慈 善良 美德 精华 int.天哪

goods [gudz] n.货物 商品

goose [gu:s] n.鹅 雌鹅 鹅肉 vt.推动

govern ['gʌvən] vt.统治 治理 支配 管理 规定 vi.统治 执行

government ['gʌvənmənt] n.政府 政体 统治 治理 政治

governor ['gʌvənə] n.州长 主管人员 统治者 管理者 理事

gown [gaun] n.长袍 长外衣

grace [greis] n.优美 文雅 雅致 魅力 恩惠 慈悲 vt.使荣耀 装饰

graceful ['greisfəl] adj.优美的 优雅的

gracious ['greiʃəs] adj.有礼貌的 仁慈的 亲切的 高尚的

grade [greid] vt. & vi.给...分等级 记成绩 n.等级 年级 阶段

gradual ['grædjuəl] adj.逐渐的 渐进的

gradually ['grædjuəli] adv.逐渐地 逐步地

graduate ['grædjueit] n. 毕业生 vi. 大学毕业 获学位 逐渐变得 vt. 毕业 (在表、计、尺上)刻 [分] 度 定等级

adj. 毕业的 获得学位的

grain [grein] n.谷物 谷粒 颗粒 细粒 纹理 vt.用谷物喂养 形成颗粒

gramme [græm] n.克(重量单位) =gram

grammar ['græmə] n.语法 语法书

grammatical [grə'mætikəl] adj.语法上的 合乎文法的

grand [grænd] adj.宏伟的 重大的 豪华的 壮丽的 主要的 n.大钢琴

granddaughter ['græn.dɔ:tə] n.孙女 外孙女

grandfather ['grænd.fɑ:ðə] n.祖父 外祖父

grandmother ['grænd.mʌðə] n.祖母 外祖母

grandson ['grændsʌn] n.孙子 外孙子

grant [grɑ:nt] n.授给物 补助金 同意 给予 财产转让 vt.授予 同意 承认

Grant [grɑ:nt] n.格兰特(男子名)

grape [greip] n.葡萄 葡萄藤 [色彩]深紫色 [军事]葡萄弹

graph [græf,grɑ:f] n.(曲线)图 图表 vt.以(曲线)图表示

grasp [grɑ:sp] vt.抓紧 抓住 紧握 掌握 领会 n.抓 把握 抓紧 领会

grass [grɑ:s] n.草 牧草 草地 vt.以牧草喂养 覆盖着草

grateful ['greitfəl] adj.感激的 感谢的 令人愉快的

gratitude ['grætitju:d] adj.感激 感谢 感恩 n.感恩之心

grave [greiv] n.坟墓 沉音符 adj.严肃的 严重的 低沉的 adv.缓慢庄严的 vt.凿

gravity ['græviti] n.重力 引力 严重性 严重 庄重 严肃

gray [grei] adj.灰色的 阴沉的 黯淡的 n.灰色 v.变灰色 [Gray] 格雷(男子名)

great [greit] adj.大的 伟大的 大量的 很好的 美好的 重要的 adv.(口语)好极了

greatly ['greitli] adv.大大地 非常 很

greedy ['gri:di] adj.贪吃的 贪婪的

Greek [gri:k] adj.希腊(人)的 n.希腊人 希腊语

green [gri:n] adj.绿色的 长满绿草的 温和的 未成熟的 n.绿色 vt. & vi.变得绿

greenhouse ['gri:nhaus] n.温室 玻璃暖房

greet [gri:t] vt.问候 招呼 反应 致敬 欢迎 映入眼帘

greeting ['gri:tiŋ] n.问候 招呼 致敬

grey [grei] adj.灰色的 n.灰色 vi.变成灰色 =gray

grieve [gri:v] vt.使悲痛 vi.悲痛

grind [graind] vt.磨(碎) 碾 磨快 挤摩擦声 无聊的苦工作 vt.磨 碾 挤压 摇动手柄 vi.磨擦 磨碎 用功

grip [grip] vt.握紧 抓牢 吸引 n.紧握 了解 手柄 手提包

groan [grəun] vi.哼 呻吟 n.呻吟

grocer ['grəusə] n.食品商 杂货商

grocery ['grəusəri] n.食品杂货店

gross [grəus] adj.总的 粗鲁的 严重的 过胖的 n.总量 vt.总共赚得

ground [graund] n.土地 战场 场地 根据 背景 vt.放在地上 使...搁浅 打基础 grind的过去式和过去分词

group [gru:p] n.小组 群 团 团体 vt. & vi.聚合 成群 聚集 [计算机] 成组

grow [grəu] vt. & vi.生长 变得 变成 增长 种植 逐渐开始

growth [grəuθ] n.增长 增长量 生长 生长物

guarantee [.gærən'ti:] n.保证(书) 担保(书) 担保人 担保物 抵押品 vt.保证 担保

guard [gɑ:d] vt. & vi.守卫 看守 保卫 警惕 n.卫兵 守卫者 警戒 护卫队 保护

guess [ges] vt. &vi. &n.猜测 推测 猜想

guest [gest] n.客人 宾客 旅客 客串演员 特邀来宾 vt.款待 vi.作客

guidance ['gaidəns] n.引导 指导 领导 导航系统

guide [gaid] n.导游者 领路者 指南 路标 vt.给...导游 领路 指导 vi.作导游

guilty ['gilti] adj.内疚的 有罪的

gulf [gʌlf] n.海湾

gum [gʌm] n.口香糖 树胶 橡皮(糖) 齿龈 粘性物质 牙床 vt.搞砸 用牙床嚼 以树胶粘合 vi.形成胶质 发粘

gun [gʌn] n.枪 炮 手枪

gunpowder ['gʌn.paudə] n.黑色火药 有烟火药

gymnasium [dʒim'neiziəm] n. 体育馆 健身房 n. (德国等培养学生升入大学的)大学预科 高级中学

(复数) gymnasia 或 gymnasiums

## H

habit ['hæbit] n.习惯 习性 隐 长袍 vt.穿衣

habitual [hə'bitjuəl] adj.习惯性的 惯常的

hair [hɛə] n.头发 毛发 毛 毛织物 极少量 极短距离

haircut['hɛəkʌt] n.理发 发式

half [hɑ:f] n.半 一半 adj.一半的 不完全的 adv.一半地

hall [hɔ:l] n.门厅 过道 会堂 礼堂

halt [hɔ:lt] vi.停止 立定 跛行 犹豫 畏缩 vt.使...停止 n.停住 adj.瘸的

hamburger ['hæmbə:gə] n.汉堡包 牛肉饼

hammer ['hæmə] n.锤 榔头 vt.锤击 锤打 严厉批评 vi.锤击 反复敲打

hand [hænd] n.手 人手 指针 掌握 协助 vt.帮助 给 adv.手动地 人工地

handful ['hændful] n.一把 少数 一小撮 棘手之事 【非正式用语】 难以控制的人或费劲的事

handkerchief ['hæŋkətʃif] n.手帕 方头巾 围巾

handle ['hændl] n.柄 把手 vt.拿 触 买卖 处理 操作 vi.驾驭

handsome ['hænsəm] adj.英俊的 慷慨大方的 可观的 灵巧的 相当大的

handwriting ['hænd.raitiŋ] n.笔迹 手迹 书法

handy ['hændi] adj.手边的 方便的 便于使用的

hang [hæŋ] vt.挂 悬 吊死 悬挂方式 vt.(hanged hanged)悬挂 绞死 vi.(使)低垂 悬浮(在空中) 逗留

happen ['hæpən] vi.发生 碰巧 恰好 出现 偶然遇到

happiness ['hæpinis] n.幸福 幸运 快乐 适当

happy ['hæpi] adj.快乐的 幸福的 乐于做某事

harbour ['hɑ:bə] n.海港 港口 vt.庇护 停泊

hard [hɑ:d] adj.硬的 困难的 艰难的 苛刻的 adv.努力地 猛烈地 坚定地

harden ['hɑ:dn] vt.使变硬 (使)坚强 vi.变硬

hardly ['hɑ:dli] ad.几乎不 简直不

hardship ['hɑ:dʃip] n.艰难 困苦

hardware ['hɑ:dwɛə] n.五金器具 (计算机)硬件

hare [hɛə] n.野兔

harm [hɑ:m] n.伤害 损害 vt.伤害 损害

harmful ['hɑ:mfəl] adj.有害的

harmony ['hɑ:məni] n.调合 协调 和谐 和弦 和睦

harness ['hɑ:nis] vt.治理 束以马具 披上甲胄 利用(产生动力) n.马具 挽具 系在身上的绳子 甲胄

harsh [hɑ:ʃ] adj.严厉的 刺耳的 粗糙的 使人不舒服的 大约的

harvest ['hɑ:vist] n.收获(期) 成果 收获物 收成 vt. & vi.收割 收获

haste [heist] n.急速 急忙 草率 v.匆忙 赶快 vi.赶紧 赶快

hasten ['heisn] vt.催促 赶快 急忙 vi.赶紧 赶快 急忙

hasty ['heisti] adj.急速的 仓促的 匆匆的 轻率的 急忙的

hat [hæt] n.帽子(一般指有边的) 职位 vt.给...戴帽子 vi.提供帽子

hatch [hætʃ] vt.孵出 vi.(蛋)孵化 破壳 n.孵化 舱口

hate [heit] vt.恨 憎恨 不喜欢 不愿 vi.表示憎恨 n.憎恨 厌恶 厌恶的人或事

hateful ['heitfəl] adj.可恨的 可恶的

hatred ['heitrid] n.憎恶 憎恨 仇恨

have [hæv] v.aux.已经 曾经 vt.有 吃 从事 得到 雇用 享有 允许

hawk [hɔ:k] n.鹰 隼 掠夺别人的人 鹰派人物 清嗓 v.用训好的鹰捕猎 像鹰一般地袭击 鹰啸 清嗓 沿街叫卖

hay [hei] n.干草

hazard ['hæzəd] n.危险 危害 公害 冒险 vt.冒险 赌运气

he [hi:] pron.(主格)他

head [hed] n.头 头脑 领袖 冷静 硬币 正面 顶端 上端 高潮 vt. & vi.率领 为首 朝向 前进 超过 起源 adj.首要的 前面的

headache ['hedeik] n.头痛 头痛的事

heading ['hediŋ] n.标题 题词 题名 航向 动词head的现在分词

headline ['hedlain] n.(pl.)新闻提要 大字标题

headmaster ['hed'mɑ:stə] n.(中小学的)校长

headquarters ['hed'kwɔ:təz] n.司令部 指挥部 总部

heal [hi:l] vt.治愈 使和解 痊愈 恢复 vi.痊愈

health [helθ] n.健康 健康状况 卫生 保健

healthy ['helθi] adj.健康的 有益健康的

heap [hi:p] n.(一)堆 大量 许多 累积 vt.使成堆 装满 [计算机] 堆

hear [hiə] vt. & vi.听见 审讯 听说 听到 得知 倾听 听证

heart [hɑ:t] n.心 内心 中心 要点 红桃 勇气

heat [hi:t] n.热 炎热 热度 高潮 激动 激烈程度 辛辣 警察 vi.变热 变暖 因热变质 vt.使...变热

heating ['hi:tiŋ] n.加热 供暖 暖气装置 adj.加热的 供暖的 动词heat的现在分词

heaven ['hevn] n.天堂 天 天空

heavily ['hevili] adv.重重地 很重地 沉重地 沉闷地 大量地

heavy ['hevi] adj.重的 大(量)的 充满的 激烈的 难消化的 费力的 沉闷的 adv.密集地 n.悲剧角色

hedge [hedʒ] n. 篱笆 树篱 障碍物 防止财产损失的方法 vt. 包围 限制 vi. 种树篱 拐弯抹角 保护自己不受财产损失adj. 树篱的 篱笆边的

heel [hi:l] n.脚后跟 踵 后跟 v.倾侧

height [hait] n.高 高度 高处 身高 最高峰 极点

heir [ɛə] n.后嗣 继承人

helicopter ['helikɔptə] n.直升机

hell [hel] n.地狱 极大的痛苦

hello [hə'ləu] int.喂 n.(见面打招呼或打电话用语)喂 哈罗

helmet ['helmit] n.头盔 钢盔 遮阳帽 甲

help [help] vt. & vi.帮助 援助 改善 n.帮手 帮助 援助

helpful ['helpfəl] adj.给予帮助的 有用的 有益的

helpless ['helplis] adj.无助的 无能的 无依靠的

hen [hen] n.母鸡 雌禽

hence [hens] ad.因此 从此 所以 今后

her [hə:] pron.(宾格)她 她的 adj.她的 abbr.=heraldry

herd [hə:d] n.兽群 牧群 人群 牧人 vt.放牧 群集 使...成群 vi.聚在一起

here [hiə] adv.这里 向这里 adj.表示强调 n.这里

hero ['hiərəu] n.男主角 英雄 勇士

heroic [hi'rəuik] adj.英雄的 英勇的 巨大的

heroine ['herəuin] n.女主角 女英雄

hers [hə:z] pron.她的(所有物)

herself [hə:'self] pron.她自己 她亲自

hesitate ['heziteit] vi.犹豫 踌躇 含糊 停顿 口吃 迟疑 vt.用犹疑的态度表示

hi [hai] ini.嗨(表示问候等) 引人注意时的喊声 打招呼

hide [haid] vt. & vi.把...藏起来 隐瞒 遮避 鞭打 n.兽皮 生活富足度

high [hai] adj.高的 高级的 轻微变质的 高尚的 严重的 强的 adv.高 好 n.高地 天堂 高气压 高度 兴奋

highly ['haili] adv.高度地 很 非常

highway ['haiwei] n.公路 大路 (水陆空)直接航线或路线

hill [hil] n.小山 小堆 斜坡 山岗 丘陵 vt.堆起 培土

hillside ['hil'said] n.(小山)山腰 山坡

him [him] pron.(宾格)他

himself [him'self] pron.他自己 他亲自

hint [hint] n.暗示 提示 示意 建议 小部分 vt. & vi.暗示 示意

hire ['haiə] vt.租借 雇请 出租 n.租用 雇用 租用

his [hiz] pron.他的 他的东西

historical [his'tɔ:rikəl] adj.历史的 有关历史的

history ['histəri] n.历史 个人经历

hit [hit] vt. 打(击) 碰撞 偶然发现 要求 符合 vi. 打(击) 袭击 达到 n. 打(击) 冲撞 讽刺 成功.一次毒品的剂量 点击量

hobby ['hɔbi] n.业余爱好 癖好

hold [həuld] vt.拿住 掌握 拥有 持有 举行 n.把握 控制 支撑点 延误 n.船舱

hole [həul] n.洞 孔眼 裂开处 穴 vt. & vi.挖洞 掘坑 进洞

holiday ['hɔlədi] n.假日 节日 假期 vi.度假

hollow ['hɔləu] adj. 空的 空洞的 虚伪的 空腹的 vt. 形成空洞 挖空 vi. 形成空洞 adv. 有闷哑声 完全地 n. 洞 窟窿 山谷

holy ['həuli] adj.神圣的 圣洁的 令人敬仰的

home [həum] n.家 家乡 住所 国内的 养老院 adj.家庭的 家(乡)的 adv.回家 在家 正确的方向 vt. 提供住处 vi. 回家 移向

honest ['ɔnist] adj.诚实的 可敬的 正直的 adv.老实说

honesty ['ɔnisti] n.诚实 正直

honey ['hʌni] n.蜜 (复数)蜂蜜 甜 甜蜜 宝贝 爱人 adj. 蜂蜜似的 深爱的 vt.加蜜使甜 对...说甜言蜜语 vi.说甜言蜜语 奉承

honeymoon ['hʌnimu:n] n.蜜月 vi.度蜜月

honour ['ɔnə] n.光荣 尊敬 敬意 荣誉 荣幸 头衔 vt.尊敬 信守承诺 方形舞中鞠躬致敬

honourable ['ɔnərəbl] adj.诚实的 光荣的 荣誉的 可敬的 体面的 =honorable(美)

hook [huk] n.钩 挂钩 钩子 钩拳 vt. & vi.钩住

hope [həup] n. &vt. &vi.希望 abbr.=Health Opportunity for People Everywhere世界健康基金会

hopeful ['həupfəl] adj.有希望的

hopeless ['həuplisli] adj.没有希望的 绝望的 无可救药的

horizon [hə'raizn] n.地平线 眼界 见识 视野

horizontal [.hɔri'zɔntl] adj.地平的 水平的 横的 n.水平线 水平面

horn [hɔ:n] n.号角 警报器 角 动物角 喇叭 触角 角状物 力量源泉

horror ['hɔrə] n.恐怖 战栗 憎恶 厌恶 adj.恐怖的 令人毛骨悚然的

horse [hɔ:s] n.马 马科动物 vt.骑马 adj.马的

horsepower ['hɔ:spauə] n.马力(功率单位)

hospital ['hɔspitl] n.医院

host [həust] n.主人 东道主 军队 大量 主人 主持人 主机 vt.主持 做东 vi.聚集

hostess ['həustis] n.女主人 空姐

hostile ['hɔstail] adj.敌方的 不友善的 怀敌意的

hot [hɔt] adj.热的 热情的 刺激的 辣的 性感的 adv.激动地 快速地 n.最高温度 滚烫的东西

hotel [həu'tel] n.旅馆 宾馆 酒店 饭店

hour [auə] n.小时 时间 时刻 钟点

house [haus] n.房屋 住宅 商号 全家人 议院 vt.提供住 安置 收容 vi.居住 躲避

household ['haushəuld] n.家庭 户 家务 adj.家庭的 家喻户晓的

housewife ['hauswaif] n.家庭主妇 针线盒

how [hau] adv.如何 怎么 怎样 多少 conj.如何 怎样 以...方式 n.方式

however [hau'evə] adv.然而 不管怎样 conj.然而 可是 不过

huge [hju:dʒ] adj.巨大的 庞大的 程度高的

human ['hju:mən] adj.人的 人类的 n.人

humble ['hʌmbl] adj.谦逊的 地位低下的 卑下的 粗陋的 vt.使...卑下 贬低

humid ['hju:mid] adj.湿的 湿气重的 潮湿的

humorous ['hju:mərəs] adj.富于幽默的 诙谐的

humour ['hju:mə] n.幽默 诙谐 幽默感 体液 vi.纵容 迁就 =humor(美)

hundred ['hʌndrəd] num.百 百个 百位 n.许多

hunger ['hʌŋgə] n.饿 饥饿 渴望 v.饿 渴望 使...饥饿

hungry ['hʌŋgri] adj.饥饿的 渴望的

hunt [hʌnt] n.打猎 狩猎 搜寻 追捕 vt.追猎 狩猎 打猎 搜索

hurry ['hʌri] vi.赶紧 快速移动 vt.催促 加快 迅速处理 n.骚乱 匆忙 急需

hurt [hə:t] vt.使受伤 使痛心 伤害 (使)疼痛 损害 vi.受疼痛 破坏 n.打击 创伤

husband ['hʌzbənd] n.丈夫 vt.节俭

hut [hʌt] n.小屋 棚屋

hydrogen ['haidridʒən] n.氢

## I

I [ai] pron.(主格)我

ice [ais] n.冰 冰块 矜持 vt.冰镇 冰冻 结冰

ice-cream ['aiskri:m] n.冰淇淋

idea [ai'diə] n.想法 思想 意见 观念

ideal [ai'diəl] adj.理想的 观念的 完美的 高尚的 n.理想 典范 目标

identical [ai'dentikəl] adj.完全相同的 同一的

identify [ai'dentifai] vt.认出 识别 鉴定 vi.认同 感同身受

idiom ['idiəm] n.习语 成语 惯用语 方言 风格 特色

idle ['aidl] adj.空闲的 懒散的 无目的的 无聊的 懒惰的 闲散的 无根据的 i.e.(=that is)那就是 即

if if conj.假如 如果 是否 即使 n.条件 设想

ignorant ['ignərənt] adj.不知道的 无知的 愚昧的

ignore [ig'nɔ:] vt.不顾 不理 忽视

ill [il] adj.有病的 坏的 引起痛苦的 敌意的 adv.坏 恶劣地 勉强地 n.邪恶 不幸 祸害 坏话

illegal [i'li:gəl] adj.不合法的 非法的 n.非法移民

illness ['ilnis] n.病 疾病

illustrate ['iləstreit] vt.(用图等)说明 举例说明 作图解 vi.举例

illustration [i.ləs'treiʃən] n.说明 图解 插图 例证

image ['imidʒ] n.像 形象 映象 图像 影像 肖像 想像 vt.想像 描绘 反映

imaginary [i'mædʒinəri] adj.想象中的 假想的 虚构的

imagination [i.mædʒi'neiʃən] n.想象 想象力 空想

imagine [i'mædʒin] vt.想象 设想 料想 幻想 猜测 vi.想像起来 认为

imitate ['imiteit] vt.模仿 仿效 仿制

immediate [i'mi:djət] adj.立即的 直接的 即刻的 最接近的

immediately [i'mi:diətli] adv.立即 直接地 conj.一...(就)

immense [i'mens] adj.巨大的 广大的 非常好的 极好的

immigrant ['imigrənt] n.移民 侨民 adj.移民的

impact ['impækt,im'pækt] n.影响(力) 作用 冲击(力) 冲突 vt.挤入 压紧 撞击 对...发生影响

impatient [im'peiʃənt] adj.不耐烦的 急躁的

implication [.impli'keiʃən] n.含义 暗示 暗指

imply [im'plai] vt.暗示 意指

import ['impɔ:t,im'pɔ:t] vt. & vi. & n.输入 进口

importance [im'pɔ:təns] n.重要 重要性

important [im'pɔ:tənt] adj.重要的 影响很大的 有势力的

impose [im'pəuz] vt.把...强加 加上 强迫 征(税) 课征

impossible [im'pɔsəbl] adj.不可能的 办不到的

impress[im'pres] vt.给...深刻印象 影响 印刻 传送 vi.给人印象 n.印象 特徵 印记

impression [im'preʃən] n.印 印象 印记 效果

impressive [im'presiv] adj.给人印象深刻的

imprison [im'prizn] vt.关押 监禁 限制 束缚

improve [im'pru:v] vt.使更好 改进 利用 (提高)价值 vi.改善 改进

improvement [im'pru:vmənt] n.改进 改善 改进处

in [in] prep.在...里 在 穿着 (表示状态或状况) 参与 在...方面 adv.进 入

inch [intʃ] n.英寸 少量 v.慢慢移动

incident ['insidənt] n.发生的事 事件 (常指)军事冲突 插曲 事变 adj.难免的 附带的 (物理)入射的

incline ['inklain,in'klain] n.倾斜 斜坡 斜面 vt.使倾斜 使 ...倾向 爱好 易于

include [in'klu:d] vt.包括 包含

income ['in.kʌm] n.收入 所得 收益 进款

incorrect [.inkə'rekt] adj.不正确的 错误的

increase ['inkri:s,in'kri:s] vt. & vi. & n.增加 增强 提高

increasingly [in'kri:siŋli] adv.日益 越来越多地 逐渐地 渐增地

indeed [in'di:d] adv.真正地 确实 的确 事实上

indefinite [in'definit] adj.不明确的 模糊的 不定的

independence [.indi'pendəns] n.独立 自主 自立

independent [.indi'pendənt] adj.独立的 自主的 私立的 无偏见的 n.独立派人士 无党派者

index ['indeks] n.索引 指数 指标 指针 vt.编入索引中 指出 vi.编索引

India ['indiə] n.印度

Indian ['indjən] adj.印度的 印度人的 印第安人的 n.印度人 印第安人

indicate ['indikeit] vt.标示 表示 指明 表明 v.显示 象征 指示

indication [.indi'keiʃən] n.指示 表示 象征 表明

indifferent [in'difrənt] adj.冷漠的 不积极的 漠不关心的 无重要性的 中立的

indignant [in'dignənt] adj.愤慨的 义愤的 愤愤不平的

indirect [.indi'rekt] adj.间接的 不坦率的 迂回的 次要的 欺骗的

indispensable [.indis'pensəbl] adj.必不可少的 必需的

individual [.indi'vidjuəl] adj.个别的 独特的 n.个人 个体

indoors ['in'dɔ:z] adv.在室内 在屋里

industrial [in'dʌstriəl] adj.工业的 产业的 n.工业公司

industrialize [in'dʌstriəlaiz] vt.(使)工业化 =industrialise(英)

industry ['indəstri] n.工业 产业 勤劳

inefficient [.ini'fiʃənt] adj.效率低的 无效率的 不称职的 无能的

inevitable [in'evitəbl] adj.不可避免的 必然(发生)的

inexpensive [.inik'spensiv] adj.花费不多的 廉价的

infant ['infənt] n.婴儿 幼儿 adj.婴儿的 幼稚的 初期的

infect [in'fekt] vt.传染 感染

infer [in'fə:] vt.推论 推断 猜想 暗示 vi.推论

inferior [in'fiəriə] adj.下等的 劣等的 次等的 较低的 不如的

infinite ['infinit] adj.无限的 无穷的 无数的 n.无限

influence ['influəns] n.影响 影响力 势力 权势 有影响的人(或事物) vt.影响 感化

influential [.influ'enʃəl] adj.有影响的 有权势的 n.有影响力的人物

inform [in'fɔ:m] vt.通知 向...报告 告诉 告发

information [.infə'meiʃən] n.消息 信息 通知 情报 新闻 资料 询问

inhabit [in'hæbit] vt.居住于 栖息于 占据

inhabitant [in'hæbitənt] n.居民 住户

inherit [in'herit] vt.继承(传统等) 遗传 vi.继承

initial [i'niʃəl] adj.最初的 词首的 (词)首字母 adj.开始的 最初的 字首的 vt.签姓名的首字母于

injection [in'dʒekʃən] n.注射 注入 注射物 数学函数 充满

injure ['indʒə] vt.伤害 损害 损伤 使...受冤屈

injury ['indʒəri] n.损害 伤害 受伤处

ink [iŋk] n.墨水 油墨

inn [in] n.客栈 小旅店 小酒店

inner ['inə] adj.内部的 内心的 里面的 n.里面

innocent ['inəsnt] adj.清白的 幼稚的 无罪的 不懂事的 无知的 无辜的

input ['input] n.输入 投入的资金

inquire [in'kwaiə] vt.打听 询问 调查 vi.询问 查究

inquiry [in'kwaiəri] n.询问 打听 调查 查问 =enquiry(英)

insect ['insekt] n.昆虫 虫

inser [in'sə:t] t vt.插入 嵌入 登载 n.插入物

inside ['in'said] prep.在...里面 n.内部 adj.里面的 adv.在里面

insist [in'sist] vi.坚持 强调 坚持要求 vi.坚决主张

inspect [in'spekt] vt.检查 审查 调查 检阅 vi.调查

inspection [in'spekʃən] n.检查 审查 检阅 视察

inspire [in'spair] vt.鼓舞 给...以灵感 影响 使...感动 激发 煽动 vi.吸入

install [in'stɔ:l] vt.安装 设置 安置 使...就职

installation [.instə'leiʃən] n.安装 装置 设施 军事驻地

instance ['instəns] n.例子 实例 事例 阶段 情况 诉讼程序 vt.举例说明

instant ['instənt] n.瞬间 立即 adj.立即的 即时的

instantly ['instəntli] adv.立即 即刻

instead [in'sted] adv.代替 顶替 反而

instinct ['instiŋkt] n.本能 直觉 生性 天性 adj.充满的

institute ['institju:t] n.研究所 学院 学会 协会 vt.创立 开始 制定

institution [.insti'tju:ʃən] n.协会 制度 习俗 机构 创立

instruct [in'strʌkt] vt.教 指示 通知 命令 指导 [计算机] 指示

instruction [in'strʌkʃən] n.命令 指令 教学 教训 说明 须知

instrument ['instrumənt] n.仪器 工具 乐器 器械

insufficient [.insə'fiʃənt] adj.不足的 不适当的

insult ['insʌlt,in'sʌlt] vt. & n.侮辱 凌辱 (对人身的)损害

insurance [in'ʃuərəns] n.保险 保险费

insure [in'ʃuə] vt.给...保险 确保 vi.买保险

intellectual [.intil'ektʃuəl] n.知识分子 凭理智做事者 adj.智力的 聪明的

intelligence [in'telidʒəns] n.智力 理解力 情报 情报工作 情报机关

intelligent [in'telidʒənt] adj.聪明的 理智的

intend [in'tend] vt.想要 打算 计划 意指

intense [in'tens] adj.强烈的 紧张的 剧烈的 热情的 激烈的

intensity [in'tensiti] n.强烈 剧烈 强度

intensive [in'tensiv] adj.加强的 精耕细作的 强烈的 密集的 精细的 强调的 n.(语法)强调成份

intention [in'tenʃən] n.意图 意向 目的

intentional [in'tenʃənəl] adj.故意的 有意识的 策划的 企图的

interaction n.相互作用 相互影响 互动交流 干扰

interest ['intərist] n.兴趣 利益 利息 嗜好 股份 vt.使...感兴趣

interesting ['intəristiŋ] adj.有趣的 引人入胜的

interfere [.intə'fiə] vi.干涉 干预 妨碍 冲突

interference [.intə'fiərəns] n.干涉 干预 干扰 阻碍 [计算机] 干涉

interior [in'tiəriə] adj.内的 内地的 国内的 内部的 n.内部 内在

intermediate [.intə'mi:diət] adj.中间的 中级的 n.中间体 媒介物 调解人 vi.调解 干涉

internal [in'tə:nəl] adj.内的 国内的 内在的 身体内部的

international [.intə'næʃənəl] adj.国际的 世界(性)的 n.国际组织

interpret[in'tə:prit] vt.解释 说明 口译 翻译 演出 [计算机] 解释

interpretation [in.tə:pri'teiʃən] n.解释 口译 翻译 演出

interpreter [in'tə:pritə] n.译员 口译者 解释程序

interrupt [.intə'rʌpt] vt.打断 打扰 中止 妨碍 vi.插嘴 n.中断 [计算机] 中断

interruption [.intə'rʌpʃən] n.中断 打断 障碍物

interval ['intəvəl] n.间隔 休息 休息时间 (数学)区间 (音乐)音程 间距

interview ['intəvju:] n.接见 会见 面谈 访问 vt.接见 对...进行面谈(试) vi.面试 采访

intimate ['intimeit,'intimit] adj.亲密的 个人的 私人的 秘密的 n.密友 vt.透露 暗示

into ['intu:] prep.进 入 进入到 到...里

introduce [.intrə'dju:s] vt.介绍 引进 传入 采用 提出 提倡 放入

introduction [.intrə'dʌkʃən] n.介绍 引进 引言 导论 序言

invade [in'veid] vt.入侵 侵略 侵害 侵袭 拥入

invasion [in'veiʒən] n.入侵 侵略 侵犯

invent [in'vent] vt.发明 创造 捏造

invention [in'venʃən] n.发明 发明物 虚构 虚构物 创造 捏造

inventor [in'ventə] n.发明者 发明家 创造者

invest [in'vest] vt.投资 投入 花费 授与 赋予(某种性质) 包围 vi.投资

investigate [in'vestigeit] vt. & vi.调查 研究 [计算机] 研究

investigation [in.vesti'geiʃən] n.调查 调查研究

investment [in'vestmənt] n.投资 投资额 投入

invisible [in'vizəbl] adj.看不见的 无形的 n.隐形人(或物品)

invitation[.invi'teiʃən] n.邀请 招待 请柬 引诱 招致

invite [in'vait] vt.邀请 聘请 招待 要求 n.邀请

involve [in'vɔlv] vt.使卷入 使陷入 使忙于 牵涉 包含

inward ['inwəd] adj.里面的 内心的 向内的 亲密的 adv.向内 n.内部 内在

iron ['aiən] n.铁 烙铁 熨斗 坚强 (复数)铁制物 vt.烫(衣) 用铁装备 vi.熨烫(平) adj.刚强的

irregular [i'regjulə] adj.不规则的 不整齐的 不合法的 n.不合规格之物

island ['ailənd] n.岛 岛屿 孤立体 vt.建成岛屿 隔离

isolate ['aisəleit] vt.使隔离 使孤立 adj.孤立的 n.分离

issue ['iʃju:] n.问题 争论点 发行 发行物 期刊号 vi. & vt.发行 流出 造成...结果

it [it] pron.这 那 它

Italian[i'tæljən] adj.意大利 n.意大利人

Italy['itəli] n.意大利

item ['aitəm] n.条 条款 一条 项目 一件商品(或物品) adv.也

its [its] pron.( it的所有格)它的

itself [it'self] pron.它自己 它本身 自身

## J

jacket['dʒækit] n.短上衣 茄克衫

jail [dʒeil] n.监狱 监狱 拘留所 vi.监禁 vt.监禁 下狱

jam [dʒæm] n.果酱 拥塞之物 堵塞 窘境 v.挤 塞满 混杂 vt.使塞满 使堵塞

January ['dʒænjuəri] n.一月

Japan [dʒə'pæn] n.日本 日本国 日本漆 日本式漆器 日本式物品

Japanese [dʒæpə'ni:z] adj.日本的 日本人的 日语的 n.日本人 日语

jar [dʒɑ:] n.罐子 坛子 广口瓶 不和谐 刺耳声 震动 震惊 vi.发刺耳声 不协调 震动 vt.扰乱 使不稳定

jaw [dʒɔ:] n.颌 颚 说教 流言蜚语 v.喋喋不休 教训

jazz [dʒæz] n.爵士音乐 爵士舞(曲) 空扯的话 以及类似的东西 vt.使...活泼 把...演奏成爵士风格 vi.闲逛 跳爵士舞

jealous ['dʒeləs] adj.妒忌的 猜疑的 戒备的

jet[dʒet] n.喷气式飞机 喷嘴 喷射 喷出 黑玉大理石 vt.射出 喷射 vi.气流中喷出 adj.煤黑色的

jew [dʒu:] n.犹太人

jewel ['dʒu:əl] n.宝石 宝石饰物 v.镶以宝石

jewish ['dʒu:iʃ] adj.犹太人的

job [dʒɔb] n. 职业 工作 零活 职责 任务 vt. 投机买卖 营私舞弊 不公正地判罚 分包任务 vi. 工作 假公济私 做分销商或批发商 adj. 与工作有关的

join [dʒɔin] vt.参加 结合 联合 加入 连接

joint [dʒɔint] n.接头 接缝 关节 接合处 adj.联合的 共同的 合资的 连带的 v.连接 贴合

joke [dʒəuk] n.笑话 玩笑 vi.说笑话 开玩笑

jolly ['dʒɔli] adj. 快活的 愉快的 高兴的 宜人的 令人高兴的 v. 恭维 哄 开玩笑 戏虐 adv. 非常 极为

n. (尤英) 欢乐 欢庆 (复数) follies: 娱乐 乐趣

journal ['dʒə:nl] n.日报 期刊 杂志 日志

journalist ['dʒə:nəlist] n.记者 新闻工作者

journey ['dʒə:ni] n.旅行 旅程 行程 v.旅行

joy [dʒɔi] n.欢乐 喜悦 乐趣 乐事 vi.享受快乐

joyful ['dʒɔifəl] adj.十分喜悦的 快乐的

judge [dʒʌdʒ] n.法官 裁判员 v.断定 判断 估计

judgement ['dʒʌdʒmənt] n.意见 审判 判断

juice [dʒu:s] n.(水果等)汁 液 果汁 细胞液 内在本质 汽油 唾液 体液 vt.加液体 榨出...的汁液

July [dʒu(:)'lai] n.七月

jump [dʒʌmp] vi. & vt.跳 跳过 跳越 惊跳 暴涨 突升 n.跳跃 跳动 上涨

June [dʒu:n] n.六月 琼(女子名)

jungle ['dʒʌŋgl] n.丛林 密林 莽丛 adj.丛林的 蛮荒的

junior ['dʒu:njə] adj.年少的 下级的 后进的 n.晚辈 年少者 地位较低者 大学三年级学生

jury ['dʒuəri] n.陪审团 评委会 评奖团 adj.临时用的 vt.挑选

just [dʒʌst] adv.刚才 只是 正好 恰好 adj.公正的 公平的 恰当的

justice ['dʒʌstis] n.正义 公正 司法 上诉法庭的法官

justify ['dʒʌstifai] vt.证明...是正当的 替 ...辩护 n.对齐 [计算机] 调整

## K

keen [ki:n] adj.热心的 激烈的 敏锐的 强烈的 精明的 热忠于

keep [ki:p] vi.保持 坚持 保存 维持 继续 保有 记录 n.生计 维持 围墙 监狱

keeper ['ki:pə] n.看护人 饲养员 保管人

kettle ['ketl] n.水壶 水锅 坑穴

key [ki:] n.钥匙 关键 答案 按键 音调 adj.主要的 关键的 vt.上锁 调节 提供解答 键入 vi.使用钥匙

keyboard ['ki:bɔ:d] n.键盘 vt.用键盘输入

kick [kik] n. 踢 冲刺 反冲 钱包 反对 极大兴趣 vt. 踢 对自己生气 戒除 vi. 踢 反抗 充满活力的运作 反冲 冲刺

kid [kid] n.小孩 儿童 少年 小山羊 小山羊皮制品 vi. (山羊 羚羊)生幼崽 玩恶作剧 vt. 开玩笑 哄骗 嘲弄 adj. 相对年幼的

kill [kil] vi.杀死 扼杀 消磨 消减 破坏 n.杀 杀戮

kilogram ['kiləgræm] n.千克 公斤

kilometer ['kilə.mi:tə] n.千米 公里 =kilometre(略作km)

kind [kaind] n.种类 本质 adj.仁慈的 友好的 和蔼的

kindness ['kaindnis] n.仁慈 好意

king [kiŋ] n.国王 君主

Kingdom ['kiŋdəm] n.天国

kingdom ['kiŋdəm] n.王国 领域 界

kiss [kis] vt.吻 轻触 vi.接吻 轻触 n.吻 蛋白小饼干 小糖果

kitchen ['kitʃin] n.厨房 灶间

kite [kait] n.风筝 v.用空头支票骗人 使上升

knee [ni:] n.膝 膝盖 膝关节 像膝盖的东西 护膝 屈膝鞠躬 vt.用膝盖顶

kneel [ni:l] vi.跪 跪下 跪着

knife [naif] n.小刀 刀 餐刀 v.用刀切割 如刀割般

knit [nit] vt.把...编结 编织 密接 结合 皱眉 vi.编织 变得紧凑 愈合 n.编织

knob [nɔb] n.门把 拉手 旋纽 瘤

knock [nɔk] vi. & vt.敲 击 打 互撞 攻击 n.敲 敲打 敲门

knot [nɔt] n.(绳的)结 (树的)节 难题 一小群 vi.打结 vt.把...打成结 系牢 捆扎

know [nəu] vt.知道 认识 通晓 n.知识 了解

knowledge ['nɔlidʒ] n.知识 学识 知道 了解

## L

lab [læb] n.实验室 研究室

label ['leibl] n.标签 签条 标记 符号 商标 称谓 [计算机]标记 DOS命令 加上或修改磁盘标签 vt. 贴标签于 分类 把...称为

laboratory ['læbrətɔ:ri] n.实验室 研究室 实验大楼

labor ['leibə] n.工作 劳动 劳力 劳方 vt.苦干 劳动 分娩 阵痛 致力于 vi.过分详尽阐述 adj.劳工的 劳动的

labour ['leibə] n.劳动 工作 劳工 Labour(英国)工党

lace [leis] n.鞋带 系带 花边 饰带 缎带 v.结带子 饰以花边

lack [læk] vi. & vi. &n.缺乏 不足 无

ladder ['lædə] n.梯子 阶梯 梯状物

lady ['leidi] n.女士 夫人 贵妇人

lag [læg] vi. 走得慢 落后 vt. 落后 投掷 盖上桶盖 耽搁 n. 落后 滞后 落后的人 (两事件之间的)时间间隔 耽搁 桶盖 adj. 最后的

lake[leik] n.湖

lamb [læm] n.羔羊 小羊 羔羊肉 温顺的人 v.产羊羔

lame [leim] adj.跛的 瘸的 残废的 无说服力的 vt.使...成残废 使...变无用 n.金属薄板

lamp [læmp] n.灯 发光或反射光的天体 灵光

land [lænd] n.陆地 国土 土地 vi.上岸 使靠岸 卸货 得到 登陆

landing ['lændiŋ] n.上岸 登陆 着陆 楼梯平台 动词land的现在分词

landlady ['lænd.leidi] n.女房东 女地主

landlord ['lændlɔ:d] n.地主 房东 店主

lane [lein] n.(乡间)小路 小巷 行车道 跑道

language ['læŋgwidʒ] n.语言 语言课程

lantern ['læntən] n.提灯 灯笼 [建筑学]天窗 屋顶气窗

lap [læp] n.膝部 膝盖 一圈 一周 下摆 舐声 vt.包围 抱...在膝上 重叠 舔 拍打 泼溅 vi.重叠 围住 轻拍 舔

large [lɑ:dʒ] adj.大的 巨大的 广泛的 强烈的 普遍地 大量地

largely **largely** adv.大部分 大量地 主要地

laser ['leizə] n.激光 镭射

last [lɑ:st] adj.最后的 末尾的 最近的 晚了 迟到了 adv.最后 最新 n.最后来的人(或发生的事) v.持续 持久 耐久

late [leit] adj.迟的 晚的 已故的 adv.迟 很迟 晚 最近地

lately ['leitli] adv.最近 不久前

later ['leitə] adv.后来 过一会儿 稍后 后来 迟到地 adj.更迟的 后面的

Latin ['lætin] adj.拉丁的 拉丁人的 拉丁语系国家的 n.拉丁语 拉丁人

latter ['lætə] adj.(两者中)后者的 后面的

laugh [lɑ:f] vi.笑 发笑 n.笑(声) 笑料 (复数)娱乐

laughter ['lɑ:ftə] n.笑 笑声

launch [lɔ:ntʃ] vt. & n.发射 投射 发动 下水 开始 升天 汽艇

laundry ['lɔ:ndri] n.洗衣房 洗衣店 要洗的衣服 洗衣

lavatory ['lævətəri] n.盥洗室 厕所

law [lɔ:] n.法律 法令 起诉 法则 规律 vi.起诉

lawn [lɔ:n] n.草地 草坪 草场 上等细麻布

lawyer ['lɔ:jə] n.律师 法学家

lay [lei] lie的过去式 vt. 置放 铺设 设置 击 产卵 使平静 打赌 压平 罚款 设计 vi. 生蛋 打赌 全力以赴

n. 隐蔽处 计划 价格 生蛋 叙事诗 诗歌 adj. 世俗的 外行的

layer ['leiə] n.层 层次 铺设者 vi.分层 vt.将某物堆积成层 n.产卵鸡 [园艺]压条 vt.[园艺]用压条法使(嫩枝)生根

layout ['leiaut] n.布局 安排 设计

lazy ['leizi] adj.懒惰的 懒散的

lead [led,li:d] vt.为...带路 领导 引导 带领 以铅接合 过...(的生活) n.铅 铅制品 领引 榜样

leader ['li:də] n.领袖 领导人 首领

leadership['li:dəʃip] n.领导 领导才干

leading ['li:diŋ] adj. 指导的 领导的 最主要的 在前的 vbl. 领导 引领 以铅接合 n. 领导 疏导 铅板 [计算机] 行间空白

leaf [li:f] n.叶 叶子 树叶 页 活动桌板 花瓣 vt.翻页 vi.长叶子 翻页浏览

league [li:g] n.同盟 联盟 联合会 社团

leak [li:k] vi.漏 泄露 漏渗 n.漏洞

lean [li:n] vi.倾斜 屈身 倚靠 依赖 vt.使倾斜 n.瘦肉 倾斜 弯曲 adj.瘦的 贫乏的 歉收的

leap [li:p] vi.跳 跃 跳跃 猛冲 vt.跃过 n.跳跃 剧增 急变

learn [lə:n] vi.学 学习 了解 习得 vt.学 学习 获悉

learned ['lə:nid] adj.有学问的 博学的 学术上的 learn的过去式和过去分词

learning ['lə:niŋ] n.学习 学问 知识 动词learn的现在分词

least [li:st] adj.最少的 最小的 n. & adv.最少 最小

leather ['leðə] n.皮革 皮革制品 adj.皮革制的 vt.用皮革覆盖 鞭打

leave [li:v] vi.离去 生出叶子 vt.离开 告假 遗留 听任 使处于某种状态 n.许可 准假 告别

lecture ['lektʃə] n. &vi.演讲 讲课 教导 教训 斥责

left [left] adj.左边的 ad.在左边 n.左边 p.p.of leave

leg [leg] n.腿 腿部 支柱 裤脚管 vi.用脚走路

legal ['li:gəl] adj.法律的 合法的 法定的

legend ['ledʒənd] n.传说 传奇 传奇人物 铭文 (地图或图片等的)说明 图例

leisure ['li:ʒə] n.空闲时间 悠闲 闲暇 休闲

lemon ['lemən] n.柠檬 柠檬树

lend [lend] vt.把...借给 贷(款) 给予 增添 vi.贷款

length [leŋθ] n.长 长度 一段 篇幅

lens [lenz] n.透镜 镜片 镜头

less [les] adj.更少的 更小的 ad.更少地 更小地 n.少量 次要 prep.减

lessen ['lesn] vt.减少 减轻 缩小 变小

lesson ['lesn] n.功课 课 课程 教训 vt.给...上课 教训

lest [lest] conj. & prep.惟恐 以免

let [let] vt.允许 让 使 释放 vi.出租 承包n.障碍 擦网球

letter ['letə] n.信 证书 字母 v.写下 印刷 n.租赁人

level ['levl] n. 水平面 水平 等级 标准 层次 a. 水平的 平坦的 齐平的 稳定的 合理的 满满的 vt. 压平 使同等 瞄准 夷为平地 推翻 指责 vi. 趋向平稳 瞄准 诚恳待人 使平等

lever ['levə] n.杆 杠杆 控制杆 似杠杆之工具 vt.撬开 使用杠杆

liable ['laiəbl] adj.易于...的 可能的 有义务的 应负责的 有 ...倾向

liar ['laiə] n.说谎的人

liberal ['libərəl] adj.心胸宽大的 慷慨的 无偏见的 不拘泥的 宽大的 开明的 n.自由主义者 宽容大度的人 (Liberal)自由党人

liberate ['libəreit] vt.解放 释放 使获自由 v.解放 使获得自由 释出 放出

liberation [.libə'reiʃən] n.解放 解放运动(为获得平等权利和地位的行为)

liberty ['libəti] n.自由 释放 许可

librarian [lai'brɛəriən] n.图书馆馆长

library ['laibrəri] n.图书馆 藏书

license ['laisns] n.许可 批准 批准 执照 许可证 vt.准许 特许

lick [lik] vt.舔 舔吃 连续击打 击败 vi.用舌头舔 拍打 n.舔 少许 打

lid [lid] n.盖子 盖 囊盖

lie [lai] vi.躺 平放 位于 说谎 欺骗 n.谎言

lieutenant [lu:'tenənt] n.陆军中尉 副职官员 助理人员

life [laif] n.生命 一生 寿命 生活 人生 生物 灵魂 无期(徒刑) adj.活的 终身的 生动的

lifetime ['laiftaim] n.一生 终身 寿命 使用期限

lift [lift] vt.提起 提高 升高 举起 还清 取消 偷窃 运送 vi. 提升 n. 电梯 举起 帮助 提升 电梯 载货量 扒窃 搭便车

light [lait] n. 光 光线 灯 光源 启发 众所周知的 通光口 眼光 adj. 轻的 不重要的 容易的 明亮的 淡色的 少量的 vi. 点燃 着火 变亮 下马 落下 碰巧发生 vt. 点燃 照亮 adv. 轻地 轻便的

lighten ['laitn] vt.照亮 使明亮 点亮 变亮 减轻

lightly ['laitli] adv.轻轻地 轻松地 轻快地 轻蔑地 轻率地 轻微地

lightning ['laitniŋ] n.闪电 闪电放电 adj.闪电般的 快速的 vi.打闪

like [laik] vt. 喜欢 喜爱 希望 愿意 想 adj. 相似的 同样的 prep. 像 如同 像要 n. 爱好 类似的人(或物) adv. 大概 和...一样 conj. 如同

likely ['laikli] adj.可能的 合适的 可信的 有前途的 有吸引力的 adv.很可能

likewise ['laikwaiz] ad.同样地 也 又 此外 (表示感觉相同)我也是

limb [lim] n.肢 臂 翼 树枝 四肢 枝干 边缘 vt.切断手足 切断树干

lime [laim] n.石灰 酸橙 酸橙树 粘鸟胶 钙 vt.施石灰 涂上粘性物质 沾上粘鸟胶 adj.与石灰或石灰石相关的

limit ['limit] n.限度 限制 范围 vt.限制 限定

limitation [.limi'teiʃən] n.限制 限度 局限

limited ['limitid] adj.有限的 被限制住的 缺乏创新性的

line [lain] n. 线 排 路线 线条行 航线 场界 皱纹 家族 v. 用做衬里 排成一行 顺...排列 vi. 排成一行 顺...排列 划线于

linen ['linin] n.亚麻布 亚麻线 亚麻织物 亚麻制品 adj.亚麻布制的 亚麻的

liner ['lainə] n.班船 班轮 班机

link [liŋk] vt.有环连接 连接 联系 n.环 环节 联系 (链的)环 圈

lion ['laiən] n.狮子 勇猛的人

lip [lip] n.嘴唇

liquid ['likwid] n.液体 a.液体的 液态的

liquor ['likə] n.酒 溶液 液剂

list [list] n.表 明细表 目录 名单 条纹 倾斜 列举 <古>意向 vt.列出 列表 (使)倾斜 <古>使高兴 愿意 <古>听

listen ['lisn] vi.听 留神听 听从 倾听 n.听 倾听

listener ['lisənə] n.听者 听众之一

liter ['li:tə] n.升 公升(容量单位) =litre(英)

literary ['litərəri] a.dj文学(上)的

literature ['litərətʃə] n.文学 文献

little ['litl] adj.小的 少 幼小的 adv.毫不 少量地 n.几乎没有(指数量或程度等)

live [laiv,liv] vi.居住 活 过着 adj.活的 生动的 精力充沛的 直播的

lively ['laivli] adj.活泼的 活跃的 栩栩如生的 真实的 逼真的

liver ['livə] n.肝 肝脏

living ['liviŋ] adj.活的 生动的 n.生活 生计 动词live的现在分词

living-room ['liviŋ'ru:m] n.起居室

load [ləud] vt.装 装满 装载 装填 担负 给(枪)装子弹 [计算机] 装入 n.负载 负荷 重担 装载量

loaf [ləuf] n.一条面包 一个面包 条 块 面包 vi.游手好闲 虚掷光阴

loan [ləun] n.贷款 暂借 借出 债权人 vt.借出 借给 供应货款

local ['ləukəl] adj.地方的 局部的 地方性的 当地的 n.当地人

locate [ləu'keit] vt.探明 找出 查出 设于 位于 建立 vi.建立 住下来

location [ləu'keiʃən] n.位置 场所 地点 定位 外景拍摄地

lock [lɔk] n.锁 刹车 水闸 牢牢紧握 一束 一缕 vt.锁上 锁住 抓牢 使固定 vi.锁(住) 纠结 过闸

locomotive [.ləukə'məutiv] n.火车头 机车 adj.运转的 火车头的 移动的

lodge [lɔdʒ] vi.暂住 借宿 投宿 临时住宿 安顿 n.小屋 门房 支部

loft [lɔ:ft] n.阁楼 顶楼

log [lɔg] n.原木 木料 记录 日志 计程仪 vt.伐木 切 记录 航行 度过(时间) vi.采伐

logic ['lɔdʒik] n.逻辑(学) (合理的)推理 逻辑性

logical ['lɔdʒikəl] adj.逻辑的 符合逻辑的 有推理能力的

lonely ['ləunli] adj.孤独的 荒凉的 寂寞的

long [lɔŋ] adj.长的 远的 很久的 冗长的 adv.长久 长期地 远地 vi.渴望 热望 极想念 n.长时间 长音节

look [luk] vi.看 显得 注意 朝着 好象 vt.打量 注视 用眼神或脸色表示 期待 n.看 神色 样子 脸色

loop [lu:p] n.圈 环 环孔 弯曲部份 回路 vt.使...成环 以圈结 以环连结 vi.形成环 环形地行动

loose [lu:s] adj.松的 宽松的 不牢固的 不精确的 n.解放 放任 放纵 v.释放 放枪 开船

loosen ['lu:sn] vt.解开 使松驰 放松 松开 解除(便秘等) 放宽 vi.变松 松弛

lord [lɔ:d] n.贵族 上帝 基督 主

lorry ['lɔri] n.运货汽车 卡车

lose [lu:z] vt.失去 遗失 迷失 输掉 失败 损失 摆脱 vi.丢失 失败 [计算机] 失去

loss [lɔs] n.遗失 损失 失败 丧失 损耗 亏损

lot [lɔt] n.许多 大量 签 抽签 阄 小组 场地 命运 adv.很 非常 pron.大量 vt.分配 划分

loud [laud] adj.响亮的 吵闹的 大声的 难闻的 adv.响亮地 大声地

loudly ['laudli] adv.大声地 高声地 花俏地

loudspeaker ['laud'spi:kə] n.扬声器 喇叭

love [lʌv] vt.爱 爱慕 热爱 喜欢 vi.爱 n.爱 恋爱 爱情 情人

lovely ['lʌvli] adj.可爱的 令人愉快的 n.漂亮的女人 美好的东西

lover ['lʌvə] n.爱好者 情人 情侣

low [ləu] adj.低的 矮的 低下的 低点(价) 深度 牛叫声 adj. 低的 消沉的 低等的 粗俗的 衰弱的 vi.牛叫

lower ['ləuə,'lauə] adj. 较低的 低的 下级的 下层的 vt. 放下 降低 跌落 减弱 贬低 vi. 皱眉 降低 阴沉 变少 n. 皱眉 下级 阴沉的样子 前兆

loyal ['lɔiəl] adj.忠诚的 忠心的

loyalty ['lɔiəlti] n.忠诚 忠心

luck [lʌk] n.运气 好运 幸运 吉祥之物 v.侥幸成功

lucky['lʌki] adj.幸运的 带来好运的 吉祥的

luggage ['lʌgidʒ] n.行李 皮箱 皮包

lumber ['lʌmbə] n.木材 木料 制材 v.伐木 vi.缓慢地移动 vt.拖累

lump [lʌmp] n.团 块 块状 肿块 瘤 很多 愚笨的人 惩罚 vt.聚成团 合并考虑 同等对待或分类 笨拙地移动 结块 忍受 vi.使成块状 沉重地移动 adj.整体的 全部的 块状的

lunch [lʌntʃ] n.午餐 (美)便餐 vi.吃午餐 vt.请客吃午餐

lung [lʌŋ] n.肺脏 肺 呼吸器官

luxury ['lʌkʃəri] n.奢侈 奢华 奢侈品

## M

machine [mə'ʃi:n] n.机器 机械 机械装置 机械般工作的人 核心 vt.以机器制造 [计算机] 机器

mad [mæd] adj.发疯的 恼火的 狂热的 着迷的 生气的 v.发怒 n.发怒

madam ['mædəm] n.夫人 女士 太太 老鸨

magazine [.mægə'zi:n] n.杂志 期刊

magic ['mædʒik] n.魔法 巫术 戏法 adj.有魔力的 神奇的 vt.使用魔(变出等)

magnet ['mægnit] n.磁铁 磁石 磁体 有强大吸引力的人或物

magnetic[mæg'netik] adj.磁的 有吸引力的 催眠术的

magnificent [mæg'nifisnt] adj.壮丽的 华丽的 宏伟的 极好的

maid [meid] n.女佣 女仆 侍女 少女

mail [meil] n.邮件 邮政 vt.邮寄

main [mein] adj.主要的 最重要的 n.主要部分 干线 (水、煤气、电等)供应系统 体力

mainly ['meinli] adv.主要地 大体上

mainland ['meinlənd] n.大陆 本土

maintain [mein'tein] vt.维持 赡养 维修 保养 坚持 断言

maintenance ['meintinəns] n.维持 保持 维修 生活费用 供给 抚养 主张

major ['meidʒə] adj.较大的 主要的 较多的 大部份的 n.专业 主修 成年人 陆军少校 巨头 vi.主修

majority [mə'dʒɔ:riti] n.多数 大多数

make [meik] vt.使 做 制造 vi.表现 前往 生产 n.牌子 型号 性格 生产

male [meil] a.男的 雄的 n.男子 雄性动物

man [mæn] n.男人 人 人类 vt.为...配备人手 使...振奋

manage ['mænidʒ] vt.管理 设法 对付 处理 维持 达成 经营 vi.管理 达成

management ['mænidʒmənt] n.管理(层) 经营 处理

manager ['mænidʒə] n.经理 管理人

mankind [mæn'kaind] n.人类 男人

manly ['mænli] adj.男子气概的 果断的 强壮的 适于男人的 adv.男子般地

manner ['mænə] n.方式 态度 礼貌 举止 习俗

manual ['mænjuəl] adj.体力的 手工的 n.手册 指南 键盘

manufacture [.mænju'fæktʃə] vt.制造 加工 vi.参与制造 n.制造 产品 制造业

manufacturer [.mænju'fæktʃərə] n.制造商 制造厂

many ['meni] adj.许多的 pron.许多人或物 n.许多人 许多

map [mæp] n.地图 图 天体图 基因图谱 vt.绘制地图 计划 [计算机] 存储区分配图

marble ['mɑ:bl] n.大理石 vt.使有大理石的花纹

march [mɑ:tʃ] n.(March)三月 示威游行 vt.使前进 使行军 vi.行军 进军 游行示威

margin ['mɑ:dʒin] n.页边的空白 边缘 差额 利润 vt.使围绕于 加边缘

marine [mə'ri:n] adj.海的 海上的 海事的 船舶的 航海的 n.(海军)士兵或军官 海景画

mark [mɑ:k] n. 斑点 污迹 记号 标记 分数 烙印 纪录 vt. 标明 做标记于 留意 打分数 留下污痕 使...明显

vi. 做标志 记分 留下印记

market ['mɑ:kit] n.市场 集市 销路 交易 推销地区 vt.在市场上交易 使...上市 vi.在市场上买卖

marriage ['mæridʒ] n.结婚 婚姻 婚礼

married ['mærid] adj.已婚的 婚姻的

marry ['mæri] vt.娶 嫁 vi.结婚 结合

marvelous ['mɑ:viləs] adj.奇迹般的 了不起的 令人惊异的 不平常的

Marxism ['mɑ:ksizəm] n.马克思主义

Marxist ['mɑ:ksist] adj.马克思主义的

mask [mɑ:sk] n.面具 伪装 口罩 掩饰 面膜 vt.掩饰 戴面具 化装 [计算机] 掩码

mass [mæs] n.众多 团 群众 块 大量 adj.群众的 大规模的 v.集中 (Mass) n.弥撒 弥撒曲 pl.群

master ['mɑ:stə] n.主人 能手 硕士 专家 adj.主人的 主要的 精通的 vt.征服 控制 精通

masterpiece ['mɑ:stəpi:s] n.杰作 名著

mat [mæt] n. 席子 草席 垫子 垫 丛 衬边 消光面 vt. 遮盖 保护 用垫子装饰 使纠缠 铺席于...上 使...无光泽 给(照片)镶框 vi. 缠结

match [mætʃ] n.比赛 竞赛 对手 火柴 般配的人 配偶 v.使...相配 与...竞争 相配

mate [meit] n.伙伴 同事 配偶 助手 v.使...配对 交配 使...一致 结伴

material [mə'tiəriəl] n.材料 原料 物质 素材 题材 资料 a.物质的 实体的 身体上的 肉体的 重要的 关键的

materialism [mə'tiəriəlizəm] n.唯物主义 唯物论 实利主义

mathematical [.mæθə'mætikl] adj.数学的 数学上的 精确的 可能性极小的

mathematics [.mæθi'mætiks] n.数学

maths [mæθs] n.(英)数学 =mathematics(英)

matter['mætə] n.事情 物质 原因 vi.要紧 有关系

mature [mə'tjuə] adj.成熟的 到期的 充分发育的 考虑周到的 vt.使…成熟 长成 vi.长成 成熟 到期

maximum ['mæksiməm] n.最大量 极点 极大 adj.最大的 最高的 最大极限的

may [mei, me] v.aux.可能 可以 祝愿 n.(May)五月 青春

maybe ['meibi] adv.大概 或许 也许 n.不确定性

mayor ['mɛ] n.市长

me [mi:] pron.(宾格)我

meadow ['medəu] n.草地 牧场 牧草地

meal [mi:l] n.膳食 一餐 谷物粗粉 vi.进餐

mean [mi:n] vt. 作...解释 意指 意谓 想要 意欲 n. 平均值 平均数 中部 adj. 低劣的 卑贱的 卑鄙的 自私的吝啬的 平均的 中等的

meaning ['mi:niŋ] n.意义 意思 含义 意图 adj.意味深长的

means [mi:nz] n.方法 手段 工具 折中点 物质财富

meantime ['mi:n'taim] n.其时 其间 adv.当时 与此同时

meanwhile ['mi:n'wail] adv.同时 当时 n.其时 其间

measurable ['meʒərəbəl] adj.可测量的

measure ['meʒə] vt.量 测量 n.分量 措施 办法 量度 尺寸

measurement ['meʒəmənt] n.衡量 测量 尺寸 大小

meat [mi:t] n.肉 主要或重要部分

mechanic [mi'kænik] n.技工 机械 机修工 adj.手工的

mechanical [mi'kænikəl] a.机械的 力学的 呆板的 n.(供制版用的)样书 版面设计

mechanically [mi'kænikəli] adv.机械地

mechanics [mi'kæniks] n.力学 机械学 (技术的 操作的)过程 手法 技术性细节

medal ['medl] n.奖章 勋章 纪念章 vi.获得奖章

medical ['medikəl] adj.医学的 内科的 医疗的 药物的 需治疗的

medicine ['medisin] n.内服药 医学

Mediterranean [.meditə'reinjən] n.地中海 adj.地中海(的)

medium ['mi:diəm] n.媒质 方法 媒介 中间 adj.中等的 适中的

meet [mi:t] vt.遇见 引见 经历 结识 对付 满足 支付 vi.相遇 集合 交锋 相交 n.集会 比赛 运动会 adj.适宜的 合适的

meeting ['mi:tiŋ] n.聚集 会合 会见 会议

melon ['melən] n.瓜 甜瓜

melt [melt] vi.融化 熔化 溶解 软化 渐渐混合 vt.使融化 n.熔化 熔化物

member ['membə] n.成员 会员 部位

memorial [mi'mɔ:riəl] adj.纪念的 追悼的 记忆的 n.纪念碑(堂) 纪念仪式

memory ['meməri] n.记忆 回忆 存储 内存

mend [mend] vt.修理 修补 缝补 修改 改进 vi.改过自新 改进 痊愈 n.修补 改进 修补处

mental ['mentl] adj.智力的 精神的 脑力的 精神错乱的 传心术的 [解剖学] 颏的

mention ['menʃən] vt. & n.提及 说起

menu ['menju:] n.菜单 饭菜 菜肴

merchant ['mə:tʃənt] n.商人 零售商 店主 专家 adj.商业的 vt.做买卖

mercury ['mə:kjuri] n.水银 汞 (Mercury)水星

mercy ['mə:si] n.仁慈 慈悲 恩惠 怜悯 宽恕 幸运 adj.仁慈的 宽恕的

mere [miə] adj.仅仅的 纯粹的 微小的 n.小湖 边界 地标

merely ['miəli] adv.仅仅 只不过

merit ['merit] n.长处 优点 功过 功绩 价值 v.博得 值得或应得

merry ['meri] adj.欢乐的 愉快的 敏捷的

mess [mes] n.混乱 混杂 肮脏 乱七八糟 一堆食物 集体用膳人员 大量 v.供给食物 把...弄乱

message ['mesidʒ] n.信息 消息 启示 要点 v.传递信息 通讯

messenger ['mesindʒə] n.送信者 信使 先驱

metal ['metl] n.金属 质料 金属制品 vt.以金属覆盖

meter ['mi:tə] n.计量器 计 表 公尺 韵律 vt.用表测量[计量 记录]

method ['meθəd] n.方法 办法 教学法 条理 类函数 类成员函数(计算机)

metre ['mi:tə] n.米 公尺 计量器 (诗的)韵律 格律 =meter(美)

metric ['metrik] adj.公制的 米制的 十进制的 n.度量标准 公制 米制 十进制

microcomputer ['maikrəukəm'pju:tə] n.微型计算机 微机

microphone ['maikrəfəun] n.话筒 麦克风 扩音器

microscope ['maikrəskəup] n.显微镜

midday ['middei] n.正午 中午

middle ['midl] n.中部 中间 腰 中项 adj.中部的 中等的 中间的 中期的

midnight ['midnait] n.午夜 子夜 夜半 漆黑

midst [midst] n.中部 中间 当中 prep.(=amidst)在...当中

might [mait] v.aux.可能 会 也许 n.力量 威力 能力 v.或许 可以

mild [maild] adj.和缓的 温柔的 味淡的 轻微的 温和的 适度的

mile [mail] n.英里 大量 海里 一英里赛跑

military ['militəri] adj.军事的 军人的 n.军队

milk [milk] n.乳 乳状物 牛奶 vt.挤(奶) 产乳

mill [mil] n.磨坊 制造厂 磨粉机 密尔：一种货币单位 使陷于困境 vt.磨细 vi.乱转 磨细

millimetre ['mili.mi:tə] n.毫米 =millimeter(美)

million ['miljən] num.百万 n.许多 百万 (the millions)群众 adj.百万的 无数的

mind [maind] n.头脑 理智 记忆 思想 主意 心意 vt.留心 注意 介意 照顾 vi.留心 关心

mine [main] pron.我的 n.地雷 矿山 宝库 vt.采矿 开凿 暗中破坏 埋地雷 vi.挖掘

miner ['mainə] n.矿工

mineral ['minərəl] n.矿物 矿石 adj.矿物的

minimum ['miniməm] n.最小量 最低限度 adj.最低的 最小的

minister ['ministə] n.部长 大臣 公使 vi.照顾 给予帮助

ministry ['ministri] n.(政府的)部 (全体)牧师 牧师工作

minor ['mainə] adj.较小的 较次要的 二流的 未成年的 n.未成年人 辅修科目 [音乐]小调 vi.辅修

minority [mai'nɔ:riti] n.少数 少数民族 未成年

minus ['mainəs] adj.负的 减的 prep.减(去) n.减号 负数 缺点

minute ['minit] n.分 分钟 一会儿 会议记录 adj.微小的 不重要的 详细的 vt.记录

miracle ['mirəkl] n.奇迹 令人惊奇的人

mirror ['mirə] n.镜子 vt.反映 反射

miserable ['mizərəbl] adj.痛苦的 悲惨的 贫乏的

mislead[mis'li:d] vt.使误入岐途 误导 使产生错误印象 欺骗

miss [mis] n.小姐 vt.想念 惦念 漏掉 错过 未看到

missile ['misail] n.发射物 导弹

missing ['misiŋ] a.缺掉的 失去的 失踪的

mission ['miʃən] n.使命 任务 使团 代表团 传教 vt.传教 [计算机] 使命

mist [mist] n.薄雾 迷蒙 朦胧不清 vt.使...模糊 蒙上雾 vi.使...模糊 下雾

mistake [mi'steik] n.错误 误会 vi.误解 弄错 犯错 vt.弄错 误认

Mister ['mistə] n.先生

mistress ['mistris] n.女主人 夫人 主妇 情妇

misunderstand ['misʌndə'stænd] vt.误解 误会 曲解

mix [miks] vt.使混合 混淆 掺入 合成 交往 联系 n.混合(物) 融合 [计算机] 混合

mixture ['mikstʃə] n.混合 混合物

moan [məun] n.呻吟声 悲叹声 抱怨声 vi.呻吟 v.抱怨 呻吟

mobile ['məubail] adj.运动的 流动的 可移动的 易变的 迁徙的 n.(可随风飘动的)悬挂装置 可动雕塑

mode [məud] n.方式 样式 模式 风格 时兴 [音乐] 调式

model ['mɔdl] n.模型 模式 模特儿 模范

moderate ['mɔdəreit,'mɔdərit] a.温和的 有节制的 适度的 稳健的 中等的 v.节制 使...稳定 使...缓和 n.稳健的人

modern ['mɔdən] a.现代的 近代的 时髦的 n.现代人 时尚人士

modest ['mɔdist] adj.有节制的 谦虚的 适度的 端庄的

modify ['mɔdifai] vt.更改 修改 修饰 调正 缓和 vi.修改

moist [mɔist] adj.湿润的 潮湿的 多雨的 (果物)多汁的 [医] 有分泌物的 湿性的 n.潮湿

moisture ['mɔistʃə] n.潮湿 湿气 温度

molecule ['mɔlikju:l] n.分子 克分子

momen ['məumənt] t n.片刻 瞬间 时刻 准确时刻 非常重要的时机 力矩

Monday ['mʌndi, 'mʌndei] n.星期一

money ['mʌni] n.货币 金钱 财富

monitor ['mɔnitə] n.班长 监视器 监听员 [计算机]显示器 监视器 v.监视 监听 监督

monkey ['mʌŋki] n.猴子 猿 v.胡闹 乱弄

month [mʌnθ] n.月 月份

monthly ['mʌnθli] adj.每月的 持续一个月的 每月发生的 adv.每月 按月 n.月刊 (复数monthlies)月经

monument ['mɔnjumənt] n.纪念碑 纪念馆 历史遗迹

mood [mu:d] n.心情 情绪 语气

moon [mu:n] n.月球 月亮 卫星

moral ['mɔrəl] adj.道德(上)的 合乎道德的 n.品行 寓意 道德

more [mɔ:] a.更多的 程度较大的 更大的 adv.更 多 更多 进一步 n.更多

moreover [mɔ:r'əuvə] adv.再者 加之 此外 而且

morning ['mɔ:niŋ] n.早晨 上午 开端

mortal ['mɔ:tl] adj.终有一死的 人世间的 致命的 终有一死的 极度恐怖的 n.一个人

mosquito [məs'ki:təu] n.蚊子

most [məust] adj.最多的 大多数的 几乎全部的 adv.最 很 最多 极其 n.最多 最大 pron.大多数

mostly ['məustli] adv.主要的 大部分 大概

mother ['mʌðə] n.母亲 妈妈 adj.母亲的 母体的 vt.生育 像母亲一样关爱

motion ['məuʃən] n.运动 打手势 提议 示意 移动 动作 大便 v.运动 向...打手势 示意

motivate ['məutiveit] vt.促动 激励 激发(兴趣或欲望) 给与动机

motive ['məutiv] n.动机 目的 主题 adj.发动的 运动的 积极的 动机的 vt.促使 使有动机

motor ['məutə] n.发动机 机动车 马达 汽车 adj.机动的 有引擎的 汽车的 v.(用引擎)驱动

mould [məuld] n.模子(样板 形状 霉菌 土 类型) 模型 vt.浇铸 模压(铸造 发霉 形成 轮廓相符合) vi.发霉

mount [maunt] vt.登上 爬上 提升 装备 vi.增长 骑上(马) n.山峰 乘用马 框 衬纸

mountain ['mauntin] n.山 山岳 山脉 大量

mourn [mɔ:n] vi.哀痛 哀悼 忧伤 服丧

mouse [maus] n.老鼠 耗子 鼠标 胆小如鼠的人 乌青色的肿块 vi.捕鼠 蹑手蹑脚行动 vt.仔细寻找

mouth [mauθ] n. 嘴 口 口腔 开口 需要供养的人 夸张说话的趋势 (河流等)入口 容器的开口 vt. 装腔作势说话 送进嘴里 喃喃自语 用唇语说 vi. 傲慢地说 嘟嘴扮鬼脸

mouthful ['mauθful] n.满口 一口 少量

move [mu:v] vt. 使移动 使...感动 推动 (有时也意译为"跳槽") n. 移动 迁居 步骤 动作 [计算机] 移动

vi. 移动 搬家 采取行动

movement ['mu:vmənt] n.动作 活动 移动 运动 变动 趋势 通便 运转装置 [音] 乐章

movie ['mu:vi] n.电影 电影院

much [mʌtʃ] adv.非常 很 adj.许多的 pron. & n.许多

mud [mʌd] n.软泥 泥浆 恶意诽谤和攻击 vt.弄脏 污蔑

muddy ['mʌdi] adj.多泥的 泥泞的 肮脏的 混浊的 v.弄脏 使...混浊 使迷惑

mug [mʌg] n.大杯 脸 一杯的量 恶棍 vi.扮鬼脸 vt.攻击 打劫

multiple ['mʌltipl] adj.复合的 并联的 许多 多种多样的

multiply ['mʌltiplai] vt.使增加 乘 vi.扩大 繁衍 做乘法 adv.多样地 多重地

murder ['mə:də] n.谋杀 凶杀 危险物 vt.谋杀 凶杀 折磨 损坏 vi.谋杀 凶杀

murderer['mə:dərə] n.杀人犯 凶手

muscle ['mʌsl] n.肌肉 肌 体力 vt.用力挤 vi.用力进行

museum [mju:'ziəm] n.博物馆 展览馆

mushroom ['mʌʃrum] n.蘑菇 菌类植物 暴发户 vi.扩张 迅速增加 采蘑菇

music ['mju:zik] n.音乐 乐曲 乐谱 伴奏 美妙的声音

musical ['mju:zikəl] adj.音乐的 和谐的 n.音乐片

musician [mju:'ziʃən] n.音乐家 作曲家

must [mʌst]v.aux.必须 必然要 n.必须 一定 绝对 发霉 果汁 麝香

mute [mju:t]adj.缄默的 哑的 无声的 n.哑巴 默音字母 弱音器 vt.减音 减弱 vi.(鸟)排泄

mutter ['mʌtə] vi.轻声低语 喃喃自语 作低沉声 抱怨 出怨言 n.喃喃低语

mutton ['mʌtn] n.羊肉

mutual ['mju:tʃuəl] adj.相互的 共同的

my [mai]pron.我的 adj.我的(I的所有格形式) int.哎呦(用于表示惊讶的感叹词)

myself [mai'self] pron.我自己 我亲自

mysterious [mis'tiəriəs] adj.神秘的 难以理解的 不可思议的

mystery ['mistəri] n.神秘 秘密 神秘的人或事物

## N

nail [neil] n.钉子 指甲 vt.钉 使牢固 抓捕 敲定

naked ['neikid] adj.裸体的 无遮敝的 明白的 手无寸铁的

name [neim]n.名字 名称 姓名 名誉 vt.说出 命名 称呼 任命 adj.有名的 著名的

namely ['neimli] adv.即 也就是

nap [næp]n.小睡 打盹 瞌睡 绒毛表面 vi.小睡 疏忽 vt.使…起毛

narrow ['nærəu] adj.狭的 狭窄的 精密的 有限的 n.狭窄部份 海峡 隘路 vt.变窄 使...狭窄 vi.变窄

nasty ['nɑ:sti] adj.龌龊的 淫猥的 令人作呕的 污秽的 低劣的 难懂的 危害的

nation ['neiʃən] n.民族 国家

national ['næʃənəl] adj.民族的 国家的 n.国民

nationality [.næʃə'næliti] n.国籍 民族 族

native ['neitiv] adj.本土的 本国的 出生地的 天生的 n.本地人 土著人

natural ['nætʃərəl] adj.自然界的 天然的 正常的 天生的 n.有天赋的人

naturally ['nætʃərəli] adv.自然地 天然地

nature ['neitʃə] n.大自然 自然界 自然状态 本性 性质 类型

naughty ['nɔ:ti] adj.顽皮的 淘气的 粗俗的

naval ['neivəl] adj.海军的 军舰的 <古>船舶的

navigation [.nævi'geiʃən] n.航行 航海术 导航

navy ['neivi] n.海军 船队 深蓝色

near [niə] adv. 近 接近 几乎 亲近 adj. 近的 接近的 亲近的 勉强的 直接的 小气的 近似的 prep. 靠近

vt. 靠拢 接近 vi. 接近

nearby ['niəbai] adj.附近的 adv.在附近 prep.在...附近

nearly ['niəli] adv.差不多 几乎 密切地

neat [ni:t] adj.整洁的 熟练的 巧妙的 端正的 极好的 纯的 adv.整齐地

necessarily ['nesəserili] adv.必然 必定 必需地

necessary ['nesə.səri] adj.必要的 必需的 必然的 n.[pl.]必需品

necessity [ni'sesiti] n.必要性 必然性 需要 必需品

neck [nek] n.颈 脖子

necklace ['neklis] n.项链 项圈

need [ni:d] vt.需要 必需 v.aux.需要 n.需要 必须 缺乏

needle ['ni:dl] n.针 缝补 编织针 vt.用针缝 激怒 嘲弄 vi.缝纫

needless ['ni:dlis] adj.不需要的 无用的

negative ['negətiv] adj.否定的 消极的 负的 n.底片 负数 否定 vt.否定

neglect [ni'glekt] vt.忽视 忽略 疏忽 不顾 n.疏忽 忽略 怠慢

Negro ['ni:grəu] n.(含歧视意味)黑人 adj.黑人的

neighbour ['neibə] n.邻居(邻近的东西 邻近值) 邻国 邻人 vt.邻接(相邻 接近)

neighbourhood ['neibəhud] n.邻居关系 邻近

neither ['ni:ðə] adj.(两者)都不的 皆不 int.两者都不 conj.既非 既不

nephew ['nefju:] n.侄子 外甥

nerve [nə:v] n.神经(生物) 勇敢 胆量 精神 勇气 叶脉 vt.鼓起勇气

nervous ['nə:vəs] adj.神经(质)的 易激动的 紧张的 焦虑的

nest [nest] n.巢 窝 穴 vt.嵌入 vi.筑巢

net [net] n.网 网状物 通信网 净利 实价 adj.净余的 纯粹的 vt.用网捕 撒网 净赚

network ['netwə:k] n.网状物 网络 系统 广播网 vt.用网覆盖 联网 vi.加入网络

neutral ['nju:trəl] adj.中立的 中性的 n.中立者 空挡的 素净色

never ['nevə] adv.永不 决不 不

nevertheless [.nevəðə'les] conj.然而 不过 adv.仍然 不过

new [nju:] adj.新的 新近出现的 附加的 adv.新的

newly ['nju:li] adv.新近 重新地 最近

news [nju:z] n.新闻 消息

newspaper ['nju:z.peipə] n.报纸 报 报社 新闻纸 v.从事新闻工作

next ['nekst] adj.紧接的 贴近的 下一个的 adv.其次 n.下一个 prep.靠近

nice [nais] adj.美好的 令人愉快的 漂亮的 善意的

niece [ni:s] n.侄女 外甥女 教士的私生女

night [nait] n.夜 夜间

nine [nain] num.九 九个

nineteen ['nain'ti:n] num.十九 十九个

ninety ['nainti] num.九十 九十个

ninth [nainθ] num.第九 九分之一 adj.第九的 九分之一的

nitrogen ['naitrədʒən] n.氮

no [nəu] adv.不 并不 adj.没有 不是 绝非 n.不 拒绝 否决票

noble ['nəubl] adj.贵族的 高尚的 宏伟的 贵族的 高贵的 n.贵族

nobody ['nəubɔdi] pron.谁也不 无人 没有人 n.无足轻重的人 小人物

nod [nɔd] vt.点(头) 点头表示 打盹 摆动 v.点头 打盹 使...摆动

noise [nɔiz] n.喧闹声 响声 噪声 vt.谣传 vi.大声说话 发出噪音

noisy ['nɔizi] adj.嘈杂的 喧闹的 发出噪声的

none [nʌn] pron.没有人 没有任何东西 adv.毫不 一点也不

nonsense ['nɔnsəns] n.胡说 废话 荒唐

noon [nu:n] n.正午 中午 全盛期 顶点

nor [nɔ:] conj. & adv.也不 也没有 不 [计算机] "或非""非或" abbr.诺曼人(=Norman) abbr.北方(=North) abbr.挪威(= Norway) abbr.挪威人(=Norwegian)

normal ['nɔ:məl] adj.正常的 正规的 普通的 n.常态 标准

normally ['nɔ:məli] adv.通常 正常地

north [nɔ:θ] n.北 北方 adj.北 北方的 adv.向北方

northeast ['nɔ:θ'i:st] n.东北 adj.位于东北的 东北的 向东北的 来自东北的 adv.向东北 来自东北

northern ['nɔ:ðən] adj.北方的 北部的

northwest ['nɔ:θ'west] n.西北 西北方 adj.位于西北的 西北的 在西北的 来自西北的 adv.向西北

nose [nəuz] n.鼻子 突出部分 嗅觉 v.嗅 嗅到

not [nɔt] adv.不 没有 [计算机] "非"

note [nəut] n.笔记 便条 注释 备忘录 音符 音调 票据 纸币 vt.记录 注解 注意

notebook ['nəutbuk] n.笔记本 笔记本电脑 期票簿

nothing ['nʌθiŋ] n.没有东西 微不足道的人或事 adv.毫不 pron.什么也没有

notice ['nəutis] vt.注意 通知 留心 n.通知 注意 布告

noticeable ['nəutisəbl] adj.显而易见的 重要的

noun [naun] n.名词

novel ['nɔvəl] n.小说 adj.新的 新奇的

November [nəu'vembə] n.十一月

now [nau] adv. 现在 立刻 于是 adj. 现存的 目前的 n. 现在 此刻 conj. 既然

abbr.=National Organization of Women 全国妇女组织

nowadays ['nauədeiz] adv.现今 现在

nowhere ['nəuwɛə] adv.任何地方都不 n.不知道的地方 无处

nuclear ['nju:kliə] adj.原子核的 原子能的 核心的

nucleus ['nju:kliəs] n.核 核心 (原子)核 细胞核

nuisance ['nju:sns] n.讨厌的东西 讨厌的人 伤害

number ['nʌmbə] n.数 数字 号码 数量 vi.总计 编号 vt.编号 把...算作

numerous ['nju:mərəs] adj.为数众多的 许多

nurse [nə:s] n.保姆 护士 奶妈 vt.看护 哺乳 vi.看护 照顾 栽培 当看护

nursery ['nə:səri] n.托儿所 苗圃

nut [nʌt] n.坚果 干果 螺母 螺帽 难对付的人(事) vi.采坚果

nylon ['nailɔn] n.尼龙 耐纶 (复数nylons)：女长袜

## O

oak [əuk] n.栎属植物 栎木 橡树 橡木 adj.橡木制的

oar [ɔ:] n.浆 橹 划手 v.划行

obey [ə'bei] vt.顺从 vi.服从 听从

object ['ɔbdʒikt] n.物 物体 目的 目标 宾语 v.反对 提出...作为反对的理由 抱反感

objection [əb'dʒekʃən] n.反对 异议 不喜欢

objective [əb'dʒektiv] adj.客观的 目标的 无偏见的 n.目标 目的 物镜 =objective case 宾语

oblige [ə'blaidʒ] vt.迫使 强制 施恩惠于 vi.帮忙

observation [.ɔbzə'veiʃən] n.注意 观察 观察力 评论 adj.被设计用来观察的

observe [əb'zə:v] vt.遵守 看到 注意到 说 观察

observer[əb'zə:və] n.观察员 观测者

obstacle ['ɔbstəkl] n.障碍 障碍物 妨害 绊脚石

obtain [əb'tein] vt.获得 得到 买到 vi.通用 流行 存在

obvious ['ɔbviəs] adj.明显的 显而易见的

obviously ['ɔbviəsli] adv.明显地 显然地

occasion [ə'keiʒən] n.场合 时刻 时机 机会 理由 vt.致使 惹起 引起

occasional [ə'keiʒənəl] adj.偶然的 临时的 不时的

occasionally [ə'keiʒənəli] adv.偶然 非经常地

occupation [.ɔkju'peiʃən] n.占领 占据 侵占 职业 居住

occupy ['ɔkjupai] vt.占领 占 占有 占据 使忙碌 居住 担任

occur [ə'kə:] vi.发生 出现 存在 想到

occurrence [ə'kʌrəns] n.发生 出现 事件

ocean ['əuʃən] n.海洋 洋 大量

Oceania [.əuʃi'einiə] n.大洋洲

o'clock [ə'klɔk] ad. =of the clock ...点钟

October [ɔk'təubə] n.十月

odd [ɔd] adj.奇数的 单只的 古怪的 剩余的

odour ['əudə] n.气味 香气 味道 =odor(美)

of [ɔv] prep.出生于 住在 关于 ...的 由于

off [ɔ:f] adv. (离)开 (停)止 出发 距离隔断 分割 不继续运行 下台 adj. 远的 休假的 空闲的 落后懈怠的

prep. 离开 脱落 下班 vi. 离开 走开

offend [ə'fend] vt.冒犯 犯罪 vi.犯过错 令人不适 违反

offer ['ɔ:fə] vt.提供 提出 出价 贡献 奉献 vi.提议 出现 求婚 n.提供 出价

office ['ɔ:fis] n.办公室 职务 事务 处 局 社

officer ['ɔ:fisə] n.官员 干事 军官 vt.提供官员 指挥

official [ə'fiʃəl] adj.官员的 官方的 正式的 有权威的 n.官员

often ['ɔ:fən] adv.经常 常常

oh [əu] int.嗬 哦 唉呀

oil [ɔil] n.油 石油 油画颜料 奉承话 vt.加油于 涂油 溶化

okay [əʊ'kei] (缩作OK)adv. & adj.<表示同意或允许>对 好 同意 vt.同意

old [əuld] a.老的 某年龄的 以前的 陈旧的 熟悉的 成熟的 ...岁的 n.古时 某年龄

omit [əu'mit] vt.省略 省去 遗漏 疏忽

on [ɔn] prep. 在...上 在...旁 由...支撑着 在(某一天) 使用 根据 [计算机] 开 adv. 穿着 向前(移动) 表示持续性 adj. 表示已连接、处于工作状态或使用中 表示发生 值班 abbr. =Ontario安大略湖[北美洲]

once [wʌns] adv.一次 曾经 n.一次 conj.一旦

one [wʌn] num.一 pron.一个人 某人 任何人 n.一 第一个 一美元纸币 adj.单一的 某一个 同一种类的 同意的 唯一的

oneself [wʌn'self] pron.自己 亲自

onion ['ʌnjən] n.洋葱 洋葱头

only ['əunli] adv.只 仅仅 专门地 唯一地 刚才 adj.最好的 唯一的 出众的 conj.除非 但是 然而 毕竟

onto ['ɔntu:] prep.到...上

open ['əupən] adj.开的开着的 开放的 公开的 vt.开 打开 公开 开放 开始 vi.开 通行 大叫 n.公开 户外 空旷

opening ['əupəniŋ] adj.开始的 n.开始 开口 揭幕 空缺的职务 动词open的现在分词

opera ['ɔpərə] n.歌剧 挪威Opera软件公司的浏览器软件

operate ['ɔpəreit] vi.操作 施行手术 运转 vt.起作用 动手术

operation [.ɔpə'reiʃən] n.操作 手术 运算 行动 活动

operational [.ɔpə'reiʃənəl] adj.操作上的 可使用的 运作的

operator ['ɔpə.reitə] n.操作人员 接线员 管理者 技工 报务员

opinion [ə'pinjən] n.意见 看法 主张 判断

opponent [ə'pəunənt] n.对手 敌手 对抗者 反对者 adj.敌对的 反对的

opportunity [.ɔpə'tju:niti] n.机会 时机 良机

oppose [ə'pəuz] vt.反对 反抗 以...对抗 抗争

opposite ['ɔpəzit] adj.对面的 相对的 相反的 n.对立物 对立者 对手 adv.在对面地 处于相反位置地 prep.在...对面

oppress [ə'pres] vt.压迫 压制(某人) 压抑 使(心情等)沉重 使烦恼

optical ['ɔptikəl] adj.眼的 眼睛的 视觉的 光学的

optimistic [.ɔpti'mistik] adj.乐观的 乐观主义的

option ['ɔpʃən] n.选择 选择权 可选物 vt.选择

optional ['ɔpʃənl] adj.可以任意选择的

or [ɔ:,ə] conj.或 或者 即 还是 否则

oral ['ɔ:rəl] adj.口头的 口的 n.口试

orange ['ɔ:rindʒ] n.橙(树) 柑(树) 橙黄色 adj.橙色的

orbit ['ɔ:bit] n.运行轨道 势力范围 眼眶 轨迹 vt.环绕 绕轨道而行 进入轨道 vi.盘旋

orchestra ['ɔ:kistrə] n.管弦乐队

order ['ɔ:də] n.次序 整齐 勋章 阶级 决议 订单 命令 vt.命令 调整 定购

orderly ['ɔ:dəli] adj.整洁的 有秩序的 一丝不苟的 和平的 adv.有秩序地 有条理地 依次地 n.勤务兵 (医院的)勤务工

ordinary ['ɔ:dnri] adj.平常的 平凡的 普通的

ore [ɔ:] n.矿 矿石 矿砂

organ ['ɔ:gən] n.器官 机构 管风琴 机关报刊

organic [ɔ:'gænik] adj.有机体的 器官的 根本的 接近自然的 n.有机物质

organism ['ɔ:gənizəm] n.生物体 有机体

organization [.ɔ:gənai'zeiʃən] n.组织 团体 机构 adj.有组织的

organize ['ɔ:gənaiz] vt.组织 编组 vi.组织起来

oriental [.ɔ(:)ri'entl] adj.东方(人)的 东方国家的 贵重的

origin ['ɔridʒin] n.起源 由来 出身

original [ə'ridʒənl] adj.最初的 新颖的 原始的 有独创性的 原版的 n.原件 原稿 有独创性的人 怪人

ornament ['ɔ:nəmənt] n.装饰物 装饰 vt.装饰

orphan ['ɔ:fən] n.孤儿 adj.无双亲的 孤儿的 vt.成孤儿

other ['ʌðə] adj.另外的 别的 其余的 pron.另一个人(或事) n.其他人(或事) adv.否则 不同地

otherwise ['ʌðəwaiz] adv.另外 要不然 否则 不同地 在不同方面 pron.其他 adj.不同的

ought [ɔ:t] v.aux.应当 应该 conj.应该 大概

ounce [auns] n.盎司 英两

our ['auə] pron.我们的 adj.我们的

ours ['auəz] pron.我们的

ourselves [.auə'selvz] pron.我们自己

out [aut] adv. 出 在外 向外 出现 现出来 离岸 外出 出去 出局 用尽 完成 结束 adj. 外面的 向外的 落伍的 不做考虑的 vt. 释放出 结露 vi. 揭发 prep. 向外 通过 n. 外面 在野者 下台者 出局的一击

outcome ['autkʌm] n.结果 后果 成果

outdoor ['autdɔ:] adj.户外的 室外的 野外的 露天的

outdoors ['aut'dɔ:z] adv.在户外 在野外 n.露天 野外 名词outdoor的复数形式

outer ['autə] adj.外部的 外面的

outlet ['autlet] n.出口 出路 通风口 批发商店 排遣

outline ['autlain] n.轮廓 略图 大纲 概要 素描 vt.描画轮廓 描述要点

outlook ['autluk] n.观点 看法 展望 前景 景色

output ['autput] n.产量 输出量 输出 输出功率 输出端 vt.输出(信息等)

outset ['autset] n.开始 开端

outside ['aut'said] n.外部 外表 adj.外部的 adv.在外面 prep.在...外 pron.在...外边

outskirt ['aut.skə:t] n.外边 郊区

outstanding [aut'stændiŋ] adj.突出的 杰出的 显著的 未支付的

outward ['autwəd] adj.外面的 向外的 表面的 外部的 adv.向外 在外 表面

outwards ['autwədz] adv.向外 在外 外表上 往海外 =outward

oven ['ʌvən] n.炉 灶 烘箱 烤炉

over['əuvə] prep.在...上方 超过 表示能控制 在...期间 adj.结束的 超过的 多余的 adv.结束 落下 穿过 完全覆盖 vt.越过

overall [əuvə'rɔ:l] n.工装裤 防护服 罩衫 adj.全面的 全部的 全体的 一切在内的 adv.总的来说 全部地

overcoat ['əuvəkəut] n.外衣 大衣

overcome [.əuvə'kʌm] vt.战胜 克服 (感情等)压倒 使受不了 vi.获胜 赢

overhead ['əuvə'hed] adj.在头顶上的 架空的 高架的 n.经常开支 普通用费 adv. 在头顶上 在空中 在高处 [计算机] 总开销

overlook [.əuvə'luk] vt.眺望 俯瞰 远眺 看漏 放任 没注意到 忽视 n.高出

overnight ['əuvə'nait] adv.一夜 昨晚一晚上 突然 adj.通宵的 晚上的 前夜的 n.前晚

overseas ['əuvə'si:z] adv.在海外 adj.在海外的

overtake [.əuvə'teik] vt.追上 赶上 压倒 突然来袭

overtime ['əuvətaim] adj.超时的 加班的 adv.在规定时间之外(超出时间地) n.加班时间(加班加点费)

owe [əu] vt.欠(债等) 应把...归功于 vi.欠钱

owl [aul] n.猫头鹰 枭

own [əun] adj.自己的 嫡亲的 vt.有 拥有 承认 vi.承认 pron.自己的

owner ['əunə] n.物主 所有人

ownership ['əunəʃip] n.所有(权) 所有制

ox [ɔks] n.牛 公牛 阉牛

## P

phase [feiz] n.阶段 时期 局面 方面 相位 vt.逐步执行 实行 按计划进行

phenomenon [fə'nɑ:minən] n.现象 非凡的人或事物

philosopher [fi'lɔsəfə] n.哲学家 思想家 达观者

philosophy [fi'lɔsəfi] n.哲学 哲理 人生观

phone [fəun] n.电话 电话机 耳机 [语] 音素 v.打电话 suf.表示“声音 说...语言的人”

photograph ['fəutəgrɑ:f] n.照片 相片 vi.照相 摄影 vt.给...照相

photographic [.fəutə'græfik] adj.摄影的 摄影用的 逼真的

phrase [freiz] n.短语 习惯用语 个人风格 乐句 vt.措词表达 将(乐曲)分成乐句(来演奏)

physical ['fizikəl] adj. 身体的 物质的 物理的 n.体检

physician [fi'ziʃən] n.医生 内科医生

physicist ['fizisist] n.物理学家 <古>自然科学家

physics ['fiziks] n.物理学

piano [pi'ænəu,pi'ɑ:nəu] n.钢琴 adj. & adv.轻柔

pick [pik] n.镐 鹤嘴锄 采摘 精选 精品 收获量 vt. 拾 摘 挑选 挑惕 剔除 戳 撬 vi.慎选 凿 吹毛求疵 采收 采摘

picnic ['piknik] n.郊游 野餐 愉快的时间 猪脊肉 vi.野餐

picture ['piktʃə] n.画 图片 照片 景色 vt.画 描写 想像

pie [pai] n.(西点)馅饼 派 杂乱 喜鹊 爱说话的人 饼图 vt.弄乱

piece [pi:s] n.碎片 块 片 篇 vt.拼合 修补 结合

pierce [piəs] vt.剌穿 穿透 洞悉 vi.穿入 n. 皮尔斯

pig [pig] n.猪 小猪 野猪

pigeon ['pidʒin] n.鸽子

pile [pail] n.堆 桩 大量 核反应堆 vt.堆叠 堆积 累积 装载 vi.形成堆 拥挤进入

pill [pil] n.药丸 丸剂 无法逃避的讨厌事 屈辱事 vt.服用药丸 把...制成药丸 被抢劫 vi.起球粒 成丸状 阶段性完成

pillar ['pilə] n.柱 柱子 栋梁 支柱 核心(人物) vt.用柱支持

pillow ['piləu] n.枕头 vt.枕 搁 垫

pilot ['pailət] n.领航员 飞行员 引航员 vt.领航 驾驶 向导 adj.引导的 示范的

pin [pin] n.针 饰针 大头针 拴 (有别针的)徽章 小东西 (复数)腿 别住 vt.将...用针别住 钉住 压住 adj.别针的

pinch [pintʃ] vt.捏 拧 掐掉 一撮 少量 困苦 偷窃 vt.掐 使...困苦 偷窃

pine [pain] n.松树 松木 vi.消瘦 憔悴 渴望

pink [piŋk] n.粉红色 石竹花 典范 adj.粉红色的 激进的 vt.刺

pint [paint] n.品脱

pioneer [.paiə'niə] n.拓荒者 先驱者 先锋 vt.提倡 开辟 vi.做先驱

pipe [paip] n.管子 导管 烟斗 笛 vt.以管输送 装管道 吹奏 尖声说 拷边 vi.吹奏管乐 尖叫

pipeline ['paip.lain] n.管道 管线

pistol ['pistl] n.手枪 vt.用手枪射击

pit [pit] n.坑 地坑 煤矿 深坑 核 矿井 陷阱 英国剧场正厅后排 凹陷疤痕 vt.使...有伤痕 去...的核 与...较量

pitch [pitʃ] n.沥青 树脂 松脂 n.程度 投掷 球场 音高 vt.投 掷 颠簸 扎牢 搭帐篷 定调 vi.用沥青覆盖 投掷

pity ['piti] n.怜悯 遗憾 遗憾的事 vt.同情 怜悯

place [pleis] n.地方 地点 住所 职位 获奖的名次 vt.放置 任命 安排 vi.获得名次

plain [plein] n.平原 草原 adj.清楚的 坦白的 简单的 平坦的 平常的 纯的

plan [plæn] n.计划 策略 方法 vt.计划 设计 打算 vi.打算

plane [plein] n.平面 飞机 刨子 (动植物的)属 adj.平的 与飞机有关的 vt.磨平 刨 vi.开飞机

planet ['plænit] n.行星

plant [plɑ:nt] n.植物 工厂 设施 设备 vt.栽种 种植 建立 设立

plantation [plæn'teiʃən] n.种植园 大农场 殖民地 栽植

plaster ['plɑ:stə] n.灰泥 硬膏 熟石膏 膏药 v.涂以灰泥 敷以膏药 使...平

plastic ['plæstik] adj.可塑料的 可塑的 体态好的 n.塑料 (外科)整形的 信用卡

plate [pleit] n.板 片 盘 碟 金属板 板块 vt.镀金 电镀 为...制印版 给...装钢板 用金属板固定

platform ['plætfɔ:m] n.平台 站台 讲台 月台 坛 计划

play [plei] vi.玩 游戏 演奏 进行比赛 播放 捉弄 假装 n.游戏 游玩 剧本 比赛 比赛中的动作 影响

player ['pleiə] n.游戏的人 比赛者 演奏者 演员 播放器

playground ['pleigraund] n.操场 运动场

pleasant ['pleznt] adj.令人愉快的 舒适的 vi.满意 喜欢 vt.使高兴

please [pli:z] vt.取悦 使高兴 合心意 请 vi.取悦 满意 adv.请

pleasure ['pleʒə] n.愉快 快乐 乐事 vt.使高兴 vi.享受快乐 高兴

plentiful ['plentifəl] adj.丰富的 富裕的 多的

plenty ['plenti] n.丰富 充足 大量 繁荣 adj.充足的 adv.非常

plot [plɔt] n.小块土地 阴谋 情节 图 vt.密谋 绘图 计划 标示位置

plough [plau] n.犁 耕地 vt. & vi.犁 耕 用犁耕田 耕犁

pluck [plʌk] vt.采 摘 拉下 猛拉 拔 n.拉 猛拉 勇气 动物内脏

plug [plʌg] n.塞子 插头 消防拴 电插头 vt.塞 射杀 宣传 vi.射杀 持续的工作

plunge [plʌndʒ] vt.使投入 使陷入 vi.投入 骤降 使...陷入 全神贯注于 n.突然跌落 跳水

plural ['pluərəl] adj.复数的 n.复数

plus [plʌs] prep.加 加上 另外 adj.正的 略高于 n.加号 正号 优势 conj.并且

P.M. [pi: 'em] n.下午 午后 abbr.下午(=post meridiem [mə'ridi:əm])

pocket ['pɔkit] n.衣袋 口袋 钱 袋 袖珍的 vt.装...在口袋里 隐藏 私吞 adj.小型的 钱的

poem ['pəuim] n.诗 韵文 诗体文

poet['pəuit] n.诗人

poetry ['pəuitri] n.诗 诗歌 诗作

point [pɔint] n.点 要点 细目 分 尖端 细节 目标

poison ['pɔizn] n.毒 毒药 败坏道德之事 毒害 vt.毒害 摧毁 中毒 adj.有毒的

poisonous ['pɔizənəs] adj.有毒的 有害的 恶意的

pole [pəul] n.杆 柱 极(点) 磁极 电极 vt.跳 支撑 vi.用篙撑船

police [pə'li:s] n.警察 警察当局 警方 vt.管辖 维持治安

policeman[pə'li:smən] n.警察

policy ['pɔlisi] n.政策 方针 保险单

polish ['pɔliʃ] vt.磨光 擦亮 使优美 推敲 完善 vi.变光亮 n.光泽 上光剂 优雅 精良

Polish ['pɔliʃ] adj.波兰的 波兰人的 波兰语的 n.波兰语

polite [pə'lait] adj.有礼貌的 有教养的 客气的

political [pə'litikəl] adj.政治的 政治上的 政党的 派系斗争的 有政治头脑的

politician [.pɔli'tiʃən] n.政治家 政客

politics ['pɔlitiks] n.政治 政治学 政纲 政策 政治活动 政见

pollute [pə'lu:t] vt.弄脏 污染 沾污

pollution [pə'lu:ʃən] n.污染 污染物

pond [pɔnd] n.池塘 vt.堵河成湖 vi.形成池塘

pool [pu:l] n.水塘 游泳池 水池 撞球 共用物 vi.形成池塘 vt.集中 积累

poor [puə] adj.贫穷的 贫乏的 可怜的 缺乏的 拙劣的

pop [pɔp] n. 流行音乐 流行歌曲 流行文化艺术 取出 砰的一声 爆破声 (尤用作称呼)爸 adj. 流行的 热门的 通俗的 vt. 取出 发出爆裂声 突然伸出 击打 开火 vi. 发出砰的一声 (因激动、惊奇)张大 adv. 突然间地 暴发地 POP abbr.=point of purchase POP abbr.=population 人口

popular ['pɔpjulə] adj.民众的 流行的 有销路的

population [.pɔpju'leiʃən] n.人口 全体居民

porch [pɔ:tʃ] n.门廊 入口处

pork [pɔ:k] n.猪肉 联邦或州政府用以笼络人心的工程或款项

porridge ['pɔridʒ] n.粥 麦片粥

port [pɔ:t] n.港 港口 舱门 码头 机场 (船 飞机的)左舷 [计算机] 端口 abbr. =portable portrait [Port] abbr. =Portugal

portable ['pɔ:təbl] adj.轻便的 手提的 n.便携的东西

porter ['pɔ:tə] n.搬运工人 门房 (火车卧铺车厢或豪华车厢的)乘务员 (医院 商店中的)清洁工 (旧时)黑啤酒 波特(人名)

portion ['pɔ:ʃən] n.一部分 一分 份 命运 分担的责任

portrait ['pɔ:trit] n.消像 画像 adj.(文件页面)竖的

Portugal ['pɔ:tjugəl] n.葡萄牙

Portuguese [.pɔ:tju'gi:z] n.葡萄牙人 葡萄牙语 adj.葡萄牙的 葡萄牙人的 葡萄牙语的

position [pə'ziʃən] n.位置 职位 姿势 状态 立场 vt.安置 决定...的位置

positive ['pɔzitiv] adj.确定的 积极的 肯定的 绝对的 正面的 正数的 阳性的

possess [pə'zes] vt.占用 拥有(财产) 持有 支配

possession [pə'zeʃən] n.有 财产 拥有 所有 占有物

possibility [.pɔsə'biliti] n.可能 可能的事 可能性 潜在的价值

possible ['pɔsəbl] adj.可能的 可能存在的

possibly ['pɔsəbli] adv.可能地 也许

post [pəust] n.柱 桩 杆 标竿 邮件 邮政 邮寄 职位 岗位 哨所 vt.贴出 邮递 发布 布置

postage ['pəustidʒ] n.邮费 邮资 邮资已付邮戳

postman ['pəustmən] n.邮递员=mailman

postpone [pəust'pəun] vt.延迟 推迟 延缓

pot [pɔt] n.锅 壶 罐 盆 一大笔(钱) vt.把...装罐 (台球)入袋

potato [pə'teitəu] n.马铃薯 土豆

potential [pə'tenʃəl] adj.潜在的 可能的 n.潜力 潜能 电位 电势

pound [paund] n.磅 英磅 宠物收留所 拘留所 vt.连续猛击 捣碎 舂烂 敲打 vi.跳动 大声播放

pour [pɔ:] vt.灌 倒 流出 倾泻 骤雨 vi.倾泻 蜂涌而来 倒

poverty ['pɔvəti] n.贫穷 贫困 贫乏

powder ['paudə] n.粉末 药粉 火药 细雪 vt.洒粉于 使变成粉 重击 vi.变成粉 涂粉

power ['pauə n.能力 力 权 权力 幂 力量 电力 vt.供电 激励 全速前进 adj.与力量有关的 与权力有关的 电力控制的

powerful ['pauəfəl] adj.强有力的 有权威的

practical ['præktikəl] adj.实践的 实用的 明智的 n.实践考试

practically ['præktikəli] adv.实际上 几乎 简直 adj.实际的 几乎

practice ['præktis] n.实践 练习 业务 实行 习惯 vt. & vi.练习 实习 开业

practise ['præktis] vt.练习 实习 训练 实践(使...练习 养成...的习惯)

praise [preiz] n.赞扬 赞美 称赞 崇拜 vi.赞扬 vt.称赞 归荣誉于 赞美

pray [prei] vt. &vi.请求 恳求 祈祷

prayer [prɛə] n.祈祷 祈求 祷告 祷文 v.祷告 祷文

precaution [pri'kɔ:ʃən] n.预防(措施) 警惕

preceding [pri'si:diŋ] adj.在前的 在先的

precious ['preʃəs] adj.珍贵的 宝贵的 矫揉造作的 adv.极其地

precise [pri'sais] adj.精确的 准确的 认真的

precision [pri'siʒən] n.精确 精密 精密度 adj.以精准的执行著称的 经得起极精细测量的

predict [pri'dikt] v.预言 预告 预测 预报

preface ['prefis] n.序言 前言 引语 v.作序 写前言 以...开始

prefer [pri'fə:] vt.宁可 宁愿 较喜欢 vt.【法】提出(控告)

preferable ['prefərəbl] adj.更可取的 更好的 更合意的

preference ['prefərəns] n.偏爱 喜爱物 优先 优先权

prejudice ['predʒudis] n.偏见 成见 伤害 vt.使...存偏见 伤害

preliminary [pri'liminəri] adj.预备的 初步的 开始的 n.初步行动 准备 初步措施

premier ['premjə] n.总理 首相 adj.第一的 首位的 最初的

preparation [.prepə'reiʃən] n.准备 预备 制备

prepare [pri'pɛə] vt. & vi.准备 预备 整合

preposition [.prepə'ziʃən] n.前置词 介词

prescribe [pris'kraib] vt.命令 规定 开处(方) vi.规定 开药方

presence ['prezns] n.出席 到场 在 在场者 所在地方 举止

present ['preznt] adj.现在的 当面的 出席的 n.礼物 赠送物 现在 目前

present [pri'zent] vt.赠送 介绍 提出 呈现

presently ['prezntli] adv.一会儿 目前 不久 现在 <古> 当即 立刻

preserve [pri'zə:v] vt.保护 保存 保持 腌渍 vi.保存 n.蜜饯 专供私人行猎或捕鱼的保留区

president ['prezidənt] n.总统 总裁 总经理 校长 会长 社长

press [pres] vt.压 按 揿 催促 逼迫 强迫服役 征用 n.新闻界 压力 强迫服役

pressure ['preʃə] n.压力 压强 压迫 压 按

pretend [pri'tend] vt.假托 借口 vi.假装 伪称 adj.假装的 赝品的 模拟的

pretty ['priti] adj.漂亮的 标致的 机灵的 恰当的 美好的 adv.相当地 n.(复数)妇女内衣 漂亮的人或物 vt.使漂亮

prevail [pri'veil] vi.胜 优胜 获胜 流行 盛行 主导

prevent [pri'vent] vt.预防 防止 阻止

previous ['pri:vjəs] adj.先的 前的 在...之前 以前的 adv.在前

previously ['pri:vjəsli] adv.先前 预先 仓促地

price [prais] n.价格 价钱 代价 vt.定格 标价 比较价格

pride [praid] n.骄傲 自豪 自尊 一群狮子 vt.自夸 以...自豪

priest [pri:st] n.教士 牧师 神父

primarily [prai'mərili] adv.首先 主要地

primary ['praiməri] adj.最初的 基本的 主要的 根本的 初等教育的 n.最主要者 初选 原色 初级线圈

prime [praim] adj. 首要的 最初的 最好的 典型的 n. 春 青春 壮年 全盛时期 n. [数]质数 vt. 事先指点 在(金属、木材等上)打底漆 vi. 变得首要

primitive ['primitiv] adj.原始的 粗糙的 n.原始人 文艺复兴前的艺术家

prince [prins] n.王子 亲王 佼佼者

princess ['prin'ses] n.公主 王妃 贵妇 adj.(礼服)合身而好看的 有贵族气息的

principal ['prinsəpəl] adj.主要的 首要的 n.负责人 首要 校长

principle ['prinsəpl] n.原则 原理 主义 信念

print [print] vt.印刷 出版 打印 n.印刷 正片 版 印刷物 印迹 adj.印刷的

prior ['praiə] adj.在先的 优先的 更重要的 较早的 adv.居先 n.小修道院院长 大修道院副院长

prison ['prizn] n.监狱 监禁

prisoner ['prizənə] n.囚犯

private ['praivit] adj.私人的 私下的 隐蔽的 n.士兵 列兵

privilege ['privilidʒ] n.特权 优惠 特别恩典 基本人权 荣幸 vt.给特权 免除

prize [praiz] n.奖赏 奖金 奖品 珍品 战利品 vt.珍视 重视 adj.奖励的 杰出的

probability [.prɔbə'biliti] n.可能性 概率 机率

probable ['prɔbəbl] adj.或有的 大概的 很可能的 n.很可能的事

probably ['prɔbəbli] adv.或许 大概

problem ['prɔbləm] n.问题 习题 adj.成问题的 难对付的

procedure [prə'si:dʒə] n.程序 手续 过程 步骤 常规的做法

proceed [prə'si:d] vi.进行 开始 着手 继续进行

process ['prɑ:ses] n.过程 工序 vt.加工 处理 起诉 vi.列队前进 adj.经过加工的

procession [prə'seʃən] n.队伍 行列 v.列队行进

proclaim [prə'kleim] vt.宣告 宣布 表明 公布 赞扬 显示出

produce [prə'dju:s] vt.生产 产生 展现 提出 引起 分娩 制片 vi.生产 制造 n.产品 农作物

product ['prɔdəkt] n.产品 产物 (乘)积

production [prə'dʌkʃən] n.生产 产品 总产量 作品 成果

productive [prə'dʌktiv] adj.生产的 丰饶的 有生产价值的 多产的

profession [prə'feʃən] n.职业 专业 声明 信仰

professional [prə'feʃənəl] adj.职业的 专业的 n.专业人员

professor [prə'fesə] n.教授 宣称者

profit ['prɔfit] n.益处 利润 盈利比率 vi.得益 利用 获利 vt.有益于

program ['prəugræm] n.节目单 大纲 程序 计划 vt.规划 拟定计划 制作节目

progress [prəu'gres] n.前进 进展 进步 vi.促进 进步 进行

progressive [prə'gresiv] adj.进步的 向前进的 渐进的 n.进步人士

prohibit [prə'hibit] vt.禁止 阻止 妨碍

project [prə'dʒekt] n.方案 工程 项目 计划 vi.伸出 投射 放映 凸出 vt.计划 设计 表达

prominent ['prɔminənt] adj.实起的 突出的 杰出的 显著的

promise['prɔmis] n.诺言 指望 约定 (有)希望 预示 vt.允诺 约定 预示 有可能

promising ['prɔmisiŋ] adj.有希望的 有前途的

promote [prə'məut] vt.促进 发扬 提升 升迁 发起 促销

prompt [prɔmpt] adj. 及时的 迅速的 敏捷的 立刻的 vt. 敦促 激起 促进 推动 提示 vi. 提示 n. 提示 提示的内容 [商] 付款期限

pronoun ['prəunaun] n.代名词

pronounce [prə'nauns] vt.发...的音 宣布 断言 发表意见

pronunciation [prə.nʌnsi'eiʃən] n.发音 发音法

proof [pru:f] n.证据 证明 校样 adj.防...的 耐...的 能防护 vt.检验 给...做防护措施

proper ['prɔpə] adj.适合的 正当的 正确的 合乎体统的

properly ['prɔpəli] adv.适当地 彻底地 当然地

property ['prɔpəti] n.财产 资产 性质 道具

proportion [prə'pɔ:ʃən] n.比 比率 部分 均衡 vt.使成比例 使均衡 分摊

proportional [prə'pɔ:ʃənəl] adj.(成)比例的 相称的 n.[数] 比例项

proposal [prə'pəuzəl] n.提议 建议 求婚

propose [prə'pəuz] vt.提议计划 打算 向...提议 vi.求婚

prospect ['prɔspekt] n.展望 前景 前程 景色 v.探勘 寻找

prosperity [prɔs'periti] n.繁荣 昌盛 兴旺

prosperous ['prɔspərəs] adj.繁荣的 昌盛的 兴旺的

protect [prə'tekt] vt.保护 保卫 警戒 投保 vi.提供保护

protection [prə'tekʃən] n.保护 警戒 防卫

protective [prə'tektiv] adj.保护的 防护的

protein ['prəuti:n] n.蛋白质 朊

protest ['prəutest,prə'test] n.抗议 反对 抗议书 vi.反对 抗议 断言 vt.坚持地表示 对...提出异议

proud [praud] adj.骄傲的 自豪的 壮观的

prove [pru:v] vt.证明 查验 显示 vi.结果是

provide [prə'vaid] vt.提供 供给 装备 规定 准备 vi.供给 提供

provided [prə'vaidid] conj.假如 若是 以...为条件 adj.预备好的 由...供给的

province ['prɔvins] n.省 范围 领域 部门

provision [prə'viʒən] n.供应(品) 预备 存粮 规定 条款 (pl.)食物 vt.向...供应食物

psychological [.saikə'lɔdʒikəl] adj.心理的 心理学的

public ['pʌblik] adj.公众的 公开的 众所周知的 n.公共场合 公众 同好者

publication [.pʌbli'keiʃən] n.公布 发表 出版 发行 出版物

publish ['pʌbliʃ] vt.公布 发表 出版 发行 vi.出版 发行

pudding ['pudiŋ] n.布丁 血肠 (英)甜食

puff [pʌf] n.(一)喷 (一)吹 一阵喷烟 粉扑 泡夫 vt.喷出 喘息 夸张 膨胀 vi.喷出 膨胀 骄傲 adj.吹嘘的

pull [pul] vt.拉 拖 拔 拉 拉力 vi.拔 拉 拖 扳动 n.拉 拖 拉绳 影响

pulse [pʌls] n.脉搏 脉冲 脉动 v.跳动 脉跳

pump [pʌmp] n. 泵 抽水机 打气筒 心脏 抽水 轻软舞鞋 vt. 用抽机抽 抽水 打气 用泵输送 盘问 涌出

vi. 用泵抽水 快速上下(或内外)运动

punch [pʌntʃ] vt.冲出 以拳重击 开洞 剪票 按(键) n.冲压机 打洞器 钻孔机 殴打 酒、水、糖等制成的鸡尾酒 拳打

punctual ['pʌŋktjuəl] adj.严守时刻的 准时的 正点的

punish ['pʌniʃ] vt.罚 惩罚 处罚

punishment ['pʌniʃmənt] n.罚 惩罚 处罚

pupil ['pju:pl] n.学生 小学生 学徒 瞳孔

puppet ['pʌpit] n.木偶 玩偶 傀儡

purchase ['pə:tʃəs] n.买 购买 购买的物品 支点 紧握 抓紧 vt.买 购买 赢得 (用设备)举起 移动

pure [pjuə] adj.纯的 纯粹的 纯洁的 纯理论的 抽象的 完全的

purely ['pjuəli] adv.纯粹地 完全地 清洁地 贞淑地

purify ['pjuərifai] vt.使纯净 使洁净 (使)精炼 涤罪

purity ['pjuəriti] n.纯净 纯洁 纯度 纯正 纯粹

purple ['pə:pl] n.紫色 皇室 皇权 adj.紫的 帝王的 辞藻华丽的 v.变紫

purpose ['pə:pəs] n.目的 意图 效果 意志 决心 议题 vt.打算 决意

purse [pə:s] n.钱包 小钱袋 手袋 资金 金钱 vt.皱起 放进钱包

pursue [pə'sju:] vt.追赶 追踪 追捕 追求 进行 继续从事

pursuit [pə'sju:t] n.追赶 追求 事务 工作

push [puʃ] vt.推 逼迫 奋力 决心 vi.推 挤 逼迫

put [put] vt.放 摆 安置 使处于 vi.提交 记下

puzzle ['pʌzl] n.难题 谜惑 vi.使迷惑 使困惑 解决 vi.迷惑 苦思

## Q

qualify ['kwɔlifai] vt.使具有资格 取得资格 有资格 限定 描述 vi.取得资格 有资格

quality ['kwɔliti] n.质量 品质 特质 特性 才能 adj.高品质的

quantity ['kwɔntiti] n.量 数量 分量 大量

quarrel ['kwɔrəl] n.争吵 吵架 口角 争论 怨言 vi.吵架 争论 挑剔

quart [kwɔ:t] n.夸脱(=2品脱)

quarter ['kwɔ:tə] n.四分之一 一刻钟 季度 25美分 vt.四等分 vi.住宿 驻扎 adj.四分之一的

quarterly ['kwɔ:təli] adj.季度的 adv.季度地 每季一次 n.季刊

queen ['kwi:n] n.王后 女王 最重要的女子或事物 女冠军 女王蜂 vi.立...为女王 当女王 vt.成为女王或王后 以女王的身份统治 vt.使卒子达底线变为王后取得纵 横 斜自由行走的资格

queer [kwiə] adj.奇怪的 古怪的 不舒服的 可疑的 n.同性恋者

question ['kwestʃən] n.发问 问题 疑问 询问 争论点 vt.询问 怀疑 vi.问问题

queue [kju:] n.行列 长队 vi.排队等候

quick [kwik] adj.快的 敏捷的 迅速的 机敏的 adv.快 迅速地 n.活肉 本质

quicken ['kwikən] vt. 使加快 使...有生气 鼓舞 使(曲线)更弯 使(斜坡)更陡 vi. 加快 变快 苏醒 (孕妇) 进入胎动期 变明亮 变活跃

quickly ['kwikli] adv.快 迅速

quiet ['kwaiət] adj.寂静的 安静的 静止的 宁静的 adv.安静地 v.使...平静 使...安心 n.安静 闲适 平静

quilt [kwilt] n.被(子) 被状物 vt.缝被 缝制 用垫料填塞 vi.制作被子 做缝制工作

quit [kwit] vt.离开 辞职 退出 停止

quite [kwait] adv.完全 很 相当 的确

quiz [kwiz] n.小型考试 (随堂)测验 恶作剧 古怪的人 vt.恶作剧 提问

quotation [kwəu'teiʃən] n.引用 引文 引语 语录 引用 报价单 行情

quote [kwəut] vt.引用 引证 报价 举证 n.引用

## R

rabbit ['ræbit] n.兔子 野兔 vi.猎兔 vt.让...见鬼去吧 该死

race [reis] n.比赛 竞赛 竞争 人种 种族 民族 v.赛跑 竞赛

racial ['reiʃəl] adj.种族的 人种的

rack [ræk] n.搁物架 行李架 拷问台 齿轨 vt.折磨 使痛苦 拷问 vi.顶风飞行

racket ['rækit] n.球拍 喧闹纷乱

radar ['reidɑ:] n.雷达 无线电探测器

radiate ['reidieit] vi.发射光线 辐射 散发 辐射

radiation [.reidi'eiʃən] n.放射 发射 辐射能

radio ['reidiəu] n.无线电 收音机 v.用无线电通讯

radioactive [.reidiəu'æktiv] adj.放射性的

radioactivity [.reidiəuæk'tiviti] n.放射性 放射(现象) 辐射

radish ['rædiʃ] n.小萝卜

radium ['reidiəm] n.镭

radius ['reidiəs] n.半径 桡骨 半径范围 辐射区

rag [ræg] n.破布 碎布 抹布 碎屑 石板瓦 散拍乐曲 vt.责骂 欺负 以散拍乐演奏

rage [reidʒ] n.(一阵)狂怒 盛怒 愤怒 情绪激动 风行一时 热量 vi.震怒 风行 席卷

raid [reid] n.袭击 突然搜查 劫掠 v.劫掠 攫取 突击搜捕 袭击

rail [reil] n.横条 横杆 铁轨 栏杆 扶手 vt.以横木围栏 铺铁轨 vi.责骂 抱怨

railroad ['reilrəud] n.铁路=railway(英) vi.由铁路运输 在铁路公司工作 vt.由铁道运输 使(议案等)草草通过

railway ['reilwei] n.铁路 铁道

rain [rein] n.雨 雨水 (复数)雨季 vi.下雨 vt.使大量落下

rainbow ['reinbəu] n.虹 彩虹 adj.五彩缤纷的

rainy ['reini] adj.下雨的 多雨的

raise [reiz] vt.举起 引起 提高 升起 饲养 提出 n.上升 增高 高地

rake [reik] n.耙子 集赌金工具 向船尾的倾斜 vi.耙 用耙子 搜索 向船尾 了望

range [reindʒ] n.排 行 山脉 炉灶 范围 行列 射程 v.排列 归类于 延伸 漫游

rank [ræŋk] n.排 横行 社会阶层 队列 排名 等级 军衔 阶级 adj.繁茂的 恶臭的 讨厌的 vt.排列 归类于

rapid ['ræpid] adj.快的 迅速的 急促的 n.急流

rapidly ['ræpidli] adv.迅速地

rare [rɛə] adj.稀薄的 稀有的 稀罕的 罕见的 珍贵的 adj.煎得嫩的

rarely ['rɛəli] adv.很少 难得

rat [ræt] n.老鼠 耗子 卑鄙的人 变节者 vi.背叛 捕鼠 vt.弄蓬松 告发

rate [reit] n.比率 速度 价格 等级 vt.估价 认为 定等级 责骂 vi.受欢迎 受推崇 责骂

rather ['rɑ:ðə] adv.宁可 宁愿 相当 有些 反而

ratio ['reiʃiəu] n.比 比率

rational ['ræʃənəl] adj.理性的 合理的 出于理性的 能推理的 n.有理数

raw [rɔ:] adj.未煮过的 生的 未加工的 原始的 不熟练的 寒冷的 n.触到某人的痛处

ray [rei] n.光线 射线 辐射线 雷(男子名)

razor ['reizə] n.剃刀

reach [ri:tʃ] vt.抵达 到达 伸出 延伸 取得联系 vi.达到 延伸 n.延伸 范围 河段

react [ri'ækt] vi.起反应 起作用 有影响 反攻 vt.作出反应

reaction [ri'ækʃən] n.反应 反作用 反作用力 化学反应

read [red,ri:d] vt.阅读 看懂 理解 识别 vi.读 读起来 研究 adj.博学的 n.阅读

reader ['ri:də] n.读者 读物 读本

readily ['redili] adv.乐意地 无困难地 不迟疑地 迅速地 轻易地

reading ['ri:diŋ] n.读 阅读 读书 adj.阅读的

ready ['redi] adj.准备好的 愿意的 情愿的 现成的 vt.做好...的准备

real ['ri:əl] adj.真的 现实的 实际的 真实的 adv.非常 n.真实的事 真实

reality [ri'æləti] n.现实 真实 实际

realize ['riəlaiz] vt.实现 认识到 了解 意识到 净得

really ['riəli] adv.真正地 实在 事实上

realm [relm] n.王国 国土 领域

reap [ri:p] vt. & vi.收割 收获 获得

rear [riə] n. 后部 后面 背面 臀部 adj. 后面的 背面的 后方的 vt. 养育 饲养 举起 建立 抚养 培养 栽种

vi. 高耸 用后腿站立

reason ['ri:zn] n.理由 原因 理智 理性 vi.推理 规劝 下判断 vt.说服 总结

reasonable ['ri:znəbl] adj情合理的 公道的 适度的 通情达理的

rebel ['rebl,ri'bel] vi.造反 n.造反者 叛徒 起义者 vi.造反 反抗 反感 adj.造反的 反抗的 反叛者的

rebellion [ri'beljən] n.造反 叛乱 反抗 谋反

recall [ri'kɔ:l] vt.回想 叫回 收回 召回 与...相似 恢复 n.回忆 召回 取消

receipt [ri'si:t] n.收到 收条 收据

receive [ri'si:v] vt.收到 得到 接待 接受 迎接 容纳 遭受 相信 vi.接到 获得

receiver [ri'si:və] n.收受者 收件人 接收器 收款员 接待者 听筒

recent ['ri:snt] adj.新近的 最近的

recently ['ri:sntli] adv.最近 新近

reception [ri'sepʃən] n.接待 招待会 接受 接收 欢迎

recite [ri'sait] vt.背诵 朗诵 逐一例举 叙述或回答问题 vi.背诵 朗诵 回答问题

recognition [.rekəg'niʃən] n.认出 识别 承认 感知 知识

recognize ['rekəgnaiz] vt.认识 认出 承认 认可 承认 意识到 表示感激

recollect [.rekə'lekt] vt.回忆 追忆 想起 回想 忆起

recommend [.rekə'mend] vt.推荐 介绍 劝告 建议 使成为可取 使受欢迎

recommendation [.rekəmen'deiʃən] n.推荐 介绍 劝告

record ['rekɔ:d,ri'kɔ:d] n.记录 纪录 履历 唱片 vt.记录 (将声音等)录下 表明 adj.关于

recorder [ri'kɔ:də] n.记录者 录音机 (英国某些市镇的)首席法官 市法院法官 竖笛

recover [ri'kʌvə] vt.重新获得 挽回 找回 vi.恢复 复原 补偿

recovery [ri'kʌvəri] n.重获 痊愈 恢复 复原

red [red] adj.红色的 n.红色

reduce [ri'dju:s] vt.减少 减小 简化 分解 减低 归约 使...陷入某种状态 vi.减少

reduction [ri'dʌkʃən] n.减少 减小 缩减 (化学)还原反应 (数学)约分

reed [ri:d] n.芦苇 芦杆 芦丛 芦笛 簧片 (Reed)里德(姓氏)

reel [ri:l] n.卷轴 卷筒 线轴 vi.蹒跚地走 旋转 骚乱 退缩 摇晃 vt.卷 卷... 轴上 旋转

refer [ri'fə:] vt.使求助于 把...提交 把...归因 归类为 vi.谈到 谈及 咨询 参考

reference ['refrəns] n.参考 出处 提及 参照 推荐人 推荐函 vt.提供参考

refine [ri'fain] vt. & vi.精炼 提纯 净化 使优雅 vi.被提纯 改进

reflect [ri'flekt] vt.反射 反映 思考 归咎

reflection [ri'flekʃən] n.反射 映象 反映 折射 沉思 影响

reflexion [ri'flekʃən] n.反射 映象 反映 反射波 折射 考虑 想法 =reflection

reform [ri'fɔ:rm] n.改革 改正 感化 vt.改革 改正 使悔改 vi.改良 悔改

refresh [ri'freʃ] vt.使清新 使...生气蓬勃 提起精神 恢复精神 vi.恢复精神

refreshment [ri'freʃmənt] n.恢复 精神爽快 提神之事物 (复数)茶点 点心 便餐

refrigerator [ri'fridʒə.reitə] n.冰箱 冷藏库

refuge ['refju:dʒ] n.避难(处) 庇护(所) 庇护者 v.庇护 避难(所)

refusal [ri'fju:zəl] n.拒绝 回绝

refuse ['refju:s,ri'fju:z] vt.拒绝 adj.无用的 n.废物 垃圾

refute [ri'fju:t] vt.驳斥 反驳 驳倒 证明

regard [ri'gɑ:d] vt.把...看作 和...有关 尊敬 尊重 视为 注视

regarding [ri'gɑ:diŋ] prep.关于 至于

regardless [ri'gɑ:dlis] adv.不顾一切地 adj.不注意的 粗心的

region ['ri:dʒən] n.地区 地带 领域 行政区

register ['redʒistə] vt.记录 登记 注册 挂号 n.暂存器 记录 登记簿 (语)语域 通风调节设备

regret [ri'gret] vt.懊悔 为...感到遗憾 后悔 惋惜 抱歉 n.懊悔 遗憾 后悔 抱歉

regular ['regjulə] adj.规则的 有规律的 整齐的 正规的

regularly ['regjuləli] adv.有规律地 定期地

regulate ['regju.leit,'regjuleit] vt.管理 控制 调整

regulation [.regju'leiʃən] n.规则 规章 管理 adj.规定的 官方的

rehearsal [ri'hə:sl] n.排练 彩排 排演 练习

reign [rein] n.君主统治(时期) 执政 王朝 支配 vi.当政 统治 占优势 支配 盛行

rein [rein] n.缰绳 统驭 支配 机会 vi.驾驭 控制 vt.以缰绳控制 统制

reinforce [.ri:in'fɔ:s] vt.增援 支援 加强 vi.得到加强

reject [ri'dʒekt] vt.拒绝 丢掉 丢弃 驳回 排斥 n.被拒之人 不及格者

rejoice [ri'dʒɔis] vi.使...欣喜 高兴

relate [ri'leit] vt.叙述 使联系 vi.追溯到 有关联 建立关系 了解

relation [ri'leiʃən] n.关系 联系 家属 亲戚 参考 陈述 故事

relationship [ri'leiʃənʃip] n.关系 联系

relative ['relətiv] adj.有关系的 相对的 比较的 涉及到的 n.亲属 同类事物 相关物

relatively ['relətivli] adv.相对地 比较地

relativity [.relə'tiviti] n.相关性 相对性

relax [ri'læks] vt.使松弛 放松 放宽(限制等) vi.松弛 放松 松懈

release [ri'li:s] vt.释放 放松 发表 准予发表 让与 发射 n.释放 让渡 发行

relevant ['relivənt] adj.有关的 相关的 贴切的 切题的 中肯的

reliability [ri.laiə'biliti] n.可靠性

reliable [ri'laiəbl] adj.可靠的 可信赖的

reliance [ri'laiəns] n.信任 信赖 信心 所信赖的人或物

relief [ri'li:f] n.减轻 救济(品) 援救 解除 安慰 浮雕 对比 adj.提供救济的 文本的

relieve [ri'li:v] vt.减轻 解除 救济

religion [ri'lidʒən] n.宗教 宗教信仰

religious [ri'lidʒəs] adj.宗教的 虔诚的 细心的 n.修道院成员 僧侣

reluctant [ri'lʌktənt] adj.不愿的 勉强的

rely [ri'lai] vi.依赖 依靠 信赖

remain [ri'mein] vi.剩下 余留 逗留 保持 依然 留下 n.遗迹 剩余物 残骸

remains [ri'meinz] n.残余(余物 遗物 化石) 余额 废墟

remark [ri'mɑ:k] vt. & vi.评论 注意 述及 谈论 n.备注 评论 注意

remarkable [ri'mɑ:kəbl] adj.异常的 非凡的 显著的 值得注意的

remedy ['remidi] n.药物 治疗法 补救 vt.治疗 补救 矫正

remember [ri'membə] vt.记得 想起 记住

remind [ri'maind] vt.提醒 使想起

remote [ri'məut] adj.相隔很远的 遥远的 冷淡的 偏僻的 远程的 (感情等)距离很大 n.远程操作

removal [ri'mu:vəl] n.移动 移居 迁移 除掉

remove [ri'mu:v] vt.移动 搬开 脱掉 除去 开除 取下 拿走 vi.迁徙 离开 搬迁 n.间距 移动

render ['rendə] vt.表示 给予 使得 使成为 提供 报答 着色 执行 实施 vi.给予补偿 n.交纳 粉刷 打底

renew [ri'nju:] vt.使更新 复始 使恢复 补充 vi.更新 重新开始

rent [rent] n.租金 租 vi.出租 vt. 出租 租用

repair [ri'pɛə] vi.去 集合 vt.修理 补救 修补 补偿 n.修理 修补 补救 n.维修工作 维修状态

repeat [ri'pi:t] vt.重说 重做 重复 复述 vi.重复 n.重复 反复 adj.重复的 反复的

repeatedly [ri'pi:tidli] adv.重复地 再三地 一再

repent [ri'pent] vi.悔悟 悔改 后悔 vt.后悔

repetition [.repi'tiʃən] n.重复 反复

replace [ri'pleis] vt.把...放回 取代 更换

reply [ri'plai] vi. & n.回答 答复

report [ri'pɔ:t] vt. & vi.报告 汇报 n.传闻 报告 成绩单 爆炸声

reporter [ri'pɔ:tə] n.记者 通讯员

represent [.repri'zent] vt.描绘 代表 象征 表现 表示 说明 阐明 vi.提出异议

representative [repri'zentətiv] adj.代表性的 代议制的 典型的 n.代表 众议员 典型

reproach [ri'prəutʃ] n.责备 耻辱 vt.责备 申斥 指责

reproduce [.ri:prə'dju:s] vt. & vi.繁殖 生殖 再生 复制

republic [ri'pʌblik] n.共和国 共和政体

republican [ri'pʌblikən] adj.共和国的 n.拥护共和政体的人 Republican:共和党人 共和党的

reputation [.repju'teiʃən] n.名誉 名声 好名声

request [ri'kwest] n. & vt.请求 要求

require [ri'kwaiə] vt.需要 要求 命令

requirement [ri'kwaiəmənt] n.需要 要求 必要条件

rescue ['reskju:] n.援救 解救 营救队 vt.援救 救出 营救

research [ri'sə:tʃ] n. & vi.调查 探究 研究

researcher [ri:'sə:tʃə] n.调查者 探究者 研究人员

resemble [ri'zembl] vt.像 类似 相似

reserve [ri'zə:v] vt.储备 保留 预定 延期 n.预备品 贮存 候补 克制 含蓄

reservior ['rezəvwɑ:] n.水库 蓄水池 积蓄 储藏 = reservoir

residence ['rezidəns] n.居住 驻扎 住处 住宅

resident ['rezidənt] adj.居住的 n.居民 住客 高级专科住院实习医生

resign [ri'zain] vt.放弃 辞职 使顺从 vi.辞职 顺从

resignation [.rezig'neiʃən] n.放弃 辞职 辞呈 顺从

resist [ri'zist] vt. & vi.抵抗 抵制 耐得住 压制 n.保护膜

resistance [ri'zistəns] n.抵抗 抵抗力 抵制 反抗 反抗行动 阻力 电阻 反对

resistant [ri'zistənt] adj.抵抗的 反抗的 n.抵抗者

resolution [.rezə'lu:ʃən] n.坚决 坚定 决定 决心 决议 果断 刚毅 解决 解答 病理状态的减退 分辨率 清晰度

resolve [ri'zɔlv] vt.解决 决心 决定 分离 表决 vi.分离 决定 n.决心 决定之事

resort [ri'zɔ:t] n. (渡假)胜地 手段 求助 凭借 vi.求助 凭借 诉诸 常去

resource [ri'sɔ:s] n.资源 物力 办法 谋略 消遣

respect [ri'spekt] n.尊敬 敬重 关系 方面 vt.尊敬 尊重 关于 涉及

respectful [ri'spektfəl] adj.恭敬的 尊重的 有礼貌的 谦恭的

respective [ri'spektiv] adj.各自的 各个的 分别的

respectively[ri'spektivli] adv.各自地 分别地

respond [ri'spɔnd] vi.作答 响应 回复 反应 应答 n.拱柱

response [ri'spɔns] n.作答 回答 响应 反应

responsibility [ri.spɔnsə'biliti] n.责任 责任心 可信度 职责

responsible [ri'spɔnsəbl] adj.有责任的 尽责的 责任重大的 可靠的

rest [rest] n.休息 安静 静止 其余的部分 其余的人 adj.其余的 vi.搁(在) 休息 信任 vt.使休息 依赖 基于某事

restaurant ['restərɔnt] n.餐馆 饭店 菜馆

restless ['restlis] adj.不安定的 焦虑的 静不下来的 得不到休息的

restore [ri'stɔ:] vt.恢复 归还 修补 翻新

restrain [ri'strein] vt.抑制 遏制 限制 阻止 束缚 剥夺

restraint [ri'streint] n.抑制 遏制 克制 束缚

restrict [ri'strikt] vt限制 限定 约束

restriction [ri'strikʃən] n.限制 限定 约束

result [ri'zʌlt] n.成果 结果 成绩 答案 vi.发生 结果 产生 致使

resume [ri'zju:m] vt.恢复 重新开始 再继续 n.简历 履历 摘要

retain [ri'tein] vt.保持 保留 保有 记住

retell [ri:'tel] vt.再讲 重述 复述

retire [ri'taiə] vi.退下 引退 撤退 退休 就寝 n.退休 退职

retreat [ri'tri:t] vi.(被迫)退却 后退 撤退 向后倾 n.休息寓所 撤退 隐居

return [ri'tə:n] n.返回 归还 报答 报告 vi.返回 归还 回来 adj.返回的

reveal [ri'vi:l] vt.展现 揭示 显示 揭露 透露 n.(外墙与门或窗之间的)窗侧 门侧 (汽车的)窗框

revenge [ri'vendʒ] vt.替...报仇 报复 n.报仇 报复 复仇愿望 获得满足的机会

reverse [ri'və:s] vt.颠倒 翻转 逆转 倒退 n.背面 相反 失败 挫折 adj.反面的 相反的 颠倒的

review [ri'vju:] vt.再检查 温习 检讨 评论 检阅 vi.写评论 n.复习 检讨 回顾 评审

revise [ri'vaiz] vt.修订 校订 校正 修改 n.校订 修正 再校稿

revolt [ri'vəult] n.叛乱 反抗 造反 反感 vi.叛乱 反抗 造反 起反感 vt.使...恶心

revolution [.revə'lu:ʃən] n.革命 旋转 绕转 转数

revolutionary [.revə'lu:ʃənəri] adj.革命的 n.革命者

reward [ri'wɔ:d] n.报答 报酬 赏金 vt.报答 给...报酬 奖赏

rhythm ['riðəm] n.韵律 格律 节奏

rib [rib] n.肋 肋骨 嘲笑 vt.嘲笑

ribbon ['ribən] n.缎带 丝带 带 色带 带状物 vt.用丝带装饰 撕碎

rice [rais] n.稻 米 饭

rich [ritʃ] adj.富的 丰富的 浓厚的 贵重的 富饶的

rid [rid] vt.使摆脱 使去掉 免除 清除 使...获自由

riddle ['ridl] n.谜 谜语 vt.解谜 出谜题 充满 打洞 筛选 vi.出谜题

ride [raid] n. 乘坐 乘车 骑马 搭便车 vi.骑 骑马 乘车 乘坐 经受住

rider ['raidə] n.骑马的人 乘车的人 骑士 附件 扶手

ridge [ridʒ] n.脊 岭 山脉 垄

ridiculous [ri'dikjuləs] adj荒谬的 可笑的

rifle ['raifl] n.步枪 来复枪 v.洗劫 抢劫

right [rait] adj. 正确的 正直的 合适的 垂直的 右面的 正常的 正面的 adv. 对 正好 恰当 径直地 正确地 立即 非常 向右边 n. 权利 右 正义 实况 vt. 纠正 扶直 公正对待 伸冤 vi. 复正

rigid ['ridʒid] adj.刚硬的 僵硬的 刻板的 严格的

ring [riŋ] n.环形物(如圈、环等) 戒指 圈 铃声 vt.包围 挂铃铛 vi.鸣响 回响 在空气中螺旋上扬 形成圈 按铃

ripe [raip] adj.熟的 成熟的 上年纪的 合适的 时机成熟的

ripen ['raipən] vt.使成熟 vi.成熟

rise [raiz] vi.起立 升起 上涨 起身 n.上升 增加 小山 斜坡

risk [risk] n.风险 危险 冒险 vt. 冒...的危险

rival ['raivəl] n.对手 同伴 竞争者 adj.竞争的 vt.与...相匹敌 比得上 vi.竞争

river ['rivə] n.江 河 水道

road [rəud] n.路 道路 公路

roar [rɔ:] vi.吼叫 大声说出 喧闹 呼喊 vt.呼喊 使轰鸣 n.吼 咆哮 轰鸣 喧闹

roast [rəust] vt. & vi.烤 炙 烘焙 adj.烘烤的 烤过的 n.烤肉 烘烤

rob [rɔb] vt. & vi.抢劫 劫掠

robber ['rɔbə] n.强盗 盗贼 robbery ['rɔbəri] n.抢劫 劫掠 盗取

robe [rəub] n.长袍 长衣 浴衣 v.(使)穿上长袍等

robot ['rəubɔt] n.机器人 自动机

rock [rɔk] vt.摇 使动摇 使震惊 使动心 vi.摇 来回摇动 跳摇摆舞 n.岩石 巨石 石头 摇滚乐 篮球

rocket ['rɔkit] n.火箭 火箭发动机 烟火 v.发射火箭 猛涨

rod [rɔd] n.杆 竿 棒 戒鞭 测量杆 视网膜 杆状菌

role [rəul] n.角色 作用 任务 影响程度

roll [rəul] n.卷 滚动 名单 压路机 颤音 vt.滚 绕 转动 摇摆 展开 vi.滚 卷 绕 运载 完成 涌入

roller ['rəulə] n.滚柱 滚筒 滚轴 滚转机

Roman[rɔmə] n.古罗马人 罗马天主教会 罗马文字 adj.罗马的 罗马帝国的

romantic [rə'mæntik] adj.浪漫的 传奇的 n.浪漫的人

roof [ru:f] n.屋顶 车顶 顶 顶盖 vt.给...覆以屋顶 覆盖

room [ru:m] n.房间 空间 地位 余地

root [ru:t] n.根 根子 根源 祖先 vi.生根 扎根 翻寻 起源于

rope [rəup] n.绳 索 v.捆 绑

rose [rəuz] n.蔷薇花 蔷薇科植物 玫瑰 v.(rise的过去式)起立 上升 升起 adj.玫瑰色的 玫瑰的

rot [rɔt] vt.烂 腐坏 腐烂 腐蚀 败坏 n.腐烂 使...枯朽 衰弱

rotary ['rəutəri] adj.旋转的 转动的 n.旋转式机器 环形交叉路

rotate [rəu'teit] vi.旋转 vt.使旋转 轮流 adj.车轮状的

rotation [rəu'teiʃən] n.旋转 转动 循环

rotten ['rɔtn] adj.腐烂的 腐朽的 发臭的

rough [rʌf] adj. 表面不平的 粗略的 粗糙的 粗暴的 艰难的 讨厌的 不适的 adv. 粗鲁地 粗略地 n.高低不平且杂草丛生的地区 草图 暴徒 vt. 殴打 草拟 adv. 粗暴地

roughly ['rʌfli] adv.粗糙地 粗略地 概略地 粗鲁地

round [raund] adj .圆的 微胖的 完整的 坦率的 圆满完成的 n. 兜圈 圆 (某种圆形)交谊舞 常规活动 一局(轮)

adv. 环形地 vt. 弄圆 绕行 使...完全 使成为整数 vi. 变圆(胖) 弯曲 prp. 围绕 遍及

rouse [rauz] vt.唤醒 唤起 惊起 鼓舞 激起 n.觉醒 奋起

route [ru:t] n.路 路线 路程 航线 途径 vt.按路线发送 改变路线

routine [ru:'ti:n] n.例行公事 常规 无聊 adj.日常的 常规的 例行的 乏味的

row [rəu,rau] n.(一)排 (一)行 船游 吵闹 vt.划船 成排 vi.划船 吵架

royal ['rɔiəl] adj.王室的 皇家的 盛大的 极度的 n.贵族 顶帆

rub [rʌb] vt.摩擦 擦 搓 涂抹上 使不愉快 vi.摩擦 n.摩擦 困难 障碍

rubber ['rʌbə] n.橡皮(擦子) 橡胶 合成橡胶 adj.橡胶的

rubbish ['rʌbiʃ] n.垃圾 废物 废话

rude [ru:d] adj.加工粗糙的 粗野的 无礼的

rug [rʌg] n.小地毯 地毯 毛毯 毯子 旅行毯

ruin ['ruin] n. (pl.)废墟 毁灭 vt.毁坏 毁灭 vi.(使)破产 毁灭

rule [ru:l] n.统治 规定 规则 习惯 惯例 标准 vi.统治 立法 vt.裁决 统治 画直线

ruler ['ru:lə] n.统治者 尺 直尺

rumour ['ru:mə] n.谣言 谣传 传闻 低语 (美)=rumor vt.传闻 谣传

run [rʌn] vi. 跑 奔 行驶 流 竞赛 运转 遭遇 持续 熔化 褪色 蔓延 传播 vt. 跑 放牧 经营 走私 使运转 浇铸 使遭受 n. 奔跑 路程 趋向 adj. 熔化的 浇铸的 洄游的

runner ['rʌnə] n.赛跑的人 跑步者

rural ['ru:rəl] adj.农村的 田园的

rush [rʌʃ] vi.冲(进) 奔 vt.催促 匆忙 猛冲 n.匆促 冲进 急流 猛冲 急需 突击 adj.紧急的

Russian ['rʌʃən] adj. 俄国的 俄罗斯的 n.俄国人 俄语

rust [rʌst] n.锈 锈 vi.生锈 氧化 变成红棕色 vt.使...生锈 损害

rusty ['rʌsti] adj.生锈的 变迟钝的

## S

sack [sæk] n.袋 麻袋 开除 劫掠 vt.解雇 劫掠 把...装入袋子

sacred ['seikrid] adj.上帝的 神圣的 受尊重的

sacrifice ['sækrifais] n.牺牲 供俸 祭品 vt.牺牲 祭祀 贱卖 vi.献祭 (棒球)作牺牲一击

sad [sæd] adj.悲哀的 令人悲痛的 伤心的 不能令人满意的

saddle ['sædl] n.鞍子 马鞍 车座 山脊 拖具 vt.装以马鞍 使负担 vi.装马鞍

sadly ['sædli] adv.悲痛地 悲哀地 悲惨地 悲伤地

sadness ['sædnis] n.悲痛 悲哀

safe [seif] adj.安全的 谨慎的 n.保险箱

safely ['seifli] adv.安全地 可靠地

safety ['seifti] n.安全 保险 安全打 vt.保护

sail [seil] n.帆 船 乘船航行 vi. (驾船)航行 vt.乘船航行 浮游 启航

sailor ['seilə] n.水手 海员 水兵 扁平的硬边草帽

saint [seint] n.圣人 圣徒 基督教徒 vt.把...封为圣人

sake [seik] n.缘故 理由 利益 好处

salad ['sæləd] n.色拉 凉拌生菜 杂烩 莴苣 生菜

salary ['sæləri] n.薪金 薪水 vt.给...薪水

sale [seil] n.卖 削价出售 拍卖 销售额(量),

salesman ['seilzmən] n.售货员 推销员

salt [sɔ:lt] n.盐 盐类 刺激 adj. 含盐的 辛辣的 vt.加盐于 用盐腌 abbr. (=Strategic Arms Limitation Talks) 限制战略武器会谈

salute [sə'lu:t] n.招呼 行礼 vi.行礼 致意 问候

same [seim] adj.同一的 相同的 adv.同样地 pron.同样的人或事

sample ['sæmpl] n.样品 实例 标本 vt.采样 品尝 adj.样例的

sand [sænd] n.沙 沙子 沙滩 v.撒沙 以沙掩盖

sandwich ['sændwitʃ] n.三明治 vt.夹入 插入 夹在中间

sandy ['sændi] adj.沙的 含沙的 沙地的 多沙的 沙色的 不稳固的

satellite ['sætəlait] n.卫星 人造卫星 卫星城 随从

satisfaction [.sætis'fækʃən] n.满意 快事 乐事 赔偿 妥善处理 确信

satisfactory [.sætis'fæktəri] adj.令人满意的 良好的

satisfy ['sætisfai] vt.满足 使满意 使...信服 vi.足够 取悦

Saturday ['sætədi] n.星期六

sauce [sɔ:s] n.调味汁 酱汁 无理的话(或举动) vt.给...调味 使...增加趣味 无理

saucer ['sɔ:sə] n.茶托 浅碟

sausage ['sɔ:sidʒ] n.香肠 腊肠

save [seiv] vt.救 解救 节省 储蓄 保存 vi.节省 储蓄 救球 prep.除...之外 conj.除...之外 只有 n.救援者 救球 节省

saving ['seiviŋ] n.搭救 挽救 节约 存款 (复数)savings:储蓄金 存款 prep.除...之外 conj.除了 除去 save的现在分词

saw [sɔ:] n.锯子 vt.锯 锯开 用锯子锯 vbl.看见(see的过去式)

say [sei] vt.说 说明 比如说 讲 表达 假设 指示 vi.说 n.发言权 adv.大约

scale [skeil] n.天平 磅秤 秤 鳞 刻度 衡量 数值范围 标度 比例 大小 v.依比例决定 攀登

scan [skæn] vt.细看 浏览 扫描 详细调查 押韵 vi.押韵 n.扫描 押韵 细查

scar [skɑ:] n.伤疤 伤痕 创伤 断崖 v.结疤 使...有伤痕 痊愈

scarce [skɛəs] adj.缺乏的 希有的 不足的 罕见的 adv.勉强

scarcely ['skɛəsli] adv.仅仅 几乎不 简直没有 勉强

scare [skɛə] vt.惊吓(惊恐 惊慌) vi.受惊 n.惊吓(惊恐 惊慌)

scarf [skɑ:f] n.围巾 头巾 领带

scatter ['skætə] vt.使消散 撒 散播 驱散 vi.分散 消散 n.散布 零星少量

scene [si:n] n.发生地点 道具 场 景 情景

scenery ['si:nəri] n.舞台布景 风景 布景 背景

scent [sent] n.气味 香味 痕迹 香水 vt.闻出 发觉 使充满味道 得到...的暗示 vi.嗅闻 循着遗臭追踪 散发气味

schedule['skedʒul] n.时间表 计划表 一览表 计划 vt.安排 计划 预定 编制目录

scheme [ski:m] n.计划 规划 方案 诡计 阴谋 v.计画 设计 体系 结构 图谋

scholar ['skɔlə] n.学者(尤指文学方面) 获奖学金的学生

scholarship ['skɔləʃip] n.学业成绩 奖学金 学问 学识

school [sku:l] n.学校 学院 学派 鱼群 vt.教育 训练 磨炼 vi.喂食鱼群 游来游去

science ['saiəns] n.科学 科学研究

scientific [.saiən'tifik] adj.科学(上)的

scientist ['saiəntist] n.(自然)科学家

scissors ['sizəz] n.剪刀 剪子

scold [skəuld] vi.责骂 vt.申斥 n.老爱责骂的人 训斥

scope [skəup] n.范围 余地 机会 能力 眼界 vt.仔细研究

score [skɔ:] n.得分 比数 成绩 刻痕 二十 乐谱 vt.记分 刻划 划线 得分 批评 给...谱曲 vi.得分 记分 得胜

scorn [skɔ:n] n.轻蔑 嘲笑 被叱责的人 vt.轻蔑 不屑做

scout [skaut] n.侦察员 侦察机 童子军 守卫 vt.侦察 物色人才 vi.寻找 物色

scrape [skreip] vt. & vi. & n.刮 擦 刮擦声

scratch [skrætʃ] vt. & vi.搔 抓 搔痒 抹掉 n.搔 抓 抓痕 抓的声音 乱写 adj.使皮肤发痒的 潦草的

scream [skri:m] vi.尖叫 呼啸 大笑 n.尖叫声

screen [skri:n] n.屏 屏幕 银幕 屏风 vt.掩蔽 放映 检查 选拔 遮蔽

screw [skru:] n.螺旋 螺旋状物 螺丝 螺丝钉 吝啬鬼 压迫 转动 监狱看守 vt.拧紧 扭紧 压榨 欺骗 强迫 vi.旋转

sea [si:] n.海 海洋

seal [si:l] n.封蜡 印记 印章 封条 海豹 vt.封 盖印 封闭 猎海豹

seaman ['si:mən] n.海员 水手 水兵

seaport ['si:pɔ:t] n.海港 港市

search [sə:tʃ] vt.在...中搜寻 搜查 探求 调查 n.搜寻 探究

season ['si:zn] n.季 季节 时节 旺季 上演期 vt.加调味料调味 使...适用于 vi.使...被调味

seat [si:t] n.座 座部 位子 所在地 中心 vt.使坐下 可容纳 vi.坐

second ['sekənd] num.第二 adj.二等的 第二的 另一个的 n.第二 秒 vt.支持

secondary ['sekənderi] adj.第二的 次要的 中级的 中等的 n.次要位置 副手

secondly ['sekəndli] adv.第二(点) 其次

secret ['si:krit] adj.秘密的 隐蔽的 难懂的 机密的 n.秘密 诀窍

secretary ['sekrətri] n.秘书 书记 大臣 部长 写字台

section ['sekʃən] n.切片 一段 部门 零件 部分 章节 区域 路段 断面 vt.划分 切开 vi.划分

secure [si'kjuə] adj. 安心的 安全的 无虑的 牢靠的 稳妥的 vt. 固定 获得 使...安全

vi. ( 海上工作人员 ) 停止工作 (船) 抛锚 停泊

security [si'kju:riti] n.安全 安全感 防护措施 保证(金) 抵押(品) 债券 证券

see [si:] vt.看见 遇见 看出 经历 明白 确保 视为 拜访 陪同 vi.注意 想象 了解 调查 n.主教教区 主角权限

seed [si:d] n.种(子) 籽 萌芽 vi.(植物)结实 播种 vt.播种

seek [si:k] vt.寻找 探索 试图 vi.寻找

seem [si:m] vi.好像 似乎

seize[si:z] vt.抓住 逮捕 夺取 攻占 理解 侵袭 vi.抓住 卡住 终止

seldom ['seldəm] adv.很少 不常 难得 adj.很少的

select [si'lekt] vt. & vi.选择 挑选 选拔 adj.精选的 优等的 n.被挑选出来的人或物

selection [si'lekʃən] n.选择 挑选 精选物 挑选出来的人或事

self [self] n.自我 自己 本身

selfish ['selfiʃ] adj.自私的 利己的

sell [sel] vi.卖 销售 促销 vt.出让 背叛 出卖 出售 n.欺骗

seller ['selə] n.卖者 售货者 行销货 畅销品

semester [si'mestə] n.半年 学期 半学年

semiconductor [.semikən'dʌktə] n.半导体

senate ['senit] n.参议院 上院

send [send] vt.送 传 寄 派遣 发射 vi.放送 n.波浪的推力

senior ['si:njə] adj.年长的 地位较高的 高级的 资深的 n.年长者 上司 毕业班学生

sense [sens] n.感官 感觉 见识 侦测 理智 意义 vt.感觉 了解 检测

sensible ['sensəbl] adj.感觉得到的 明智的 可察觉的 意识到的 实用的 n.可感知物

sensitive ['sensitiv] adj.敏感的 灵敏的 易受伤害的 感光的 善解人意的 n.敏感的人

sentence ['sentəns] n.判决 句子 意见 宣判 刑罚 命题 vt.宣判 判决 使遭受

separate ['sepəreit] adj.分离的 个别的 各自的 单独的 n.分开 抽印本 v.分开 隔开 分居

separately ['sepəritli] adv.分离地 分别地 个别地 独立地

separation [.sepə'reiʃən] n.分离 分开 分居 缺口

September [səp'tembə] n.九月

sequence ['si:kwəns] n.连续 继续 次序 序列 一系列 顺序 一组镜头 vt.安排顺序

series ['siəri:z] n.连续 系列 系列节目 丛书

serious ['siəriəs] adj.严肃的 认真的 庄重的 严重的 危急的

seriously ['siriəsli] adv.严肃地 认真地 严重地

servant ['sə:vənt] n.仆人 佣人 雇工

serve [sə:v] vt.为...服务 招待 可作...用 对待 发球

service ['sə:vis] n.服务 行政部门 公务部门 服役 发球 发球方式 vt.维护 保养

session ['seʃən] n.会议 一段时间 开庭期 市盘 学期 adj.短期的

set [set] vt.放 安置 设定 vi.(太阳)落下去 n.组合 集合 设定 一套 一副 一局比赛

setting ['setiŋ] n.安装 调整 环境 放置 周围 (为诗等谱写的)乐曲

settle ['setl] vt.安排 安放 安顿 调停 解决 决定 付清(欠款) vi.定居 降落 n.有背的长凳

settlement ['setlmənt] n.解决 殖民 殖民地 安置 协议 结算 住宅区

seven ['sevn] num.七 七个(人或物)

seventeen [.sevn'ti:n] num.十七 十七个

seventh ['sevnθ] num.第七 七分之一 n.第七 七分之一 adj.第七的 adv.第七

seventy['sevnti] num.七十 七十个

several ['sevərəl] adj.几个 数个 各自的 pron.若干

severe [si'viə] adj.严格的 严厉的 剧烈的 严重的 严峻的

severely [sə'virli] adv.严厉地 严格地 严重地(艰难地 紧凑地 纯朴地)

sew [səu] vt.缝制 缝合 vi.缝纫

sex [seks] n.性别 性 性欲 vt.区别...性别 引起...性欲

shade [ʃeid] n.荫 遮光物 荫凉处 色度 稍微 vi.荫蔽 遮蔽 渐变 微减

shadow ['ʃædəu] n.阴影 阴 影子 阴暗 预兆 跟踪盯梢的人

shady ['ʃeidi] adj.成荫的 阴凉的 可疑的

shake [ʃeik] vt. 摇 使震动 挥拳 摇晃 去除 动摇 排掉(灰尘) 握手 vi. 摇晃 颤抖 跌跌撞撞 握手 发颤音

n 摇动 摇晃 震动 地震 颤抖(音) 奶昔 短时间

shall [ʃæl] aux.v.将要 必须 应 应该 一定

shallow ['ʃæləu] adj.浅的 浅薄的 肤浅的 浅显易懂的 n.(pl.)浅滩 浅处 v.变浅

shame [ʃeim] n.羞耻 羞愧 羞辱 遗憾的事 vt.使羞愧

shampoo [ʃæm'pu:] vt.用洗发剂洗 洗(头发 地毯等) n.洗头 洗发剂 洗涤剂

shape [ʃeip] n.形状 情况 形式 定形 身材 vt.定形 使 ... 成形 塑造

share [ʃɛə] n.份 份额 股份 部份 分担 犁头 vt.分享 共有 分配 vi.分享

sharp [ʃɑ:p] adj.锋利的 敏锐的 刺耳的 急剧的 清晰的 adv.[时刻]正 准 n.升半音 v.升音

sharpen ['ʃɑ:pən] vt.削尖 使敏锐 使...尖锐 变为锐利 加剧(苦痛等)

sharply ['ʃɑ:pli] adv.锐利地 敏锐地 严厉地 厉害地 轮廓清晰、线条分明地

shave [ʃeiv] vt.剃 修面 擦过 消减价格 vi.修面 刮胡子 勉强通过 n.修面 刮胡子

she [ʃi:] pron.(主格)她

shear [ʃiə] vt.剪 剪掉 剥夺 割 vi.剪 修剪 切割 剥夺 穿越 n.修剪 剪下的东西 羊的一岁

shed [ʃed] vt.散发 脱落 使...流出 除掉 n.车棚 小屋 贷棚 脱落物

sheep [ʃi:p] n.羊 绵羊

sheet [ʃi:t] n.被单 床单 纸张 薄板 张 片 vt.盖上被单 遍布 使成片状 vi.成片落下或流动 adj.片状的

shelf [ʃelf] n.搁板(状物) 岩棚 架子

shell [ʃel] n. 壳 贝壳 外形 炮弹 v. 去壳 脱落 炮击 拾贝壳 n. [计算机] DOS命令：安装备用的COMMAND.C文件 并改变环境尺寸

shelter ['ʃeltə] n.隐蔽处 掩蔽 庇护 庇护所 避难所 v.庇护 保护 隐匿

shepherd ['ʃepəd] n.牧羊人 羊倌 牧师 指导者 v.看守 领导 指导

shield ['ʃi:ld] n.盾 盾状物 防护物 vt.保护 遮蔽

shift [ʃift] vt.替换 转移 变声 vi.改变 定责 更衣 n.转换 交换 变化 移动 接班者

shilling ['ʃiliŋ] n.先令(货币单位)

shine [ʃain] vi.照耀 发光 表现出众 闪耀着 vt.使发光 照射在...上 擦亮 n.光 光泽 阳光 光辉 喜欢

ship [ʃip] n.(大)船 海船 舰 船员 飞船 vt.以船运送 乘船 运送 进水 在船上工作 vi.上船 坐船 运送 在船上工作

shirt [ʃə:t] n.(男式)衬衫

shiver ['ʃivə] vi.颤抖 哆嗦 vt.颤动(碎裂) n.冷颤(碎块)

shock [ʃɔk] n.冲击 震惊 冲突 震惊 休克 vi.震动 冲突 使...受电击

shoe [ʃu:] n.鞋

shoot [ʃu:t] vt.发射 射中 射击 发出 发芽 vi.飞驰 n.发芽 射击 芽 发射

shop [ʃɔp] n.商店 店铺 车间 vi.购物 寻找 vt.购物 逛商店

shopkeeper ['ʃɔp.ki:pə] n.店主 零售商人

shopping ['ʃɔpiŋ] n.买东西 购物 动词shop的现在分词形式

shore [ʃɔ:] n.滨 岸 国家 陆地 斜撑柱 vt.(up)支撑 支持

short [ʃɔ:t] adj.短的 短期的 矮的 短暂的 n.短裤 adv.短暂地 突然地

shortage ['ʃɔ:tidʒ] n.不足 缺少 不足额

shortcoming [ʃɔ:t.kʌmiŋ] n.短处 缺点

shortly ['ʃɔ:tli] adv.立刻 马上 简短地

shot [ʃɔt] n.发射 弹丸 射门 炮弹 射击 射手 努力 v.射击 发出 发芽(shoot的过去式和过去分词) adj.杂色的 破烂不堪

should [ʃud] aux.v.将 万一 就 应该 竟然 可能

shoulder ['ʃəuldə] n.肩 肩膀 挑起 角度 交接处

shout [ʃaut] n.呼喊 突发的大叫 vt.高声呼喊 呼叫 vi.呼叫 突然喊叫

show [ʃəu] vt.给...看 表明 表示 显现 展现 指示 n.显示 表现 表演 事情

shower ['ʃauə] n.阵雨 (一)阵 淋浴 vi.倾盆而下 淋浴 vt.淋湿 下骤雨 使(大量东西)落在某人身上

shriek [ʃri:k] vi.尖声喊叫 n.尖叫声

shrink [ʃriŋk] vi.收缩 缩小 退缩 萎缩 vt.使收缩 n.收缩 萎缩

shut [ʃʌt] vt.关上 闭上 关闭 防止进入 禁闭 合上 停止或中止 vi.关闭 停止或中止 n.关闭 adj.关闭的 合拢的 围绕的

shy [ʃai] adj.易受惊的 害羞的 胆怯的 迟疑的 vi.惊退 畏缩 乱投 vt.乱投 n.投掷 惊避

sick [sik] adj.有病的 恶心的 腻烦的 晕的 n.病人 呕吐物 vt.追逐 攻击 唆使

sickness ['siknis] n.疾病

side [said] n.边 旁边 一方 侧 方面 adj.旁的 侧的 vt.支持 同意 vi.支持

sideways ['saidweiz] adv.斜着 斜向一边地 向傍边 斜地里 向侧面地 adj.傍边的 向侧面的

sigh [sai] n.呼吸 叹息(带有情感) vi.发出叹息的声音 感叹 vt.叹息

sight [sait] n.视力 见 情景 景观 眼界 vt.看见 观看 vi.瞄准

sightseeing ['sait.si:iŋ] n.观光 游览 动词sightsee的现在分词

sign [sain] n.符号 招牌 正负号 手势 迹象 签名 v.签 做手势 做标记

signal ['signəl] n.信号 导火线 动机 vi.发信号 vt.发信号 标志 adj.显著的

signature ['signitʃə] n.署名 签字 签名 签署 书帖 有特征的符号

significance [sig'nifikəns] n.意义 意味 重要性

significant [sig'nifikənt] n.有意义的 重要的 意味深长的 相当数量的 重大的 n.(复)有意义的事物 标志

silence ['sailəns] n.沉默 寂静 vt.使沉默 使安静

silent ['sailənt] adj.沉默的 寂静无声的 安静的 不吵闹的 无言的 n.(复数)默剧

silk [silk] n.蚕丝 丝 绸 丝织品 adj.丝的

silly ['sili] adj.傻的 愚蠢的 淳朴的 晕的

silver ['silvə] n.银 银子 银器 adj.银色的 银的 vt.镀银 变成银白色

similar ['similə] adj.相似的 类似的 同样的

similarly ['similəli] adv.类似地 相似地

simple ['simpl] adj.简单的 朴素的 无辜的 天真的 基本的 纯粹的 n.平民 精神病人 草药

simplicity [sim'plisiti] n.简单 简易 朴素 单纯 简朴

simplify ['simplifai] vt.简化 使单纯

simply ['simpli] adv.简单地 朴素地 仅仅 完全地

sin [sin] n.罪 罪孽 过失 vi.犯罪 违反

since [sins] conj.从...以来 因为 既然 自从 adv.后来 prep.自从

sincere [sin'siə] adj.真诚的 真挚的

sing [siŋ] vt.唱 鸣唱 用诗赞美 叫 响 vi.唱歌 n.合唱

singer ['siŋə] n.歌唱家 歌手 鸣禽 烧毛工 烧毛机 Singer：辛格(姓氏)

single ['siŋgl] adj.单一的 独身的 单纯的 n.单程票 单曲 单人房 单身者 vt.单独挑出 vi.(棒球)作一垒打

singular ['siŋgjulə] adj.单一的 非凡的 个人的 单数的 独一的 唯一的 n.单数

sink [siŋk] vi.下沉 下垂 降低 下落 陷于 倾斜 恶化 渗透 vt.下沉 陷于 倾斜 减少 n.接收端 沟渠 污水槽 散热器

sir [sə:] n.先生 阁下 ...爵士

sister ['sistə] n.姐妹 姐 妹 护士长

sit [sit] vi.坐 就坐 孵 vt.使坐 担任 开会 孵 n.坐

site [sait] n.地点 地基 场所 位置 现场 vt.使...位于

situation [.sitju'eiʃən] n.位置 处境 形势 局面 状况 职位

six [siks] num.六 六个

sixteen ['siks'ti:n] num.十六 十六个

sixth [siksθ] num.第六 六分之一 n.六分之一 六度音 adj.第六的

sixty ['siksti] num.六十 六十个 n.六十 adj.六十的 六十个的

size [saiz] n.大小 体积 尺码 尺寸 vt.按大小排列(或分类) 用... 填充(某物) adj.一定大小的

skate [skeit] n.冰鞋 滑冰 vi.滑冰 溜冰 vt.滑冰

sketch [sketʃ] n.略图 速写 素描 概略 概述 草图 小品 vt.描绘略图 vi.写生

ski [ski:] n.滑橇 滑雪 滑雪板 vi.滑雪 adj.滑雪(用)的

skill ['skil] n.技能 技巧 熟练

skilled [skild] adj.有技能的 熟练的

skillful ['skilfəl] adj.灵巧的 娴熟的

skim [skim] vt. 掠过 擦过 滑过 略读 撇去浮沫 vi. 掠过 覆盖了一层浮沫 侵吞钱财

n. 一层浮沫 略读 脱脂物 adj. 撇去浮沫的 由脱脂乳做的

skin [skin] n.皮 皮肤 兽皮 外皮 皮囊 v.剥皮 削皮 adj.色情的

skirt [skə:t] n.女裙 边缘 郊区 vt.位于...边缘 装边于 回避 绕开 vi.形成边缘 绕...走 避开

sky [skai] n.天 天空 v. <英>把...投向空中 将...高挂

slam [slæm] vt.使劲关 砰地放下 猛然关上 砰地关上 猛烈抨击 vi.砰地关上 猛攻 猛烈抨击 n.砰然声 猛然 猛烈的抨击

slave [sleiv] n.奴隶 苦工 完全受控制的人 劳工 从动装置 vt.对另一种装置作出直接回应 vi.拼命工作 贩卖奴隶

slavery ['sleivəri] n.奴隶制度 苦役 奴役 奴隶身份

sleep [sli:p] n.睡眠 睡觉 vi. & vt.睡

sleepy ['sli:pi] adj.想睡的 寂静的

sleeve [sli:v] n.袖子 袖套 套管

slender ['slendə] adj.细长的 微薄的 苗条的 少量的

slice [slais] n.薄片 切片 薄的切片 部分 锅铲 vt.切成薄片 大幅降低

slide [slaid] vi.滑 跌落 vt.使滑动 n.滑 幻灯片 雪崩 山崩 滑坡 滑梯

slight [slait] adj.细长的 轻微的 纤细的 脆弱的 vt.轻视 忽略 n.轻蔑

slightly ['slaitli] adv.稍微 些微地 有点 苗条地

slip [slip] vi. 滑跤 滑落 溜 滑行 下跌 脱落 犯错 vt. 使滑动 偷偷 迅速塞(给) 匆忙穿或脱 摆脱 忽略 使脱臼 n. 滑 疏漏 小错误 下跌 船台 码头 衬裙 枕头套

slipper ['slipə] n.拖鞋 便鞋

slippery ['slipəri] adj.滑的 使人滑跤的 狡猾的 不可靠的

slit [slit] n.裂缝 投币口 vt.切开 纵切 撕开 扯裂

slogan ['sləugən] n.标语 口号

slope [sləup] n.倾斜 斜面 斜坡 斜线 斜率 vi.倾斜 有坡度 vt.使倾斜

slow [sləu] adj.慢的 迟钝的 adv. 慢慢地 v.放慢 减速

slowly ['sləuli] adv.慢慢地

slum [slʌm] n.贫民窟 贫民区 vi.访问贫民区

sly [slai] adj.狡猾的 躲躲闪闪的 诡密的 偷偷摸摸的

small [smɔ:l] adj.小的 少的 小型的 adv.些许地 n.狭小部分

smart [smɑ:t] adj. 巧妙的 洒脱的 聪明的 时髦的 漂亮的 敏捷的 轻快的 整洁的 vi. 刺痛 难过 烦恼

n. 刺痛 苦恼 adv. 轻快地 机灵地 整齐地 漂亮地

smell [smel] n.嗅觉 气味 味道 vt.嗅 闻 探出 发臭 vi.闻 发臭

smile [smail] vi.微笑 赞许 一笑而过 vt.微笑 微笑地表示 以微笑完成 n.微笑 笑容

smog [smɔg] n.烟雾

smoke [sməuk] n.烟 抽烟 无常的事物 vi.冒烟 吸烟 以烟熏

smooth [smu:ð] adj.平滑的 平静的 平稳的 流畅的 安祥的 圆滑的 搅拌均匀的 vt.使光滑 磨光 安慰 理顺(羽毛) vi.变光滑

smoothly [smu:ðli] adv.光滑地 平滑地 平稳地 流畅地

snake [sneik] n.蛇 v.蜿蜒 扭曲

snow [snəu] n.雪 下雪 似雪花的东西 粉状物 vi.下雪 vt.雪一般落下 使雪白 欺骗

snowstorm ['snəustɔ:m] n.雪暴 暴风雪

snowy [snəui] adj.雪的 下雪的 多雪的 被雪所覆盖着的

so [səu] adv.这样 这么 非常 如此 如是 同样 conj.所以 因此 adj.真的 pron.这样

soak [səuk] vt. 浸 泡 使...湿透 浸泡在液体中 浸洗 吸入 使某人喝醉 向某人敲竹杠 vi. 浸泡 浸入液体 渗透 感化(某人) 无节制地喝酒 n.浸 湿透 喝醉

soap [səup] n.肥皂 vt.用肥皂洗 阿谀奉承

sob [sɔb] vi.啜泣 呜咽 n.空间轨道的轰炸机 呜咽 哭泣

sober ['səubə] adj.清醒的 适度的 沉着冷静的 稳重的 颜色暗淡的 vt.使清醒 使沉着 vi.清醒 冷静下来

so-called ['səu'kɔ:ld] adj.所谓的 号称的

soccer ['sɔkə] n.英式足球

social ['səuʃəl] adj.社会的 社交的 群居的 社团的

socialism ['səuʃəlizəm] n.社会主义

socialist ['səuʃəlist] adj.社会主义的 n.社会主义者

society [sə'saiəti] n.社会 团体 社交界 阶层圈子

sock [sɔk] n.短袜 vt. 给...穿短袜 v.狠揍 痛打 n.重击 犁头 犁刃 轻软鞋 喜剧

soda ['səudə] n.碳酸钠 纯碱 汽水 苏打

soft [sɔft] adj.软的 柔软的 柔和的 温和的 不强烈的 轻的 adv.柔软地 温和地 n.柔软的东西

softly ['sɔftli] adv.柔软地 温柔地 柔和地 静静地

soil [sɔil] n.泥土 土壤 土地 污物 粪便 vt.弄脏 弄污 vi.变脏 vt.用青草饲料喂养

solar ['səulə] adj.太阳的 太阳能的 日光的

soldier ['səuldʒə] n.(陆军)士兵 军人 偷懒者 [动]兵蚁 vi.当兵

sole [səul] n.脚底 鞋底 袜底 sole或soles鳎鱼 adj.唯一的 单独的 独占的 vt.给(鞋)换底

solely ['səulli] adv.单独地 独自地 唯一地

solemn ['sɔləm] adj.庄严的 严肃的 隆重的

solid ['sɔlid] adj.固体的 结实的 实心的 可靠的 n.固体 立方体 adv.一致地

soluble ['sɔljubl] adj.可溶的 可以解决的

solution [sə'lu:ʃən] n.解决 解答 解决办法 溶解 溶液

solve [sɔlv] vt.解答 解释 解决

some [sʌm, səm, sm] adj.一些 若干 adv.大约 在某种程度上 pron.一些(人 物) 若干

somebody ['sʌmbədi] pron.某人 有人 n.重要人物

somehow ['sʌmhau] adv.由于某种原因 不知怎么地 以某种方式

someone ['sʌmwʌn] pron.某人 有人

something ['sʌmθiŋ] pron.某事 某物 有价值或重要的人或物 adv.稍微 有点 非常

sometime ['sʌmtaim] adv.在某一时候 从前 改天 来日 adj.以前的 某一时间

sometimes ['sʌmtaimz] adv.不时 有时 间或

somewhat ['sʌmwɔt] pron.一点儿 一些 某物 adv.有点 多少 几分

somewhere['sʌmwɛə] ad.在某处 到某种程度 大约 差不多 n.某地 某处

son [sʌn] n.儿子 孩子(长者对后辈的称呼)

song [sɔŋ] n.歌唱 歌曲 歌词 (鸟等)鸣声

soon [su:n] adv.不久 很快 早 快

sophisticated [sə'fistikeitid] adj.老于世故的 高级的 诡辩的 精密的 老练的 尖端的

sore [sɔ:] adj.痛的 痛心的 恼火的 n.疮 痛处 adv.疼痛地

sorrow ['sɔrəu] n.悲痛 悲哀 悲伤 伤心事 vi.悲伤 惋惜 悔恨

sorry ['sɔri] adj.对不起的 抱歉的 难过的 遗憾的 int.对不起

sort [sɔ:t] n.种类 类别 特征 方法 vt.分类 整理 使明确 vi.交往 协调

soul [səul] n.灵魂 精神 人 心灵 精髓 鬼魂 人 黑人灵歌 adj.美国黑人文化的 黑人的

sound [saund] adj. 健康的 健全的 完好的 可靠的 合理的 adv. 酣畅地 n. 声音 吵闹 海峡 鱼鳔 探针 v. 听起来 发出声音 探测深度 探询 vi. 响 发声

soup [su:p] n.汤 羹 n.困境

sour ['sauə] adj.酸的 酸腐的 刻薄的 脾气坏的 v.变酸 n.酸物

source [sɔ:s] n.河的源头 根源 发源地 来源 原始资料

south [sauθ] n.南 南方 adj.南方的 adv.在南方 向南方 自南方

southeast [.sauθ'i:st] n.东南 东南地区 adj.位于东南的 东南的 向东南的 来自东南的 adv.往东南 来自东南 在东南

southern ['sʌðən] adj.南方的 南部的 来自南方的

southwest [.sauθ'west] n.西南 西南风 西南地区 adj.西南的 来自西南方的 adv.往西南 来自西南

Soviet ['səʊviət, 'sɔviət] n.苏维埃 adj.苏维埃的 苏联的

sow [sau] vt.播(种) 散布 vi.播种 散布 n.母猪, 大母熊, 铁水沟 铁锭

space [speis] n.空间 场地 空地 空白 太空 间隔 v.留间隔 分隔

spacecraft ['speiskrɑ:ft] n.航天器 宇宙飞船

spaceship ['speisʃip] n.航天飞船

spade [speid] n.铲 铁锹 在扑克牌中指黑桃 v.铲 vt.(用铲子)铲

span [spæn] n.跨距 跨度 间距 一段时间 vt.横跨 贯穿 估量 动词spin的过去式形式 n.共轭动物

Spanish ['spæniʃ] adj.西班牙的 西班牙人的 西班牙语的 n.西班牙人 西班牙语

spare [spɛə] vt.节约 抽出 饶恕 赦免 adj.多余的 闲置的 备用的 简陋的 n.剩余 备用品

spark [spɑ:k] n.火花 火星 朝气 情人 俗丽的年轻人 vi.闪烁 冒火花 热情回应 vt.激发 引起 追求

sparkle ['spɑ:kl] vi.发火花 闪耀 冒火花 vt.使闪耀 n.闪耀 火花

sparrow ['spærəu] n.麻雀

speak [spi:k] vi.说话 发言 vt.说 讲 演说

speaker ['spi:kə] n.说话者 发言者 说某种语言者 扬声器

spear [spiə] n.矛(正负电子对撞机) 枪 vt用矛刺

special ['speʃəl] adj.特殊的 特别的 专门的 n.特别的东西 特价 特刊 特色菜

specialist ['speʃəlist] n.专家

speciality [.speʃi'æliti] n.专业 特长 擅长 特产

specialize ['speʃəlaiz] vi.成为...专家 专攻 特殊化 vt.专门研究 专攻 使...特殊化

specially ['speʃəli] adv.专门地 特别地

specific [spi'sifik] adj.特有的 特殊的 明确的 具体的 有特效的 n.特效药 特性 详情

specify ['spesifai] vi.指定 详细说明 阐述

specimen ['spesimən] n.样本 标本 样品

spectacle ['spektəkl] n.光景 景象 眼镜 值得看的东西

speech [spi:tʃ] n.言语 演说 演讲 语言能力 方言

speed [spi:d] n.快 速率 速度 迅速 vi.快行 加速 急行

spell [spel] vt.拼写 缓慢地或吃力地读 轮班 休息 用符咒镇住(某人) vi.拼字 n.魅力 符咒 一段时间 轮班

spelling ['speliŋ] n.拼字 拼法 拼写法

spend [spend] vt.用钱 花费 浪费 度过

sphere [sfiə] n.球 圆体 范围 领域 天体 vt.形成球体 包围 围绕

spider ['spaidə] n.蜘蛛

spill [spil] vt.使溢出 洒 使...流出 泄漏 vi.溢出 n.溢出 流 n.木片 小杆

spin [spin] vt.使旋转 疾驰 纺织 结网 眩晕 n.旋转 纺织 眩晕 疾驰

spirit ['spirit] n.精神 气魄 情绪 心灵 幽灵 精灵 (复)spirits: 精力 情绪 烈酒 vt.使精神振作 偷偷带走

spiritual ['spiritjuəl] adj.精神的 心灵的 与上帝有关的 n.(尤指美国南部黑人的)圣歌

spit [spit] vi.吐 吐唾沫 吐出 降小雨 vt.用炙叉叉住 n.唾液 口水 小雨 烤肉叉 海峡 海角

splash [splæʃ] n.飞溅的水 污点 卖弄 vt.溅湿 溅开 泼 飞溅

splendid ['splendid] adj.壮丽的 显著的 极好的 辉煌的 壮观的 杰出的

split [split] vt.劈开 分离 分开 vi.被劈开 adj.分散的 n.劈开 裂片 裂口

spoil [spɔil] vt.损坏 破坏 糟蹋 宠坏 溺爱 腐坏 n.战利品 奖品

sponge [spʌndʒ] n.海绵 海绵状的东西 vt.用海绵擦拭 吸收掉 抹掉

sponsor ['spɔnsə] n.发起者 保证人 赞助者 主办者 vt.发起 赞助 倡议

spontaneous [spɔn'teiniəs] adj.自发的 自然产生的 本能的

spoon [spu:n] n.匙 调羹 匙状物 vt.以匙舀起 vi.调情

sport [spɔ:t] n.运动 体育 运动会 vi.炫耀 戏弄 嬉戏 adj.运动的 户外穿戴的

sportsman ['spɔ:tsmən] n.运动员

spot [spɔt] n. 点 斑点 污点 地点 场所 斑点 职位 聚光灯 插播节目 少量 adj. 当场的 现场买卖的 抽样的 插播的 v. 使沾上污点 认出 用灯光照射

spray [sprei] n.浪花 喷雾 水沫 喷雾器 vt.喷 喷雾 扫射 喷射

spread [spred] vt.伸开 伸展 展开 传播 散布 铺开 涂撒 n.传播 伸展 铺开 幅度 范围 盛宴 差额

spring [spriŋ] vi. 跳 跃 射出 涌入(出) 发芽 长高 vt. 弹起 使喷涌 越过 弹出 释放 发生 跳 裂开 n. 春天 春季 跳跃 弹跳 泉 泉水 弹簧 投射

springtime ['spriŋtaim] n.春季 春天 青春期 初期

sprinkle ['spriŋkl] n.洒 撒 散置 小雨

spur [spə:] n.刺激 刺激物 马刺 鞭策 支脉 vt.刺激 鞭策 促进

spy [spai] n.间谍 特务 侦探 侦察 vt.侦察 侦探 看到 找出 vi.看 侦察

square [skwɛə] n.正方形 广场 街区 平方 adj.正方形的 正直的 公正的 结实的 断然的 v.使...成方形 调整 一致 收买 adv.成直角 直接的 正方形地

squeeze [skwi:z] vt. & vi.榨 挤 压榨 挤压 塞进 n.压榨 勒索 榨取

squirrel ['skwirəl] n.松鼠 松鼠皮毛 vt.储存

stab [stæb] vt. & vi.刺 戳 刺伤 n.刺 戳 刺痛 尝试 突然的一阵

stability [stə'biliti] n.稳定 稳定性 巩固 居于修道院

stable ['steibl] adj.稳定的 不变的 安定的 可靠的 n.厩 马厩 马棚 牛棚 一批人 vt.使(马)入厩 vi.入厩

stack [stæk] n.堆 堆叠 垛 vt.堆积 堆放

stadium ['steidiəm] n.露天大型运动场 竞技场

staff [stɑ:f] n.工作人员 参谋 全体人员 同事 拐杖 杆 棒 五线谱 vt.配备员工

stage [steidʒ] n.舞台 驿站 戏剧 阶段 vt.上演 实行 进行

stain [stein] vt. 玷污 弄脏 给...着色 n.污点 瑕疵 染料 着色剂

stair [stɛə] n.楼梯

staircase ['stɛəkeis] n.楼梯 楼梯间

stake [steik] n.桩 赌金 赌注 利害关系 奖品 v.下注 用桩支撑

stale [steil] n.陈腐的 走了气的 (动物的)尿 adj.不新鲜的 陈腐的 失效的 vt.使腐坏 vi.变得不新鲜 腐坏 (动物)撒尿

stamp [stæmp] n.戳子 邮票 标志 图章 印 跺脚 v.跺脚 盖章

stand [stænd] vi.站 坐落 位于 停滞 忍受 n.架 台 站立 停顿 楼台 货摊

standard ['stændəd] n.标准 规格 adj.标准的 规范的 普通的

standpoint ['stændpɔint] n.立场 观点

star [stɑ:] n.星 恒星 明星 v.以星状物装饰 变成演员

stare [steə(r)] vi.盯 凝视 vt.显眼 (毛发)竖立 n.凝视

start [stɑ:t] vi.跳起 出发 开始 起身 vt.开始 使震惊 n.起点 震动 领先优势 起推动作用的外力

startle ['stɑ:tl] vt.使大吃一惊 使...惊愕 n.吃惊 惊愕

starve [stɑ:v] vi.饿死 挨饿 受饿 vt.使饿死 使挨饿

state [steit] n.状态 国家 州 情形 adj.国家的 州的 正式的 vt.说 说明 阐明 陈述 声明 规定

statement ['steitmənt] n.陈述 声明

statesman ['steitsmən] n.政治家 国务活动家

static ['stætik] adj.静的 静态的 静力的 静止的 静电的 n.静电 静电干扰 (猛烈的)反对或抨击

station ['steiʃən] n.站 台 所 局 车站 位置 地位 vt.安置 驻扎 配置

statistical [stə'tistikəl] adj.统计的 统计学的

statue ['stætju:] n.塑像 雕像 铸像

status ['steitəs] n.地位 身分 情形 状况

stay [stei] vi.停留 逗留 住 n.逗留 停留 支撑物 倚赖的人或事

steadily ['stedili] adv.稳定地 不变地 稳固地 坚定地

steady ['stedi] adj.稳固的 稳定的 坚定的 vt.使稳定 使稳固 (变得)稳固 adv.平稳地 稳固地 n.固定的事物

steal [sti:l] vt.偷 窍取 n.偷窃 便宜货 (欺诈的)政治交易

steam [sti:m] n.蒸汽 水蒸气 vi.蒸发 蒸 冒热气 vt.蒸

steamer ['sti:mə] n.轮船 汽船 蒸笼 [动]沙海螂

steel [sti:l] n.钢 钢铁 钢制品 vt.使...硬如钢 使坚强 使下定决心 adj.钢的

steep [sti:p] adj.险峻的 陡峭的 (价格)过高的 n.陡坡 n.浸泡 浸泡液 v.浸泡 沉浸

steer [stiə] vt. & vi.驾驶 掌舵 引导 操纵 n.(食用的)公牛 暗示

stem [stem] n.茎 (树)干 柄 船首 vi.起源 vt.抽去...的梗 给...装柄 逆...而行 n.血统 [语]词干 v.堵住 阻止 [体]内转

step [step] n.(脚)步 步骤 步伐 踏级 vi.走 vt.踏 以步测量

stern [stə:n] adj.严厉的 坚定的 坚决的 苛刻的 n.尾部 船尾

steward ['stju:əd] n.乘务员 服务员 v.做乘务员 管理

stewardess ['stju:ədis] n.空中小姐 女乘务员 女管家

stick [stik] n.棍 棒 枝 杆 手杖 vt.刺 粘贴 插于 刺入 竖起 vi.钉住 粘贴 坚持

sticky ['stiki] adj.粘性的 胶粘的 闷热的 困难的 令人不满意的

stiff [stif] adj. 硬的 僵直的 生硬的 拘谨的 不灵活的 adv. 僵硬地 极其 n. 死尸 过于拘谨的人 流浪汉 吝啬鬼 醉鬼vt. 诈骗 失信 不给小费

stiffen ['stifn] vt.使硬 使僵硬 变为浓粘

still [stil] adj. 静止的 不动的 静寂的 n. 寂静 沉默 定格照 酒厂 adv. 仍然 更 静止地 conj. 然而 但是

vi. 变沉默 vt. 使冷静 抓住 使停止 使放松

stimulate ['stimjuleit] vt.刺激 激励 激发 鼓舞 vi.起刺激作用

sting [stiŋ] vt.刺 刺痛 使苦恼 欺诈 vi.刺 叮 刺痛 n.刺痛刺 讽刺

stir [stə:] vt.动 拨动 激动 惹起 搅和 vi.挪动 活动 离开 搅拌 激动 n.感动(激动 愤怒或震动), 搅拌 骚乱

stitch [stitʃ] n.一针 针脚 疼痛 碎布条 vt.缝 缝合

stock [stɔk] n. 原料 库存品 存货 股本 股份 储备 树干 血统 家畜 adj. 存货的 平凡的 惯用的 股票的 畜牧的 vt. 进货 采购 储存 供给 vi. 出新芽 进货

stocking ['stɔkiŋ] n.长(统)袜

stomach ['stʌmək] n.胃 胃口 肚子 食欲 vt.忍受 容忍

stone [stəun] n. 石 石头 石块 石料 (水果的)硬核 宝石 英石(重量单位 合14磅) vt. 向...扔石头 去掉(水果的)核 adj. 石的 石制的 adv. (用于复合词) 全然地

stony ['stəuni] adj.多石的 石头的 石质的 冷酷的 铁石心肠的 <古>石造的

stool [stu:l] n.凳子 搁脚凳 便桶 根株 vi.引诱

stoop [stu:p] vi.俯身 弯下 弯下上身 屈服 n.弯腰 佝偻 屈服

stop [stɔp] vt.塞住 阻止 停止 停留 n.停止 车站 逗留

storage ['stɔridʒ] n.贮藏 贮藏量 存储 保管 保管费 仓库 [计]存储器

store [stɔ:] n.贮藏 贮存品 商店 仓库 vt.储存 贮藏 供给 保存 (在计算机里)存贮 记忆 adj.储备的

storey ['stɔ:ri] n.(层)楼 =story(美)

storm [stɔ:m] n.风暴 暴(风)雨 v.猛攻 捣毁 起风暴 暴怒

stormy ['stɔ:mi] adj.有暴风雨的 激烈的 粗暴的

story ['stɔ:ri] n.故事 小说 传说 叙述 新闻报道 楼层(=storey) vt.以历史事件为图案装饰

stove [stəuv] n.炉子 火炉 电炉 火炉窑 烘房 【主英】温室

straight [streit] adj. 直的 正直的 准的 坦率的 连续的 异性恋的 adv. 直 直接 笔直地 平正了 坦率的 连续不断地 n. 直的东西 传统的人

strain [strein] vt.拉紧 劳累 过份使用 vi.尽力 n.紧张 拉紧 血统

strange [streindʒ] adj.陌生的 奇怪的 adv.奇怪地

stranger ['streindʒə] n.陌生人 新来者 外地人 新手

strap [stræp] n.带子 皮带 vt.(用带)捆扎 (用带)打

strategy ['strætidʒi] n.战略 策略

straw [strɔ:] n.稻草 麦杆 吸管 adj.用稻草做的 用来作稻草的 麦杆色的 无价值的 象稻草人的 非正式民意测验所发现的

strawberry ['strɔ:bəri] n.草莓

stream [stri:m] n.河 (人 车 气)流 水流 组 vi. & vt.流 流出 飘动 接踵而来

street [stri:t] n.街 街道 行车道 adj.街道的

strength [streŋθ] n.力 力量 力气 长处 强度

strengthen ['streŋθən] vt.加强 变坚固 巩固

stress [stres] n.压力 紧张 重音 负载 vt.着重 强调 vi.强调 着重

stretch [stretʃ] vt.伸展 张开 vi.伸 延伸 n.伸展 张开 adj.可伸缩的

strict [strikt] adj.严格的 严谨的 精确的 完全的

strictly ['striktli] adv.严格地 严谨地

strike [straik] vt. & vi.打 击 撞 罢工 划燃 n.罢工 打击 殴打

string [striŋ] n.线 细绳 一串 字串 vt.串起 成串 收紧 悬挂 系 扎 vi.给...装弦 adj.线的

strip [strip] vt.剥夺 剥去 夺去 脱衣 n.条带 长条 条状 脱衣舞

stripe [straip] n.条纹 条子 条纹布 军士军阶 种类 鞭打 抽打 vt.使...带有条纹

stroke [strəuk] n.打 击 鸣声 中风 笔画 连续的动作 心跳 抚 摩 捋 vt.奉承 轻抚 摩 捋 vi.击打 心跳...

strong [strɔŋ] adj.强壮的 强烈的 坚强的 浓的 擅长的

strongly ['strɔŋli] adv.强壮地 强烈地 强有力地 坚固地 坚强地 激烈地

structural ['strʌktʃərəl] adj.结构的 构造的 建筑的 建筑用的

structure ['strʌktʃə] n.结构 构造 建筑物 vt.建造 构造

struggle ['strʌgl] n. & vi.斗争 挣扎 竞争 奋斗

student ['stju:dənt] n.学生 研究生 学者

study ['stʌdi] n.学习 研究 学科 沉思 书房 vt.学 读书 计划 研究 记忆 调查 vi.学习 研究 沉思

stuff [stʌf] n.材料 原料 东西 素质 本质 废物 vt.装 填 塞 填满 塞满

stumble ['stʌmbl] vi.绊倒 犯错 失策 踌躇 无意中发现 vt.绊倒 困惑 n.绊倒 失策

stupid ['stju:pid] adj.愚蠢的 感觉迟钝的 笨拙的 n.傻瓜

style [stail] n.风格 文体 式样 时尚 类型 vt.称呼 设计

subject ['sʌbdʒikt,səb'dʒekt] n. 题目 学科 [语]主语 主题 话题 题材 主观 (君主国)国民 实验对象

adj. 服从的 易患的 vt. 使隶属 使服从 使遭遇

submarine ['sʌbməri:n] adj.水下的 海底的 n.潜水艇 vt.用潜艇袭击 vi.滑入...底下

submerge [səb'mə:dʒ] vt.使浸没 潜入水中 覆盖 vi.潜入水中 浸没

submit [səb'mit] vt.使服从 屈服 呈送 递交 主张 vi.服从 屈从 忍受

subsequent ['sʌbsikwənt] adj.随后的 后来的

substance ['sʌbstəns] n.物质 实质 本旨 内容 重要性 财产

substantial [səb'stænʃəl] adj.物质的 坚固的 实质的 可观的 大量的 n.重要部份 本质

substitute ['sʌbstitju:t] n.代替人 代用品 vt.用...代替 代以 代替

subtract [səb'trækt] vt.减 减去 去掉 扣掉 vi.减少

suburb ['sʌbə:b] n.郊区 郊外 近郊

subway ['sʌbwei] n.地道 地下铁路

succeed [sək'si:d] vt.继...之后 vi.成功 成功 继承 接着发生

success [sək'ses] n.成功 成就 胜利 发财 成功者 成功的事物

successful [sək'sesful] adj.成功的 结果良好的 圆满的

successfully [sək'sesfəli] advS.成功地

succession [sək'seʃən] n.连续 继任 继位 继承 继承权

successive [sək'sesiv] adj.连续的 接连的 接二连三的

such [sʌtʃ, sətʃ] adj.这样的 如此的 pron.这样的人或事 adv.如此地

suck [sʌk] vt.吸 吮 啜 吸收 吸入 vi.吸 吸收 奉承 <俚>烂 令人讨厌 n.吸吮 吸力

sudden ['sʌdn] adj.突然的 意外的 快速的 出乎意料的 n.突发之事

suddenly ['sʌdnli] adv.突然地

suffer ['sʌfə] vt.遭受 忍受 容许 允许 vi.受苦 患病 受损失

sufficient [sə'fiʃənt] adj.足够的 充分的

sufficiently [sə'fiʃəntli] adv.足够地 充分地

sugar ['ʃugə] n.糖 vt.使甜 使具有吸引力 vi.形成糖状 变成颗粒状 造枫糖浆

suggest [sə'dʒest] vt.建议 暗示 启发 要求

suggestion [sə'dʒestʃən] n.建议 意见 暗示 迹象 联想

suit [sju:t] n.起拆 诉讼 请求 套装 vt.适合 取悦 满足

suitable ['sju:təbl] adj.合适的 适宜的 adv.<古>合适地 恰当地

sulphur ['sʌlfə] n.硫(磺) 硫黄

sum [sʌm] n.总数 金额 顶点 概略 一笔钱 vi.共计 总计 概括

summarize ['sʌməraiz] vt.概括 概述 总结 摘要而言

summary ['sʌməri] n.摘要 概要 一览 adj.概要的 简略的

summer ['sʌmə] n.夏 夏季 adj.夏季的 v.度过夏天

sun [sʌn] n.太阳 日 阳光 中心人物 像太阳般的东西 v.晒太阳

Sunday ['sʌndi] n.星期日 礼拜日 adj.星期日的

sunlight ['sʌnlait] n.日光 阳光

sunny ['sʌni] adj.阳光充足的 睛朗的 明媚的 快活的

sunrise ['sʌnraiz] n.日出(时分)

sunset ['sʌnset] n.日落(时分) 衰落时期(尤指人的晚年)

sunshine ['sʌnʃain] n.(直射)日光 阳光

super ['sju:pə] adj.极好的 超级的 n.主管人 负责人 警务长 adv.特别 格外

superficial [.su:pə'fiʃəl] adj.表面的 肤浅的

superior [su:'piəriə] adj.较高的 优越的 上层的 上好的 出众的 高傲的 n.上级 高手 上标

supermarket ['sju:pə.mɑ:kit] n.超级市场

supper ['sʌpə] n.晚饭 晚餐 晚餐会

supplement ['sʌplimənt,'sʌpliment] n.补充物 补充 增刊 增补 vt.补充 增补

supply [sə'plai] n.补给 供给 供应 贮备 vt.补给 供给 提供 提供足够以满足

support [sə'pɔ:t] vt.支撑 支持 供养 帮助 n. 支持 援助 供养

suppose [sə'pəuz] vt.猜想 假定 让 以为 认为 应该 获准(be suppose to) 让(虚拟语气) vi.推测

supreme [sju:'pri:m] adj.最高的 最大的 至上的 极度的

sure [ʃuə] adj.确信的 确实的 肯定的 adv.当然 确实地 无疑地

surely ['ʃuəli] adv.确实 稳当地 安全地

surface ['sə:fis] n.地面 表面 外表 平面 adj.表面的 肤浅的 vt.在...上加表层 使...成平面 vi.浮到水面 披露出来 在表面工作

surgeon ['sə:dʒən] n.外科医师 军医

surgery ['sə:dʒəri] n.外科 外科手术 诊所

surname ['sə:neim] n.姓 vt.冠姓

surprise [sə'praiz] vt.使惊奇 使诧异 奇袭 突然袭击 n.惊奇 诧异 惊讶 突然袭击

surprising [sə'praiziŋ] adj.惊人的 出人意外的 令人惊讶的 动词surprise的现在分词

surprisingly [sə'praiziŋli] adv.惊人地(意外地)

surrender [sə'rendə] v.交出 让与 投降 屈服 n.投降 屈服 放弃

surround [sə'raund] vt.围 围绕 圈住 包围 环绕 n.环绕物

surroundings [sə'raundiŋz] n.周围的事物 环境 名词surrounding的复数形式

survey [sə:'vei] vt.俯瞰 检查 调查 测量 勘定 纵览 环视 n.调查 纵览 视察 测量

survive [sə'vaiv] vt.幸免于 比...活得长 艰难度过 vi.活下来 活着 继续存在

suspect [səs'pekt] vt.怀疑 猜想 vi.疑心 猜想 n.嫌疑犯 adj.令人怀疑的 不可信的

suspend [səs'pend] vt.吊 推迟 悬挂 暂停 取消 vi.悬挂 中止 停止偿付债务

suspicion [səs'piʃən] n.怀疑 疑心 猜疑

sustain [səs'tein] vt.支撑 支持 供养 忍受 承受 经受 维持 认可

swallow ['swɔləu] n.燕子 吞咽 一次吞咽的量 vt.吞下 咽下 忍受 不流露感情 收回 vi.咽口水 忍气吞声 吞下 咽下

swamp [swɔmp] n.沼泽 沼泽地 湿地 v.淹没 陷于沼泽

swan [swɔn] n.天鹅 vi.闲荡 游荡

swarm [swɔ:m] n.一大群 蜂窝 vi.密集 云集 充满

sway [swei] vt. & vi.摇摆 摇动 支配 影响 n.摇摆 动摇 支配

swear [swɛə] vt.宣(誓) 发誓 vi.诅咒 n.宣誓 誓言

sweat [swet] n. 汗 汗水 (凝结在物体表面的)水珠 焦急 繁重工作 vi. 出汗 渗出 焦躁 辛苦工作

vt. (使)出汗 使渗出 榨取 逼问 使焦虑

sweater ['swetə] n.厚运动衫 毛线衫 毛衣 出汗者

sweep [swi:p] vt. 扫 扫过 扫除 拂去 刮起 掠过 席卷 扫视 全胜 vi. 打扫 席卷 昂首阔步地走 连绵 全胜

n. 扫除 席卷 搜寻 挥动 范围 风帆 (竞赛中)全胜 连绵一片 赌金

sweet [swi:t] adj.甜的 愉快的 可爱的 n. 糖 甜食 adv.甜蜜地

swell [swel] vi. 膨胀 隆起 增长 vt. (使)膨胀 (使)鼓起 (使)增长 n. 膨胀 肿胀 增大 衣着时髦的人 风头人物 adj. <口> 很棒的 一流的 时髦的

swift [swift] adj.快的 反应快的 迅速的 n.雨燕 线轴 n.斯威夫特(人名) adv.快速地

swim [swim] vi.游 游泳 眼花 n.游泳

swing [swiŋ] vi.摇摆 回转 旋转 动摇 n.摇摆 改变 冲力

Swiss [swis] adj.瑞士的 n.瑞士人

switch [switʃ] n.开关 转换 鞭子 vt.转换 改变 交换 鞭打 n.(女用)假发

sword [sɔ:d] n.剑 刀

symbol ['simbəl] n.象征 符号 记号 标志

sympathetic [.simpə'θetik] adj.同情的 和谐的 共鸣的 n.交感神经

sympathize ['simpəθaiz] vi.同情 同感 共鸣

sympathy ['simpəθi] n.同情 同情心 一致 同感 赞同 慰问

synthetic [sin'θetik] adj.综合的 合成的 人造的 n.人工制品

system ['sistəm] n.系统 体系 制度

systematic(al) [.sisti'mætik(l)] adj.有系统的 有条理的 有计划的 分类的 体系的

## T

table ['teibl] n.桌子 台子 餐桌 表格 项目表 vt.将...放于桌上 提交讨论 搁置 列入表中 adj.桌子的

tablet ['tæblit] n.碑 匾 片状物 药片

tag [tæg] n.附加语 附属物 名称 谚语 标签 vt.附以签条 尾随 添饰 指责 连接 vi.紧随 尾随

tail [teil] n. 尾巴 尾部 后部 辫子 跟随者 (复)tails：燕尾服 (硬币的)反面 vt. 跟踪 盯...的梢 为...装尾巴

vi. 尾随 附于其后 逐渐减少 n. 限嗣继承 adj. 限嗣继承的

tailor ['teilə] n.裁缝 vt.裁制衣服 剪裁 vi.做裁缝

take [teik] vt.拿 抓 吃 拿走 取 修(课程) 花费 记录 认为 搭乘 n.拿取 取得物

tale [teil] n.故事 传说 叙述 谎话 总计

talent ['tælənt] n.天才 才能 人才 天资

talk [tɔ:k] vi.讲话 说 vt.谈论 商讨 n.谈话 会谈 讲话 讨论

tall [tɔ:l] adj.高的 身材高的 难以置信的 难以办成的 夸大的 adv.夸大地

tame [teim] adj.驯服的 顺从的 柔顺的 乏味的 vt.驯养 使...驯服 vi.变得驯服

tan [tæn] n.棕褐色 黝黑 adj.棕黄色的 棕褐色的 v.晒黑 鞣(革) 使晒成棕褐色

tank [tæŋk] n.坦克 大容器 槽 箱 罐 贮水池 vt.把...储于槽中

tap [tæp] n.龙头 塞子 窃听 轻拍 轻敲 鞋掌 vi.轻拍 轻击 补鞋掌 选择 vt.装龙头 窃听 利用 开发

tape [teip, tep] n.线带 磁带 录音带 磁带 胶布 v.(用带子)捆 录音 adj.录音的, 用来录音的

target ['tɑ:git] n.靶 标的 目标 对象 vt.把...作为目标 瞄准

task [tɑ:sk] n.任务 工作 作业 vt.交给某人(任务)

taste [teist] vt.尝 尝到 品尝 尝试 有...味道的 n.味觉 味道 品味 尝试

tax [tæks] n.税 税款 负担 重负 vt.课税 使...负重荷 斥责

taxi ['tæksi] n.出租汽车 v.用出租车运送 乘计程车 (飞机)滑行

tea [ti:] n.茶 茶叶 茶树 茶点

teach [ti:tʃ] vt.讲 教育 教 教书 教导 教训 vi.给予指导

teacher ['ti:tʃə] n.教师 老师 先生

teaching ['ti:tʃiŋ] n.教学 教义 讲授 教导 adj.教学的 动词teach的现在分词

team [ti:m] n.队 组 vt.协同工作 vi.结成一队 adj.团队的

tear [tiə] n.眼泪 泪珠 (撕破的)洞或裂缝 撕扯 vt.撕掉 撕开 撕裂 扯下 扰乱 vi.流泪 撕破

technical ['teknikəl] adj.技术的 工艺的

technician [tek'niʃən] n.技术员 技师

technique [tek'ni:k] n.技术 技巧 技能

technology [tek'nɔlədʒi] n.工艺学 工艺 技术

tedious ['ti:diəs] adj.冗长乏味的 单调乏味的 沉闷的

teenager ['ti:n.eidʒə] n.青少年 13岁到19岁的年轻人

telegram ['teligræm] n.电报 vt.向...发电报

telegraph ['teligrɑ:f] n.电报机 电报 vt.打电报 显示

telephone ['telifəun] n.电话 vi.打电话

telescope ['teliskəup] n.望远镜 v.缩短 压缩

television ['teliviʒən] n.电视 电视机 电视节目 电视行业

tell [tel] vt.讲述 吩咐 辨别 告诉 vi.讲述 告诉 产生效果 告发

temper ['tempə] n.调剂 韧度 回火度 心情 情绪 脾气 性情 vt.使缓和 调和 vt.使回火

temperature ['tempritʃə(r)] n.温度 体温 气温 发烧

temple ['templ] n.圣堂 神殿 庙宇 寺院

temporary ['tempərəri] adj.暂时的 n.临时工

tempt [tempt] vt.引诱 诱惑 吸引 勾引

temptation [temp'teiʃən] n.诱惑 引诱

ten [ten] num.十 十个

tenant ['tenənt] n.承担人 房客 佃户 占用者 vt.租借 租赁

tend [tend] vt.照管 照料 护理 趋向 易于 走向 倾向

tendency ['tendənsi] n.趋向 趋势 倾向

tender ['tendə] adj. 嫩的 脆弱的 温柔的 亲切的 敏感的 未成熟的 痛的 (船)易倾斜的 v. 提供 投标 (使)变柔软 n. 提出 投标 支付 偿还 照管人 补给船 煤水车

tennis ['tenis] n.网球 网球运动

tense [tens] n.(动词)时态 时 adj.拉紧的 绷紧的 紧张的 v.使紧绷 拉紧

tent [tent] n.帐篷 v.住帐篷 宿营

tenth [tenθ] num.第十 十分之一 n.第十 十分之一

term [tə:m] n.期 学期 条件 词 术语 名词 期限 条款 vt.把...称为

terminal ['tə:minl] adj.末端的 终点的 晚期的 (每)学期的 n.末端 终端机 终点 终点站

terrible ['terəbl] adj.可怕的 极度的 糟糕的

terrific [tə'rifik] adj.可怕的 极大的 极好的 非常的

territory ['teritəri] n.领土 版图 领域 范围

terror ['terə] n.恐怖 惊骇 <口>令人惧怕或讨厌的人或事物

test [test] n.测试 试验 考验 vt.测试 试验 接受测验 adj.测试的

text [tekst] n.原文 本文 课文 正文 v.发短信

textbook ['tekstbuk] n.课本 教科书

textile ['tekstail] n.纺织品 adj.纺织的

than [ðæn] conj.(比较级)比 prep.比(用于两者之间的比较)

thank [θæŋk] vt.谢谢 感谢 n.感谢 谢意

that [ðæt] adj.如此的 上文提到的 特定的 pron.那 adv.那样 非常 conj.引导从句

the [ði:, ði] art.(定冠词)这 那 这些 那些 adv.(用于比较级前)更加 (用于形容词、副词比较级前)越...越...

theatre ['θiətə] n.戏院 戏剧 教室 =theater

their [ðɛə] pron.他(她、它)们的 adj.他们的 她们的 它们的

theirs [ðɛəz] pron.他(她)们的东西

them [θem] pron.(宾格)他(她、它)们

themselves [ðəm'selvz] pron.他(她、它)们自己

then [ðen] adv.当时 然后 那么 而且 adj.当时的 n.当时

theoretical [θiə'retikəl] adj.理论(上)的

theory ['θiəri] n.理论 原理 学说 意见

there [ðɛə, ðə] adv.在那里 在那点上 从那里 到那里 n.那一点 那个地方 pron.表示存在(与be连用)

thereby ['ðɛə'bai] adv.因此 从而 由此

therefore ['ðɛəfɔ:] adv.因此 所以

thermometer [θə'mɔmitə] n.温度计 寒暑表

these [ði:z] adj. & pron.这些

they [ðei] pron.(主格)他(她)们

thick [θik] adj.厚的 粗的 密的 浓 adv.厚 浓 密 n.密集处 最厚处

thickness ['θiknis] n.厚(度) 密(度) 一层 含混不清

thief [θi:f] n.窃贼 偷窃犯 小偷

thin [θin] adj.薄的 淡的 稀的 vi.变薄 变瘦 adv.薄薄地 稀疏地

thing [θiŋ] n.物 用品 事 情况 (复)things：所有物 财产 用品

think [θiŋk] vt.想 想要 认为 考虑 记起 n.想

third [θə:d] num.第三 三分之一 adj.第三的 三分之一的

thirdly ['θə:dli] adv.第三

thirst [θə:st] n.渴 口渴 渴望 vi.(for)渴望 渴求

thirsty ['θə:sti] adj.口渴的 渴望的

thirteen ['θə:ti:n] num.十三 十三个

thirty ['θə:ti] num.三十 三十个

this [ðis] adj.这 今 pron.这 这个 adv.这么 这样

thorn [θɔ:n] n.刺 棘 荆棘 蒺藜

thorough ['θʌrə] adj.彻底的 详尽的 完全的 精心的

those [ðəuz] pron.那些 adj.那些

though [ðəu] conj.虽然 尽管 adv.可是 然而 不过

thought [θɔ:t] n.思想 思维 想法 观念 意向 动词think的过去式和过去分词

thoughtful ['θɔ:tfəl] adj.沉思的 体贴的

thousand ['θauzənd] num.千 千个 n.无数 千 (复)thousands：成千上万 许许多多

thread [θred] n.线 细丝 螺纹 头绪 线索 思路 vt.穿线于 穿过 遍布 vi.小心地通过 蜿蜒前进 滴下成丝状

threat [θret] n.威胁 恐吓 凶兆 vt.威胁 恐吓

threaten ['θretn] vt. & vi.威胁 恐吓

three [θri:] num.三 三个 pron. & adj.三(个 只...)

thrill [θril] n.一阵激动 震颤 激动 v.(使)震颤 兴奋 激动

thrive [θraiv] vi.兴旺 繁荣 旺盛 茁壮成长

throat [θrəut] n.咽喉 喉咙 嗓音 vt.低语 压着嗓子说(唱)

throne [θrəun] n.宝座 御座 王位 君主

throng [θrɔ:ŋ] n.群 人群 群众 一大群 大量 vt.挤满 群集 挤压 adj.拥挤的 忙的

through [θru:] prep.经过 穿过 凭借 adv.从头到尾 穿越 完全 adj.完成的 完结的 贯穿的 直通的

throughout [θru:'aut] prep.遍及 贯穿 adv.到处 自始至终

throw [θrəu] vt.投 掷 抛 扔 n.投掷 抛

thrust [θrʌst] vt.插 刺 推挤 n.插 讽刺 推力 刺 力推

thumb [θʌm] n.(大)拇指 v.翻阅 示意要求搭车

thunder ['θʌndə] n.雷 雷声 vi.打雷 轰隆地响 vt.吼出 大声喊出

Thursday ['θə:zdi] n.星期四

thus [ðʌs] adv.如此 这样 因而 因此 从而

tick [tik] n. 滴答声 记号 勾号 虱类 扁虱 被套 枕套 褥套 条纹厚棉布 <英>赊账 赊欠 vt. 给...标记号 vi.滴答作响 运作

ticket ['tikit] n.票 券 标签 罚款单 纲领 传票 vt.加标签于 售票

tide [taid] n.潮 潮汐 潮流 潮水 趋势

tidy ['taidi] adj.整洁的 整齐的 <口>相当大的 vt.使整洁 整理 vi.整理 收拾 n.(沙发,椅子等的)背罩 装杂物的容器

tie [tai] n.领带 结 束缚 不分胜负 联系 vt.系 约束 捆绑 vi.打结 连结

tiger ['taigə] n.老虎 凶残之人

tight [tait] adj.紧的 紧身的 严厉的 密集的 拮据的 adv.紧紧地 牢固地 迅速地

till [til] prep. & conj.直到...为止 vt.耕作 n.放钱的抽屉 冰碛

timber ['timbə] n.木材 木料

time [taim] n.时间 时机 次数 回 倍 vt.为...安排时间 计时 定时 乘 v.[计算机]DOS命令 显示并允许改变系统时间 adj.根据时间的 时间上的

timetable ['taim.teibl] n.时间表 时刻表 课程表

timid ['timid] adj.胆怯的 羞怯的

tin [tin] n.锡 罐头 听头 adj.锡制的 vt.镀锡于

tiny ['taini] adj.微小的 极小的

tip [tip] n.梢 末端 尖 尖端 小费 提示 轻击 翻倒 倾斜 vi.给小费 装顶端 去掉...的顶部 倾斜 翻倒 vt.轻击

tire ['taiə] vi.疲劳 累 厌倦 vt.使疲倦 厌烦 n.轮胎 金属圈

tired ['taiəd] adj.疲劳的 疲倦的 累的 厌倦的 装备以轮胎的 动词tire的过去式和过去分词形式

tissue ['tiʃu:] n.薄绢 薄纸 (生物的)组织 织物 纸巾

title ['taitl] n.标题 题目 称号 头衔 称号 vt.予称号 加标题于 adj.标题的

to [tu:] prep.(表示方向)到 向 达到 (表示间接关系)给 离 对于 为了 adv.(表示方向)去 (门)关上

toast [təust] n.烤面包 吐司 祝酒 祝酒词 被祝酒的人 vt.烘 烤 vi. (为 ... )祝酒

tobacco [tə'bækəu] n.烟草 烟叶

today [tə'dei] adv.在今天 现在 当今 n.今天 当今 adj.现在的

toe [təu] n.脚趾 足尖 vt.趾触 趾踢 vi.踮着脚尖走

together [tə'geðə] adv.共同 一起 adj.妥当的 镇定的

toilet ['tɔilit] n.厕所 盥洗室 浴室 梳洗打扮 礼服 创口洗涤 v.打扮 使用厕所

tolerance ['tɔlərəns] n.忍受 容忍 忍耐力 宽容 公差

tolerate ['tɔləreit] vt.忍受 容忍 宽恕

tomato [tə'mɑ:təu, tə'meitəu] n.番茄 西红柿

tomb [tu:m] n.坟 冢 vt.埋葬

tomorrow [tə'mɔrəu] adv.在明天 n.明天 将来

ton [tʌn] n.吨 大量

tone [təun] n. 音 腔调 声调 音调 语气 品质 色调 vt. 使更健壮 装腔作势地说 定调 调色 vi.调和(颜色) 呈现悦目色调

tongue [tʌŋ] n.舌 舌头 舌状物 语言 口才 v.舔 用舌吹奏管乐器 发音

tonight [tə'nait] adv.在今夜 n.今夜

too [tu:] adv.也 还 太 过分 很

tool [tu:l] n.工具 器具 用具 手段 被利用者 印压的图案 vt.用工具加工 vi.使用工具

tooth [tu:θ] n.牙齿 齿状物 粗糙表面 嗜好 (复)teeth [ti:θ]：有效手段

top [tɔp] n.顶 顶部 首位 adj.顶的 最高的 顶端的 v.高过 加顶 高耸

topic ['tɔpik] n.题目 论题 话题

torch [tɔ:tʃ] n.火炬 火把 手电筒 纵火犯 v.用火炬点燃

torrent ['tɔrənt] n.奔流 激流 洪流 山洪 adj.奔流的 汹涌的

tortoise ['tɔ:təs] n.龟 乌龟

torture ['tɔ:tʃə] n.拷问 折磨 痛苦 vt.拷打 拷问 使痛苦 曲解

toss [tɔs] vt. & vi.扔 抛 掷 投掷 摇荡 辗转 n.投掷 震荡

total ['təutl] adj.总的 全体的 完全的 vi.合计 总共 n.总数 合计

touch [tʌtʃ] vt.触摸 触动 吃 喝 触及 感动 vi.接触 联系 n.触 触觉 接触 少许 痕迹 修饰

tough [tʌf] adj.坚韧的 健壮的 强硬的 艰苦的 棘手的 严厉的 adv.以强硬的态度 n.暴徒 恶棍

tour [tuə] n. & vi.旅行 游历 旅游 周游 观光 巡回

tourist ['tuərist] n.旅游者 观光者

toward [tə'wɔ:d] prep.向 对于 接近 关于 将近 朝 adj.有希望的 有利的 逼近的

towards [tə'wɔ:dz] prep.向 朝 关于 adj.在即的 有希望的 = toward

towel ['tauəl, taul] n.毛巾 手巾 v.用毛巾擦

tower ['tauə] n.塔 高楼 vi.屹立 高耸

town [taun] n.镇 市镇 城镇 城市

toy [tɔi] n.玩具 玩物 adj.供玩耍的 玩具的 v.玩弄

trace [treis] n.痕迹 踪迹 微量 丝毫 vt.跟踪 追踪 找出根源 描绘 vi.追踪

track [træk] n.行踪 踪迹 路径 轨道 小路 跑道 惯例 常规

tractor ['træktə] n.拖拉机 牵引车 螺旋桨飞机

trade [treid] n.贸易 交易 买卖 职业 vi.交易 进行交易 做买卖 交换 adj.贸易的 商业的

tradition [trə'diʃən] n.传统 惯例

traditional [trə'diʃənəl] adj.传统的 惯例的

traffic ['træfik] n.交通 通行 (人或车等)交通流量 不正当生意(走私) v.做生意(多指违法的) 游览

tragedy ['trædʒidi] n.悲剧 惨事 惨案 灾难

trail [treil] n.痕迹 踪迹 小径 尾 vt.跟踪 追踪 拖拽 vi.拖 尾随 追踪 落后于

train [trein] vt.培养 训练 培训 n.列车 火车 队列 行列 一系列

training ['treiniŋ] n.训练 锻炼 培养 动词train的现在分词和动名词形式

traitor ['treitə] n.叛徒 卖国贼 出卖朋友者

tram [træm] n.(有轨)电车 矿车 煤车 缆车 (复数)trams：有轨电车轨道 vt.用缆车 电车等运输

tramp [træmp] n.流浪者 游民 重步声 v.践踏 重步行走

transfer [træns'fə:] vt. & vi.转移 调动 调转 调任 n.迁移 移动 换车

transform [træns'fɔ:m] vt.改变 改造 变换 转换 变形 vi.改变 n.变形

transformation [.trænsfə'meiʃən] n.变化 改造 转变 转型 转化

transformer [træns'fɔ:mə] n.变压器 转换器

transistor [træn'zistə] n.晶体管 晶体管收音机

translate [træns'leit] vt. & vi.翻译 译 解释 转移 调动

translation [træns'leiʃən] n.翻译 译文 译本 RNA的转录 平移

transmission [træns'miʃən] n.传送 传动 发射 播送 变速器 传递的信息

transmit [trænz'mit] vt.传送 传达 发射 传输 代代相传 vi.(以无线电或有线电的方式)发送信号

transparent [træns'perənt] adj.透明的 易识破的 明显的 清晰的

transport [træns'pɔ:t] vt.运输 流放 为强烈的情绪所激动 n.运输、运输工具 (常用复数)强烈的情绪(狂喜或狂怒)

transportation [.trænspə'teiʃən] n.运输 运送 客运 运输系统 运输工具

trap [træp] n.陷阱 诡计 圈套 困境 双轮轻便马车 vt.诱骗 设圈套 陷入(困境)

travel ['trævl] vt. & vi. & n.旅行 传播 移动

tray [trei] n.(浅)盘 托盘 碟

treason ['tri:zn] n.谋反 通敌 叛国罪 叛逆 背叛

treasure ['treʒə] n.财富 珍宝 vt.珍视

treat [tri:t] vt.对待 处理 治疗 请客 视为 n.款待 宴飨

treatment ['tri:tmənt] n.待遇 对待 处理 治疗 疗法

treaty ['tri:ti] n.条约 协议 协定

tree [tri:] n.树 树状物 vt.把…赶上树 使…走投无路

tremble ['trembl] vi.发抖 哆嗦 摇动 战悚 忧虑 微动 n.战悚 颤抖

tremendous [tri'mendəs] adj.极大的 非常的 巨大的 惊人的

trend [trend] vi.伸向 倾向 转向 n.趋势 倾向 方位

trial ['traiəl] n.试 试验 试用 审判 讯问 审讯 尝试 努力 讨厌的事 人 adj.尝试性的 审讯的

triangle ['traiæŋgl] n.三角(形)

tribe [traib] n.部落 宗族 种族 一伙人

trick [trik] n.诡计 欺诈 把戏 诀窍 vt.哄骗 戏弄 欺骗 adj.有诡计的 有阴谋的

trifle ['traifl] n.小事 琐事 少许 少量 v.浪费 玩忽

trim [trim] adj.整齐的 匀称的 vt.使整齐 使整洁 修除 削剪 装备 痛打 击败 n.整齐 装饰 修剪下来的部分

trip [trip] vi. 绊 绊倒 失足 轻快地走 失误 结巴 旅行 vt. 绊倒 使跌倒 使犯错 使失败 起锚 竖帆

n. 旅行 摔倒 差错 轻快的步伐

triumph ['traiəmf] n.凯旋 胜利 欢欣 vi.成功 得胜 庆功

troop [tru:p] n.军队 一群 组 大量 vi.群集 结队 成群前行

tropical ['trɔpikəl] adj.热带的 炎热的 热带植物的

trouble ['trʌbl] n.烦恼 困难 烦恼 vi.烦恼 麻烦 苦恼 费神

troublesome ['trʌblsəm] adj.令人烦恼的 麻烦的 讨厌的 困难的 棘手的

trousers ['trauzəz] n.裤子 长裤

truck [trʌk] n.卡车 载重汽车 对...进行交易 来往 实物工资 v.以货易货 交往 以卡车运输

true [tru:] adj.真的 真实的 忠实的 adv.真实地 n.真相 精确 vt.使...平衡

truly ['tru:li] adv.真正地 忠实地 真实地 真诚地 正确地

trumpet ['trʌmpit] n.喇叭 小号 喇叭声 喇叭手 vt.宣扬 鼓吹 吹嘘 vi.吹喇叭 发出喇叭似的声音

trunk [trʌŋk] n.树干 大衣箱 皮箱 躯干 干线 象鼻 (汽车后部)行李箱 (复)trunks：男用运动裤

trust [trʌst] n.信任 信托 vt.相信 委托 vi.信任

truth [tru:θ] n.真理 真实 真实性 真像 事实

try [trai] vt.试 尝试 试验 审问 审判 考验 vi.尝试 试验 n.尝试 试验

tub [tʌb] n.桶 盆 浴缸 浴盆 v.装入桶 洗澡

tube ['tju:b] n.管 管状物 电子管 显像管 电视 地铁 气管

tuck [tʌk] vt.折短 卷起 塞 打摺 挤进 畅饮 vi.缝褶裥 缩拢 畅饮 n.缝摺 船尾突出部下方 食品 鼓声

Tuesday ['tju:zdi] n.星期二

tuition [tju:'iʃən] n.教 教诲 教学 学费

tumble ['tʌmbl] vi.摔倒 跌倒 打滚 下跌 倒塌 翻跟斗 偶然发现 理解 vt.使摔倒 使混乱 翻滚 n.跌倒 混乱

tune [tju:n] n.调子 和谐 曲调 协调 调整 vt.调整 为...调音 vi.和谐 调音

tunnel ['tʌnl] n.隧道 坑道 地道 v.挖隧道 挖地道

turbine ['tə:bin] n.叶轮机 汽轮机 涡轮

turbulent ['tə:bjulənt] adj.骚动的 骚乱的 狂暴的 动荡的 汹涌的

turkey ['tə:ki] n.火鸡 火鸡肉 笨蛋 失败之作 Turkey：土耳其

turn [tə:n] vt.转 翻 vi.转动 旋转 转向 轮流 翻转 n.转动 轮流

turning ['tə:niŋ] n.旋转 回转 转向 变向 转弯处

turnip ['tə:nip] n.萝卜 芜菁 大头菜 <俚>大怀表

tutor ['tju:tə] n.家庭教师 导师 v.当家庭教师 当导师 指导

twelfth [twelfθ] num.第十二 adj.第十二的 十二分之一的 n.第十二 十二日 第十二音

twelve [twelv] num.十二 十二个

twentieth ['twentiiθ] num.第二十 n.二十分之一 adj.第二十的

twenty ['twenti] num.二十 二十个

twice [twais] adv.两次 两倍

twin [twin] adj.孪生的 n.孪生儿 双胞胎之一

twinkle ['twiŋkl] vi.闪烁 闪耀 使...闪耀 眨眼 迅速移动 n.闪烁 闪耀 眨眼 瞬息

twist [twist] vt. & vi.拧 捻 搓 扭曲 弯曲 n.扭曲 弯曲 盘旋 捻 拧

two [tu:] num.二 两个

type [taip] n.型 类型 种类 典型 铅字 vi.打字

typewriter ['taip.raitə] n.打字机 打字员

typhoon [tai'fu:n] n.台风

typical ['tipikəl] adj.典型的 代表性的 特有的 独特的

typist ['taipist] n.打字员

tyre [taiə] n.轮胎 车胎 =tire(美) Tyre：提尔(古代腓尼基的有名港口 现属黎巴嫩)

## U

ugly ['ʌgli] adj.丑陋的 难看的 可怕的

ultimate ['ʌltimit] adj.最后的 最终的 终极的 根本的 极限的 n.终极 根本 精华

ultimately ['ʌltimitli] adv.最终 最后

umbrella [ʌm'brelə] n.伞 雨伞 adj.像伞状分布的 vt.(用伞)遮住

unable [ʌn'eibl] adj.不能的 不会的

unbearable ['ʌn'bɛərəbl] adj.难堪的 忍受不了的

uncertain [ʌn'sə:tn] adj.无常的 不确定的

uncle ['ʌŋkl] n.伯父 叔父 舅父 姨父 姑父 伯 舅 叔

uncomfortable [ʌn'kʌmftəbl] adj.不舒服的 不自在的

unconscious [ʌn'kɔnʃəs] adj.不省人事的 失去知觉的

uncover [ʌn'kʌvə] vt.揭开(...的盖子) 揭露

under ['ʌndə] prep.在...下面 低于 在...内 在...控制下 adv.在...下 在昏迷中 adj.下面的 从属的 少于的

undergo [.ʌndə'gəu] vt.经历 经受 忍受 遭受

undergraduate [.ʌndə'grædər'grædʒuət] n.大学肆业生 大学生 adj.大学生的 大学的

underground ['ʌndəgraund] adj.地下的 秘密的 adv.在地下 秘密地 n.地下 地铁 地道 秘密活动

underline [.ʌndə'lain] vt.划线于...之下 在...下面划线

underneath [.ʌndə'ni:θ] adv.在下面 在底下 n.下部 底部 prep.在...下面

understand [.ʌndə'stænd] vt.获悉 理解 听说 将...理解为 认为 vi.懂得 理解 获悉

understanding [.ʌndə'stændiŋ] n.理解 理解力 协定 了解 个人见解 非正式协议 谅解 同情心 洞察力 adj.有同情心的

undertake [.ʌndə'teik] vt.从事 承担 保证 答应

undertaking [.ʌndə'teikiŋ] n.任务 事业 企业 保证 许诺

undo ['ʌn'du:] vt.解开 打开 松开 取消

undoubtedly [ʌn'dautidli] adv.无容置疑 肯定地

uneasy [ʌn'i:zi] adj.心神不安的 忧虑的 不自在的 不稳定的 不舒服的

unexpected ['ʌnik'spektid] adj.想不到的 意外的

unfair [ʌn'fɛə] adj.不公平的 不公正的 (商业上)不正当的

unfortunate [ʌn'fɔ:tʃənit] adj.不幸的 可取的 令人遗憾的 不成功的 n.不幸的人

unfortunately [ʌn'fɔ:tʃənitli] adv.不幸地

unhappy [ʌn'hæpi] adj.不幸福的 不快乐的

uniform ['ju:nifɔ:m] adj.一样的 一致的 统一的 n.制服 vt.穿制服 使统一

union ['ju:njən] n.联合 团结 协会 结合 政党 工会 连接轴 adj.联邦政府的 工会的

unique [ju:'ni:k] adj.唯一的 独一无二的 独特的 稀罕的

unit ['ju:nit] n.单位 单元 部件 组件

unite [ju(:)'nait] vi.联合 合并 统一 团结 vt.使联合

unity ['ju:niti] n.单一 统一 团结 个体 一致 结合

universal [.ju:ni'və:səl] adj.宇宙的 普遍的 通用的 全体的 全世界的 n.通用

universe ['ju:nivə:s] n.宇宙 世界 万物

university [.ju:ni'və:siti] n.大学 综合性大学

unjust ['ʌn'dʒʌst] adj.非正义的 不公平的

unkind [ʌn'kaind] adj.不仁慈的 不和善的 无情的 不厚道的

unknown ['ʌn'nəun] adj.不知道的 未知的 不出名的

unless [ən'les] conj.除非 如果不 prep.除...外

unlike ['ʌn'laik] adj.不同的 不相似的 prep.不象... 和...不同

unlikely [ʌn'laikli] adj.未必的 未必可能的

unload ['ʌn'ləud] vt.卸(货) 卸下 摆脱...之负担 (大量)倾销 vi.卸货 摆脱负担 退出(弹药 胶片等)

unlucky [ʌn'lʌki] adj.不幸的 不吉的 倒霉的

unnecessary [ʌn'nesisəri] adj.不必要的 多余的

unpleasant [ʌn'pleznt] adj.令人不快的 讨厌的

unsatisfactory ['ʌn.sætisfæktəri] adj.不能令人满意的

unstable ['ʌn'steibl] adj.不稳固的 不稳定的 易变的

unsuitable['ʌn'sju:təbl] adj.不合适的 不适宜的

until [ən'til] prep. & conj.直到...为止

unusual [ʌn'ju:ʒuəl] adj.不平常的 独特的 异常的

unusually [ʌn'ju:ʒuəli] adj.不平常地 非常 异乎寻常地 显著地

unwilling ['ʌn'wiliŋ] adj.不愿意的

up [ʌp] adv.向上 起床 起来 往上 上升 prep.向上 adj.向上的 上行的 n.上升 v.上升 增加

upon [ə'pɔn] prep.在...上 在...旁

upper ['ʌpə] adj.上面的 地位较高的 上部的 较高的 n.鞋帮 上铺 n.兴奋剂 令人兴奋的经历

upright ['ʌp'rait] adj.垂直的 正直的 诚实的 合乎正道的

upset [ʌp'set] vt.弄翻 打翻 倾覆 推翻 扰乱 使心烦 adj.心烦的 苦恼的 不安的 n.推翻 混乱

upside-down ['ʌpsaid'daun] adj.颠倒的 乱七八糟的

upstairs ['ʌp'stɛəz] adv.在楼上 往楼上 高水平地 上层地 头脑上地 adj.楼上的 n.楼上

up-to-date ['ʌptə'deit] adj.直到最近的 现代的 最新的

upward ['ʌpwəd] adj.向上的 上升的 adv.向上 上升

upwards ['ʌpwədz] adv.趋向上升 以上 往上 向上地

urge [ə:dʒ] vt.推进 催促 驱策 鼓励 力陈 催促 vi.极力主张 n.冲动

urgent ['ə:dʒənt] adj.紧急的 强求的 坚持要求的 急迫的

us [ʌs] pron.( we的宾格)我们

usage ['ju:sidʒ] n.使用 用法 对待 惯用法 习惯

use [ju:s] vt.用 使用 利用 耗费 对待 v.<口>吸(烟) 服用(毒品) n.用 运用 用法 使用权 价值

used [ju:st] vi.过去常常 adj.使用过的 二手的 习惯的 动词use的过去式和过去分词

useful ['ju:sfəl] adj.有用的 有益的 实用的

useless ['ju:slis] adj.无用的 无价值的 无效的

user ['ju:zə] n.用户 使用者

usual ['ju:ʒuəl] adj.通常的 平常的 惯常的 惯例的

usually ['ju:ʒuəli] adv.通常

utility [ju:'tiliti] n.效用 实用 实用品 公共设施 公用程序 adj.多效用的 多功能的

utilize ['ju:tilaiz] vt.利用

utmost ['ʌtməust] adj.最远的 极度的 n.极限 最大可能

utter ['ʌtə] adj.完全的 彻底的 全然的 绝对 vt.发出 作声 发表 说 讲

## V

vacant ['veikənt] adj.空的 空虚的 木然的 未被占用的

vacation [vei'keiʃən] n.假期 休假 腾出 vi.度假

vacuum ['vækjuəm] n.真空 空间 真空吸尘器 adj.真空的 产生(利用)真空的 v.用吸尘器打扫

vague [veig] adj.模糊的 含糊的 不明确的 犹豫不决的 茫然的

vain [vein] adj.徒劳的 无效的 自负的 虚荣的

valid ['vælid] adj.有效的 有根据的 正当的 合法的

valley ['væli] n.(山)谷 溪谷 流域 adj.流域 山谷 峡谷

valuable ['væljuəbl] adj.值钱的 有价值的 n.(pl.)贵重物品

value ['vælju:] n.价值 重要性 价格 (pl)价值观 vt.评价 估价 重视

van [væn] n.大篷车 运货车 面包车 vt.(用货车)运 n.前卫 前锋 n.翼

vanish ['væniʃ] vi.突然不见 消失 绝迹 vt.消失

vanity ['væniti] n.虚荣心 虚夸 浮华 无价值的东西 adj.虚荣心的

vapour ['veipə] n.汽 蒸气 雾气 v.蒸发 吹嘘 =vapor(英)

variable ['vɛəriəbl] adj.易变的 可变的 n.变量 易变的东西

variation [.vɛəri'eiʃən] n.变化 变动 变异 变种 变奏曲

variety [və'raiəti] n.多样化 种类 变种 杂耍

various ['vɛəriəs] adj.各种各样的 不同的

vary ['vɛəri] vt.改变 变化 使多样化

vase [veis] n.瓶 花瓶

vast [vɑ:st] adj.巨大的 大量的 广阔的 n.浩瀚的太空

vegetable ['vedʒitəbl] n.植物 蔬菜 植物人 adj.蔬菜的 植物的

vehicle ['vi:ikl] n.车辆 机动车 交通工具 手段 工具 传播媒介

veil [veil] n.面纱 面罩 遮蔽物 掩饰物 修女 vt.给...戴面纱或面罩 vi.带面纱或面罩

velocity [vi'lɔsiti] n.速度 速率 迅速

velvet ['velvit] n.丝绒 天鹅绒 adj.天鹅绒的

venture ['ventʃə] n. & vi.冒险 风险 投机 vt.敢于 尝试 谨慎地做 冒险一试

verb [və:b] n.动词

verify ['verifai] vt.证实 查证 证明 核实

version ['və:ʃən] n.译文 说法 改写本 版本 译本 形式

vertical ['və:tikəl] a.垂直的 竖式的 顶点的 纵向的 n.垂直物 垂直的位置

very ['veri] adv.很 完全 真正地 adj.真的 真正的 同一的 仅仅的 恰好的

vessel ['vesl] n.容器 器皿 船 飞船 管 脉管 血管 [植]导管

vest [vest] n.汗衫 背心 马甲 内衣 vt.授予 给予 使穿衣 vi.(权力、财产等)归于 穿衣服

veteran ['vetərən] n.老兵 老手 adj.老练的

vex [veks] vt.使烦恼 使恼火 恼怒

via ['vaiə] prep.经过 通过

vibrate ['vaibreit] vt.使颤动 (使)振动 (使)摇摆 vi.颤动 回响

vibration [vai'breiʃən] n.颤动 振动 摆动

vice [vais] n.罪恶 恶习 缺点 邪恶 老虎钳 adj.副的 代理的 prep.代替

victim ['viktim] n.牺牲者 受害者 牺牲

victorious [vik'tɔ:riəs] adj.胜利的 得胜的

victory ['viktəri] n.胜利 战胜 成功

video ['vidiəu] adj.电视的 视频的 录像的 n.电视 视频 录像

view [vju:] n.看 视力 风景 视野 见解 方法 检查

viewpoint ['vju:pɔint] n.观点 看法 见解

vigorous ['vigərəs] adj.朝气蓬勃的 精力充沛的 元气旺盛的 有力的

village ['vilidʒ] n.乡村 村庄

vine [vain] n.葡萄树 攀爬植物 藤 蔓

vinegar ['vinigə] n.醋

violence ['vaiələns] n.猛烈 激烈 暴力 强暴 暴行

violent ['vaiələnt] adj.猛烈的 狂暴的 暴力的 极端的

violet ['vaiəlit] n.紫罗兰 紫色 adj.紫色的

violin [.vaiə'lin] n.小提琴

virtually ['və:tʃuəli] ad.v实际上 事实上 几乎

virtue ['və:tju:] n.善 美德 优点 德行 贞操

visible ['vizəbl] adj.可见的 看得见的 n.可见物

vision ['viʒən] n.视力 视觉 眼力 想象力 梦幻 幻觉 vt.幻想 设想

visit ['vizit] vt. & n.访问 参观 拜访 探访

visitor ['vizitə] n.访问者 游客 参观者

visual ['vizjuəl] adj.看的 看得见的, 视觉的 n.画面 图象

vital ['vaitl] adj.生命的 有生命力的 至关重要的 生死攸关的 有活力的

vitamin ['vaitəmin] n.维生素 维他命

vivid ['vivid] adj.鲜艳的 生动的 栩栩如生的

vocabulary [və'kæbjuləri] n.词汇表 词汇 语汇 词汇量

voice [vɔis] n.说话声 声音 嗓音 意见 [语]语态 意见 发言权 vt.表达 发音 调整声音

volcano [vɔl'keinəu] n.火山

volleyball ['vɔlibɔ:l] n.排球 排球运动

volt [vəult] n.(电压单位)伏特 伏 voltage ['vəultidʒ] n.电压

volume ['vɔljum] n.卷 册 容积 体积 音量

voluntary ['vɔləntəri] adj.自愿的 志愿的 n.(教堂礼拜仪式前后或进行期间的)乐器独奏

vote [vəʊt] n.选举 投票 表决 投票结果 选举权 v.投票 选举 表决

voyage ['vɔiidʒ] n. & vi.航海 航空 航行 旅行 旅程

## W

wage [weidʒ] n.工资 报酬 vt.从事 开展 进行(战争、运动)

waggon ['wægən] n.四轮运货马车 敞篷货车 =wagon(美)

waist [weist] n.腰 腰部

wait [weit] vi.等 等候 n.等待 等候 公共乐队队员 沿街唱圣诗的人们

waiter ['weitə] n.侍者 (男)服务员 托盘

wake [weik] vi.醒 醒来 vt.唤醒 n.清醒 守夜

waken ['weikən] vi.醒来 vt.弄醒 动词wake的过去分词

walk [wɔ:k] vi.步行 散步 走 vt.陪...走 走过 使行走 n.散步 步行 人行道

wall [wɔ:l] n.墙 壁 围墙 城墙 墙状物 vt.用墙围住 用墙隔开 v.演戏般地转动(眼睛)

wallet ['wɔlit] n.钱包 皮夹子

wander ['wɔndə] vi.漫游 迷路 离题 徘徊 漫步 闲逛 蜿蜒 vt.漫步于 n.漫游 闲逛 徘徊

want [wɔnt] vt.要 想要 n.需要 缺乏 需求品

war [wɔ:] n.战争 冲突 斗争 竞争 vi.作战 斗争

warm [wɔ:m] adj.温暖的 暖和的 热烈的 热情的 v.使暖和 变暖 adv.温暖地

warmth [wɔ:mθ] n.暖和 温暖 热烈 热情 热心

warn [wɔ:n] vt.警告 告诫 预先通知 vi.发生警告

wash [wɔʃ] vt.洗 冲出 vi.洗涤 洗清 n.洗 洗濯 (美国西部)干河床

waste [weist] n.浪费 垃圾 荒地 v.浪费 滥用 消耗 adj.浪费的 无用的 荒芜的

watch [wɔtʃ] vt. & vi.观看 注视 看守 n.手表 看管 监视

water ['wɔ:tə] n.水 雨水 海水 海域 vt.使湿 灌溉 给...浇水 给...供水 加水稀释 形成水 喝水

waterfall ['wɔ:təfɔ:l] n.瀑布

waterproof ['wɔ:təpru:f] adj.不透水的 防水的 n.防水材料 (英)雨衣 vt.使...能防水

wave [weiv] n.波 波涛 波浪 波纹 vi.波动 挥动 飘动

wavelength ['weiv.leŋθ] n.波长 波段

wax [wæks] n.蜡 蜂蜡 (一阵)情绪 增大 vt. (用蜡)涂 得到...(更好的) vi.变大 增大

way [wei] n.路 路线 道路 方法 方式 方向 adv.非常 远远地 adj.与路线的中间点相关的,路线的中间点

we pron.(主格)我们 abbr.=Wednesday

weak [wi:k] adj.弱的 软弱的 虚弱的 无力的 淡薄的 差的

weaken ['wi:kən] vt.削弱 弄淡 vi.变弱 使...弱

weakness ['wi:knis] n.虚弱 软弱 弱点

wealth [welθ] n.财富 财产 丰富 大量

wealthy ['welθi] adj.富的 富裕的 n.富人

weapon ['wepən] n.武器 兵器 vt.武装

wear [wɛə] vt.穿着 戴 磨损 n.穿着 磨损 耐久性

weary a.疲倦的 vt.使疲乏

weather ['weðə] n.天气 气候 气象 vt.经受住 使风化 使日晒雨淋 vi.风化 褪色 adj.上风的

weave [wi:v] vt.织 编 组合 编排 vi.纺织 编织 n.编法 织法 编织

wedding ['wediŋ] n.婚礼 结合

Wednesday ['wenzdei, 'wenzdi] n.星期三

weed [wi:d] n.杂草 野草 weeds：衣服 丧服 vi.除草 清除 淘汰

week [wi:k] n.星期 周

weekday ['wi:kdei] n.周日 工作日

weekend [wi:k'end, 'wi:kend] n.周末 周末假期 vi.过周末 adj.周末的

weekly ['wi:kli] adj.每周的 一周一次的 周刊的 adv.每周 一周一次 n.周刊 周报

weep [wi:p] vi.哭泣 流泪 悲叹 渗出 n.哭泣 <俚>眼泪

weigh [wei] vt.称...的重量 掂量 衡量 重达... n.重量 道路

weight [weit] n.重 体重 砝码 重担 vt.使...负重 使倾斜

welcome ['welkəm] vt.欢迎 adj.受欢迎的 n.欢迎 int.欢迎(客人来访或新成员的加入)

weld [weld] vt.焊接 熔接 使结合 n.焊接 焊缝

welfare ['welfɛə] n.幸福 福利 福利事业 adj.福利的

well [wel] n.井 源泉 楼梯井 adj.健康的 良好的 适宜的 adv.很好地 适当地 完全地 vi.涌出 int.好吧 啊 嘿

well-known ['wel'nəun] adj.众所周知的 出名的

west [west] n.西 西方 西部 西洋 adj.西方的 adv.向(在)西方

western ['westən] adj.西方的 西部的 西洋的 n.西方人 西部片 西部小说

westward ['westwəd] adj.向西的 西方的 adv.向西 n.西方 西部

wet [wet] adj.湿的 下雨的 雨天的 n.潮气 雨天 vt.(使)弄湿

what [wɔt] pron.什么 adj.什么 任何 adv.什么 怎么

whatever [wɔt'evə] pron.无论什么 adj.不管怎样的 adv.无论如何 不管怎样

wheat [wi:t] n.小麦 小麦色

wheel [wi:l] n.轮 车轮 方向盘 周期 旋转 vi.旋转 转动 改变方向 vt.给...装轮子 用车运 转动

when [(h)wen] adv.什么时候 当...时 conj.当...时 然后 可是 pron.什么时候 n.时间

whenever [wen'evə] conj.无论何时 每当 随时 adv.无论何时

where [(h)wɛə] adv.在哪里 pron.哪里 conj.在...地方

wherever [wɛər'evə] adv.究竟在哪里 无论什么地方 在...的各个地方 conj.无论在哪里

whether ['(h)weðə] conj.是否 无论 不管

which [(h)witʃ] pron.哪一个 哪些 adj.哪一个 哪一些

whichever [witʃ'evə] adj.无论哪个 无论哪些 pron.无论哪个

while [(h)wail] conj.当...的时候 然而 虽然 尽管 n.一会儿 一段时间 vt.消磨

whilst [wailst] conj. & n.当...的时候 可是 虽然 只要

whip [wip] vt. 鞭笞 抽打 鞭策 快速移动 搅拌 搅打 煽动 锁边 n. 鞭子 鞭打 鞭状物 蛋奶水果甜点心 搅拌器 政党的组织秘书

whirl [wə:l] vt. & vi.使...旋转 急转 回旋 迅速移动 n.回旋 旋转 繁忙 混乱 尝试

whisky ['wiski] n.威士忌酒 =whiskey

whisper ['wispə] vt. 低声地讲 耳语 私语 vi. 低语 低声说 窃窃私语 飒飒地响 n. 低语 窃窃私语 飒飒的声音 流言蜚语 踪迹 暗示

whistle ['wisl] n. 口哨 汽笛 厂笛 啸啸声 用于召唤或发布命令的哨声 吹口哨 vi. 吹口哨 鸣汽笛 呼啸 吹口哨召唤 vt. 以吹口哨的方式发出 吹口哨发出信号或指引...

white [wait] adj.白色的 白种的 纯洁的 幸运的 n.白色 白种人

whitewash ['waitwɔʃ] vt.粉刷 涂白 用白色涂料粉刷 掩盖或粉饰 n.白色涂料 粉饰

who [hu:] pron.谁 ...的人

whoever [hu'evə] pron.无论谁 不管谁 任何人

whole [həul] adj.完整的 全部的 整体的 n.全部 完整 整体 adv.完全

wholly ['həulli] adv.完全地 全部 一概

whom [hu:m] pron.( who的宾格)谁

whose [hu:z] pron.谁的 哪个人的 adj.谁的

why [wai] adv.为什么 conj.为什么 ...的理由 int.咳 哎呀 n.理由 谜

wicked ['wikid] adj.坏的 令人厌恶的 邪恶的 缺德的 adv.极端地 非常地 vt.吸收

wide [waid] adj.宽阔的 广泛的 adv.全部地 广大地

widely ['waidli] adv.广 广泛地 adj.广泛的

widen ['waidn] vt.加宽 使变宽 vi.变宽

widespread ['waidspred] adj.分布(或散布)广的 普遍的

widow ['widəu] n.寡妇 遗孀 vt.使丧偶

width [widθ] n.宽阔 广阔 宽度

wife [waif] n.妻子 夫人

wild [waild] adj.野生的 野蛮的

will [wil] aux.v.将要 会 愿 必须

willing ['wiliŋ] adj.愿意的 心甘情愿的

win [win] vi.获胜 赢 vt.赢得 n.胜利

wind [wind,waind] n.风 气息 呼吸

window ['windəu] n.窗子 窗户 窗口 橱窗 视窗

wine [wain] n.葡萄酒 果酒 酒 紫红色 vi.喝酒 vt.请...喝酒

wing [wiŋ] n.翼 翅膀 翅

winner ['winə] n.获胜者 优胜者

winter ['wintə] n.冬天 冬季

wipe [waip] vt.擦 抹 揩 消除 拭去 用力打 n.擦拭 用力打

wire ['waiə] n.金属线 电缆 电线 电信 电报

wireless ['waiəlis] adj.不用电线的 无线的

wisdom ['wizdəm] n.智慧 才智 明智 至理名言 聪明

wise [waiz] adj.有智慧的 聪明的 明智的 有见识的

wish [wiʃ] vt.祝 想要 希望 n.希望 渴望 愿望 (复数)祝颂 命令

wit [wit] n.智力 才智 智能 机智

with [wið] prep.和...一起 具有 用 随着 包括

withdraw [wið'drɔ:] vt.收回 撤回 取回 撤退 vi.撤退 退回 走开

within [wi'ðin] prep.在...里面 adv.在内 n.里面 adj.在里面的

without [wið'aut] prep.无 没有 不 在外面

withstand [wið'stænd] vt.抵挡 反抗 对抗 经得起 承受

witness ['witnis] n.证据 目击者 证人

wolf [wulf] n.狼 残暴的人 v.狼吞虎咽

woman ['wumən] n.妇女 女人 女性 adj.女人的

wonder ['wʌndə] n.惊异 惊奇 奇迹 惊谔

wonderful ['wʌndəfəl] adj.惊人的 极好的 精彩的

wood [wud] n.树林 森林 木头 木材

wooden ['wudn] adj.木制的 呆板的 生硬笨拙的

wool [wul] n.羊毛 毛线 绒线 毛织品

woollen ['wulən] adj.羊毛制的 毛线的 n.毛织品

word [wə:d] n.词 单词 话 消息 语言 诺言 vt.用词语表达

work [wə:k] n. 工作 作品 职业

worker ['wə:kə] n.工人 工作者 人员

workman ['wə:kmən] n.工人 劳动者 工匠 技工

workshop ['wə:kʃɔp] n.车间 工场 创作室 研讨会 讲习班

world [wə:ld] n.世界 世人 世间 地球 领域 adj.世界的

world-wide ['wə:ld'waild] adj.遍及全球的 世界范围的

worm [wə:m] n.虫 蠕虫

worry ['wʌri] vt.使烦恼 使焦虑 撕咬

worse [wə:s] adj.更坏的 更差的

worship ['wə:ʃip] n.礼拜 做礼拜 崇拜 爱慕 vt.崇拜 爱慕 vi.做礼拜

worst [wə:st] adj.最坏的 最差的 最不利的

worth [wə:θ] adj.值...的 值得的 值钱的 n.价值 财富

worthless ['wə:θlis] adj.无价值的 无用的 可鄙的

worthwhile ['wə:θ'wail] adj.值得花时间(做)的

worthy ['wə:ði] adj.有价值的 可尊敬的 值得的 配得上的

would [wud] aux.v.将 愿 愿意 总是 总会 可能

wound [wu:nd] n.创伤 伤口 伤 vt.使受伤 使伤害

wrap [ræp] vt.裹 包 捆 覆盖 包围 n.披肩 围巾 包装材料

wreath [ri:θ] n.花环 花圈 花冠

wreck [rek] n.失事 残骸 vt.破坏 使遇难 使失事 遇难

wrist [rist] n.腕 腕关节 手腕 护腕

write [rait] vt.书写 写 vi.写

writer ['raitə] n.作者 作家 文学家

writing ['raitiŋ] n.书写 写 著作 写作 笔迹 作品

wrong [rɔŋ] adj.错误的 有毛病的 不正常的

## X

X-ray ['eksrei] n.X射线 X光 X光机 X光照片 vt.用X射线检查

## Y

yard [jɑ:d] n.院子 庭院 场地 码(长度单位)

yawn [jɔ:n] vi.打呵欠 n.呵欠

year [jə:, jiə] n.年 年年 年度 年龄 学年

yearly ['jə:li] adj.每年的 adv.一年一度地 每年地

yell [jel] vi.叫喊 n.大喊

yellow ['jeləu] adj.黄(色)的 n.黄(色) 蛋黄 黄种人 v.(使)变黄

yes [jes] adv.是 是的 n.是(表示肯定)

yesterday ['jestədi] n. & adv.昨天 昨日 近来

yet [jet] adv.还 已经 仍然 conj.而 然而 但是

yield [ji:ld] vt. & vi.出产 屈服 投降 n.产量 生产量

you [ju:] pron.你 你们

young [jʌŋ] adj.年轻的 没有经验的 初期的 n.青年人 幼崽

your [juə] pron.你的 你们的

yours [jɔ:z] pron.你们的(东西)

yourself [jɔ:'self] pron.你(们)自己 你(们)亲自

youth [ju:θ] n.青春 青年人 青年 年轻 初期

youthful ['ju:θfəl] adj.年轻的 青年的 有青春活力的

## Z

zeal [zi:l] n.热心 热情 热忱

zealous ['zeləs] adj.热心的 热情的 狂热的 热衷的

zebra ['zi:brə] n.斑马 斑马线

zero ['ziərəu] n. 零 零点 零度 最低点 完全没有 无足轻重的人或物

zone [zəun] n.地区 区域 范围 地域 地带

zoo [zu:] n.动物园 拥挤杂乱的地方

单词总结:一天一口气26个字母的单词都重复好几遍，效果不错！

# 第2季-小应用大智慧(Linux应用编程)-上学期

# 第01课-课程规划与学习方法

# 第02课-GDB程序调试

编译加上可调试手段：$ gcc –c –g hello.c –o hello.o

链接加上可调试手段：$ gcc –g hello.o –o hello.elf

$ gdb hello.elf 启动GDB

(gdb) list 列出源码

(gdb) list 12 列出12开始的源码

(gdb) break func1 在func1处设置断点

(gdb) break 15 在第15行设置断点

(gdb) break f1.c:13 在指定文件的指定行添加断点

(gdb) info breakpoint 查看所有断点

(gdb) delete 断点编号 删除指定断点

(gdb) run 开始运行程序

(gdb) next 单步不进函数

(gdb) step 单步进入函数

(gdb) continue 继续

(gdb) print var1 打印程序中var1的值

(gdb) set var=value 设置变量的值

(gdb) quit 退出GDB调试

# 第03课-Coredump程序故障分析

gdb调试core文件：

<http://blog.csdn.net/houzhuoming1/article/details/49950065>

Coredump定义

Coredump又叫核心转存。当程序在运行过程中发生异常，这时Linux系统可以把程序出错时的内存内容存储在一个core文件中，这种过程叫coredump。

## Segment fault出现原因

Linux应用程序在运行过程中，经常会遇到segment fault（段错误）这样的错误。原因有4：

\*\* 数组访问越界

\*\* 访问空指针

\*\* 栈溢出

\*\* 修改只读内存

## Core dump使能

在Linux系统中，默认是关闭coredump功能的，但是可以通过如下命令：

关闭：$ ulimit -c 0打开：$ ulimit -c unlimited

## Core文件分析

发生coredump之后，可以使用gdb进行查看core文件的内容，以定位程序出错的位置，用法：

用法：gdb 程序名 core文件

范例：gdb ./test test.core

范例演示

[root@redhat6 redhat6]# gcc -g test.c

[root@redhat6 redhat6]# ulimit -c unlimited

[root@redhat6 redhat6]# ./a.out

Segmentation fault (core dumped)

[root@redhat6 redhat6]# gdb a.out core.1108

GNU gdb (GDB) Red Hat Enterprise Linux (7.2-56.el6)

Copyright (C) 2010 Free Software Foundation, Inc.

License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software: you are free to change and redistribute it.

There is NO WARRANTY, to the extent permitted by law. Type "show copying"

and "show warranty" for details.

This GDB was configured as "i686-redhat-linux-gnu".

For bug reporting instructions, please see:

<http://www.gnu.org/software/gdb/bugs/>...

Reading symbols from /home/redhat6/a.out...done.

[New Thread 1108]

Missing separate debuginfo for

Try: yum --disablerepo='\*' --enablerepo='\*-debug\*' install /usr/lib/debug/.build-id/05/14ca88cad3d3d3eee1b7561eaf052da205c024

Reading symbols from /lib/libc.so.6...(no debugging symbols found)...done.

Loaded symbols for /lib/libc.so.6

Reading symbols from /lib/ld-linux.so.2...(no debugging symbols found)...done.

Loaded symbols for /lib/ld-linux.so.2

Core was generated by `./a.out'.

Program terminated with signal 11, Segmentation fault.

#0 0x080483a4 in main () at test.c:7

7 \*str = 16;

Missing separate debuginfos, use: debuginfo-install glibc-2.12-1.80.el6.i686

(gdb) list 7

2 #include <stdlib.h>

3 int main (void)

4 {

5 char \*str;

6 str = NULL;

7 \*str = 16;

8 return 0;

9 }

10

(gdb)

# 第04课-Linux应用程序地址布局

构成Linux应用程序重要组成部分：代码段+数据段+BSS段

BSS段：未初始化数据段、堆、栈。

Linux应用程序在内存中的布局

从低地址（0x8048000）到高地址分别为:

代码段+数据段+BSS段+堆+栈。堆向高内存生长，栈向低内存生长。

## Linux应用程序（代码段+数据段+BSS段）详解

范例程序：addr.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/types.h>

#include <unistd.h>

int global\_init\_a = 1;

int global\_uninit\_a;

static int static\_global\_init\_a = 1;

static int static\_global\_uninit\_a;

const int const\_global\_a = 1;

int main(void)

{

int local\_init\_a = 1;

int local\_uninit\_a;

static int static\_local\_init\_a = 1;

static int static\_local\_uninit\_a;

const int const\_local\_a = 1;

int \*malloc\_point\_a;

malloc\_point\_a = (int \*)malloc(sizeof(int));

printf("&global\_init\_a = %p\n", &global\_init\_a);

printf("&global\_uninit\_a = %p\n", &global\_uninit\_a);

printf("&static\_global\_init\_a = %p\n", &static\_global\_init\_a);

printf("&static\_global\_uninit\_a = %p\n", &static\_global\_uninit\_a);

printf("&const\_global\_a = %p\n", &const\_global\_a);

printf("&local\_init\_a = %p\n", &local\_init\_a);

printf("&local\_uninit\_a = %p\n", &local\_uninit\_a);

printf("&static\_local\_init\_a = %p\n", &static\_local\_init\_a);

printf("&static\_local\_uninit\_a = %p\n", &static\_local\_uninit\_a);

printf("&const\_local\_a = %p\n", &const\_local\_a);

printf("malloc\_point\_a = %p\n", malloc\_point\_a);

printf ("CURRENT PROCESS: %d\n", getpid());//获取本程序进程号

也可通过终端来得知该进程的进程号：# ps aux | grep “a.out”

while (1) ;//通过死循环让程序挂起以方便查看其在内存中的内容

return 0;

}

[root@redhat6 test]# ps -aux | grep "a.out"

Warning: bad syntax, perhaps a bogus '-'? See /usr/share/doc/procps-3.2.8/FAQ

root 3883 91.4 0.0 1980 408 pts/0 R+ 20:48 0:08 ./a.out

root 3891 0.0 0.0 4336 784 pts/1 S+ 20:48 0:00 grep a.out

[root@redhat6 test]# ls /proc/3883/

[root@redhat6 test]# 进程各个部分在内存中的地址分布情况

[root@redhat6 test]# cat /proc/3883/maps

001dc000-001fa000 r-xp 00000000 fd:00 542923 /lib/ld-2.12.so

001fa000-001fb000 r--p 0001d000 fd:00 542923 /lib/ld-2.12.so

001fb000-001fc000 rw-p 0001e000 fd:00 542923 /lib/ld-2.12.so

00202000-0038e000 r-xp 00000000 fd:00 542924 /lib/libc-2.12.so

0038e000-00390000 r--p 0018c000 fd:00 542924 /lib/libc-2.12.so

00390000-00391000 rw-p 0018e000 fd:00 542924 /lib/libc-2.12.so

00391000-00394000 rw-p 00000000 00:00 0

00eba000-00ebb000 r-xp 00000000 00:00 0 [vdso]

08048000-08049000r-xp 00000000 fd:00 297421 /home/redhat6/test/a.out代码段=只读+可执行

08049000-0804a000 rw-p 00000000 fd:00 297421 /home/redhat6/test/a.out数据段=紧跟代码段

0823d000-0825e000 rw-p 00000000 00:00 0 [heap] 堆

b7791000-b7792000 rw-p 00000000 00:00 0

b77a9000-b77ab000 rw-p 00000000 00:00 0

bfd1e000-bfd33000 rw-p 00000000 00:00 0 [stack]栈

神总结：某个变量地址，根据这个地址落在/proc/xxx/maps里的哪个区间，那么这个变量就属于哪个段。

[root@redhat6 test]# ls

addr.c a.out

[root@redhat6 test]# ./a.out

&global\_init\_a = 0x80498dc //结合进程对应maps:数据段

&global\_uninit\_a = 0x80498f8 //结合进程对应maps:数据段(bss)

&static\_global\_init\_a = 0x80498e0 //结合进程对应maps:数据段

&static\_global\_uninit\_a = 0x80498f0 //结合进程对应maps:数据段 (bss)

&const\_global\_a = 0x8048614 //结合进程对应maps:代码段

&local\_init\_a = 0xbfd31b98 //结合进程对应maps:栈

&local\_uninit\_a = 0xbfd31b94 //结合进程对应maps:栈

&static\_local\_init\_a = 0x80498e4 //结合进程对应maps:数据段

&static\_local\_uninit\_a = 0x80498f4 //结合进程对应maps:数据段 (bss)

&const\_local\_a = 0xbfd31b90 //结合进程对应maps:栈

malloc\_point\_a = 0x823d008 //结合进程对应maps:堆

CURRENT PROCESS: 3883 //通过getpid()函数得到该进程号

BSS段:readelf -S 程序名

[root@redhat6 test]# readelf -S a.out

There are 30 section headers, starting at offset 0xa3c:

Section Headers:

[Nr] Name Type Addr Off Size ES Flg Lk Inf Al

[ 0] NULL 00000000 000000 000000 00 0 0 0

[ 1] .interp PROGBITS 08048134 000134 000013 00 A 0 0 1

...

[24] .data PROGBITS 080498d8 0008d8 000010 00 WA 0 0 4

[25] .bss NOBITS 080498e8 0008e8 000014 00 WA 0 0 4

[26] .comment PROGBITS 00000000 0008e8 000058 01 MS 0 0 1

## 各个段存放内容

代码段：代码，全局常量(const)，字符串常量

数据段：全局变量(init/uninit)，静态变量(全局/局部，init/uninit)

BSS段：除了栈以外的所有未初始化变量

堆：动态分配的区域

栈：局部变量(init/uninit，但不包含静态变量)，局部只读变量,局部常量

# 第05课-Linux编程规范

文件头：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*文件名：addr.c

\*创建者：谢伟

\*创建时间：2016/11/15

\*程序说明：打印各种变量地址，分析程序布局

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

大括号对其

缩进：4空格缩进

# 第06课-静态函数库设计

外部函数

Linux应用程序设计中需要的外部函数主要由函数库和系统调用来提供。

函数库

静态链接库,　动态链接库ls /lib ; ls /usr/lib \*.so \*.a

静态链接库-特点

程序所要用的库函数代码在链接时全部被copy到程序中。

导致的问题：如果多个进程在内存中同时运行，并且使用了相同的库函数，那么会有多份拷贝，这就是对空间的浪费。

## 查看程序所依赖的库文件

$ readelf -d hello.elf

## 静态编译

$ gcc –static–g hello.c –o hello.elf

## 制作静态库

$ gcc –c hello.c –o hello.o

$ ar -cqs libhello.a hello.o

$ 将制作好的libhello.a复制到/usr/lib目录下

$ 使用外部的静态库

$ gcc test.c -lhello–o test.elf

# 第07课-动态函数库设计

## 制作动态链接库

$ gcc -c mylib.c -o mylib.o

$ gcc -shared -fPIC mylib.o -o libmylib.so

$ 将制作好的libmylib.so复制到/usr/lib目录下

-fPIC使输出的对象模块是按照可重定位地址方式生成的。

-shared指明产生动态链接库。

# 第08课-系统调用方式文件编程

文件描述符

在中国，每个成年的公民都会有一个身份证编号，他的本质就是一个数字，我们可以利用这个数字来标记这个公民。

在Linux系统中，所有打开的文件也对应一个数字，这个数字由系统来分配，我们称之为：文件描述符。

Linux应用程序编程学习方法

参考手册(资料)：《UNIX环境高级编程》

根据手册找到打开文件编程位置，找到相应的打开函数名，man这个函数名，参数，返回值。

## 打开文件

《UNIX环境高级编程》中找到文件打开函数名**:** open

在Linux中man这个函数名**:**man open**(**相关头文件，参数，返回值**)**

NAME

open**,** creat **-** open and possibly create a file or device

SYNOPSIS

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

int open**(**constchar**\***pathname**,**int flags**);**//打开文件

int open**(**constchar**\***pathname**,**int flags**,** mode\_t mode**);**//创建文件

int creat**(**constchar**\***pathname**,** mode\_t mode**);**

RETURN VALUE

open**()** and creat**()return** the new file descriptor**,** or **-**1**if** an error

occurred **(**in which **case,** errno is set appropriately**).**

## 创建文件

NAME

open**,** creat **-** open and possibly create a file or device

SYNOPSIS

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

int open**(**constchar**\***pathname**,**int flags**);**//打开文件

int open**(**constchar**\***pathname**,**int flags**,** mode\_t mode**);**//创建文件

int creat**(**constchar**\***pathname**,** mode\_t mode**);**

RETURN VALUE

open**()** and creat**()return** the new file descriptor**,** or **-**1**if** an error

occurred **(**in which **case,** errno is set appropriately**).**

## 关闭文件

NAME

close **-** close a file descriptor

SYNOPSIS

#include <unistd.h>

int close**(**int fd**);**

RETURN VALUE

close**()** returns zero on success**.** On error**,-**1 is returned**,** and errno

is set appropriately**.**

## 读文件:man 2 read

NAME

read **-** read from a file descriptor

SYNOPSIS

#include <unistd.h>

ssize\_t read**(**int fd**,**void**\***buf**,** size\_t count**);**

RETURN VALUE

On success**,** the number of bytes read is returned **(**zero indicates end of

file**),** and the file position is advanced by this number**.** It is not an

error **if** this number is smaller than the number of bytes requested**;**

this may happen **for** example because fewer bytes are actually available

right now **(**maybe because we were close to end**-**of**-**file**,** or because we

are reading from a pipe**,** or from a terminal**),** or because read**()** was

interrupted by a signal**.** On error**,-**1 is returned**,** and errno is set

appropriately**.** In this **case** it is left unspecified whether the file

position **(if** any**)** changes**.**

## 写文件:man 2 write

NAME

write - write to a file descriptor

SYNOPSIS

#include <unistd.h>

ssize\_t write(int fd, const void \*buf, size\_t count);

RETURN VALUE

On success, the number of bytes written is returned (zero indicates

nothing was written). On error, -1 is returned, and errno is set

appropriately.

If count is zero and fd refers to a regular file, then write() may

return a failure status if one of the errors below is detected. If no

errors are detected, 0 will be returned without causing any other

effect. If count is zero and fd refers to a file other than a regular

file, the results are not specified.

## 文件定位:man 2 lseek

NAME

lseek - reposition read/write file offset

SYNOPSIS

#include <sys/types.h>

#include <unistd.h>

off\_t lseek(int fd, off\_t offset, int whence);

RETURN VALUE

Upon successful completion, lseek() returns the resulting offset loca-

tion as measured in bytes from the beginning of the file. Otherwise, a

value of (off\_t) -1 is returned and errno is set to indicate the error.

## 复制文件描述符

NAME

dup, dup2, dup3 - duplicate a file descriptor

SYNOPSIS

#include <unistd.h>

int dup(int oldfd);

int dup2(int oldfd, int newfd);

#define \_GNU\_SOURCE

#include <unistd.h>

int dup3(int oldfd, int newfd, int flags);

RETURN VALUE

On success, these system calls return the new descriptor. On error, -1

is returned, and errno is set appropriately.

## 综合-文件复制程序

//文件名:fcopy.c

//作者:seafly

//创建时间:2016/11/15

//文件功能:文件复制程序(文件编程的综合应用)

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

int main (int argc, char \*argv[])

{

if (3 != argc)

{

printf ("Usage: %s %s %s\n", argv[0],"<src>","<dst>");

exit (1);

}

int fd1 = 0;

int fd2 = 0;

long nr = 0;

long nw = 0;

char buf[1024] = {0};

fd1 = open (argv[1], O\_RDONLY);

if (0 > fd1)

{

printf ("Error:open():%s:error\n", argv[1]);

exit (1);

}

fd2 = open (argv[2], O\_WRONLY|O\_CREAT, 0666);

if (0 > fd2)

{

printf ("Error:open():%s:error\n", argv[2]);

exit (1);

}

while ((nr = read (fd1, buf, sizeof(buf)))!=0)

{

nw = write (fd2, buf, nr);

if (0 > nw)

{

printf ("Error:write():error\n");

exit (1);

}

nr = lseek (fd1, 0, SEEK\_CUR);

if (0 > nr)

{

printf ("Error:lseek():error\n");

exit (1);

}

memset (buf, 0, sizeof(buf));

}

printf ("File copy successfully!\n");

return 0;

}

## 第\*\*课-Linux下C语言大文件读写(大于4G）

如何create大文件  
要大就非常大，1T吧。  
有两种方法：  
   
一.dd  
dd if=/dev/zero of=1T.img bs=1G seek=1024 count=0  
bs=1G表示每一次读写1G数据，count=0表示读写0次，seek=1024表示略过1024个Block不写，前面block size是1G，所以共略过1T！  
这是创建大型sparse文件最简单的方法。  
   
二.ftruncate64/ftruncate  
如果用系统函数就稍微有些麻烦，因为涉及到宏的问题。我会结合一个实际例子详细说明，其中OPTION标志的就是测试项。  
文件sparse.c:

# 第09课-库函数方式文件编程

库函数

基于C库函数的文件编程是独立于具体的操作系统平台的，不管是在windows、Linux还是其他的操作系统中，都是使用这些函数。使用库函数进行程序设计可提高程序的可移植性。

流

流其实是一种动态的概念，对于标准的C函数库，它们的操作都是围绕流来进行的。

流是一个抽象的概念，当程序需要读取数据的时候，就会一个通向数据源的流，这个数据源可以是文件，内存，或是网络连接。当文件写入时同理。

文件指针

在系统调用实现文件访问中，使用文件描述符来指向一个文件。在库函数方式的文件访问中，使用FILE类型来表示一个打开的文件，这个类型中包含了管理文件流的信息。而指向该类型的指针FILE \*则被称之为文件指针。

## 打开文件

NAME

fopen, fdopen, freopen - stream open functions

SYNOPSIS

#include <stdio.h>

FILE \*fopen(const char \*path, const char \*mode);

FILE \*fdopen(int fd, const char \*mode);

FILE \*freopen(const char \*path, const char \*mode, FILE \*stream);

DESCRIPTION

The fopen() function opens the file whose name is the string pointed to

by path and associates a stream with it.

r+ Open for reading and writing. The stream is positioned at the

beginning of the file.//修改文件特定位置内容时需要

RETURN VALUE

Upon successful completion fopen(), fdopen() and freopen() return a

FILE pointer. Otherwise, NULL is returned and errno is set to indicate

the error.

## 关闭文件

NAME

fclose - close a stream

SYNOPSIS

#include <stdio.h>

int fclose(FILE \*fp);

DESCRIPTION

The fclose() function will flushes the stream pointed to by fp (writing

any buffered output data using fflush(3)) and closes the underlying

file descriptor.

RETURN VALUE

Upon successful completion 0 is returned. Otherwise, EOF is returned

and errno is set to indicate the error. In either case any further

access (including another call to fclose()) to the stream results in

undefined behavior.

## 读文件

NAME

fread, fwrite - binary stream input/output

SYNOPSIS

#include <stdio.h>

size\_t fread(void \*ptr, size\_t size, size\_t nmemb, FILE \*stream);

size\_t fwrite(const void \*ptr, size\_t size, size\_t nmemb,

FILE \*stream);

DESCRIPTION

The function fread() reads nmemb elements of data, each size bytes

long, from the stream pointed to by stream, storing them at the loca-

tion given by ptr.

The function fwrite() writes nmemb elements of data, each size bytes

long, to the stream pointed to by stream, obtaining them from the loca-

tion given by ptr.

For non-locking counterparts, see unlocked\_stdio(3).

RETURN VALUE

fread() and fwrite() return the number of items successfully read or

written (i.e., not the number of characters). If an error occurs, or

the end-of-file is reached, the return value is a short item count (or

zero).

## 写文件

NAME

fread, fwrite - binary stream input/output

SYNOPSIS

#include <stdio.h>

size\_t fread(void \*ptr, size\_t size, size\_t nmemb, FILE \*stream);

size\_t fwrite(const void \*ptr, size\_t size, size\_t nmemb,

FILE \*stream);

DESCRIPTION

The function fread() reads nmemb elements of data, each size bytes

long, from the stream pointed to by stream, storing them at the loca-

tion given by ptr.

The function fwrite() writes nmemb elements of data, each size bytes

long, to the stream pointed to by stream, obtaining them from the loca-

tion given by ptr.

For non-locking counterparts, see unlocked\_stdio(3).

RETURN VALUE

fread() and fwrite() return the number of items successfully read or

written (i.e., not the number of characters). If an error occurs, or

the end-of-file is reached, the return value is a short item count (or

zero).

## 文件定位

NAME

fgetpos, fseek, fsetpos, ftell, rewind - reposition a stream

SYNOPSIS

#include <stdio.h>

int fseek(FILE \*stream, long offset, int whence);

long ftell(FILE \*stream);

void rewind(FILE \*stream);

int fgetpos(FILE \*stream, fpos\_t \*pos);

int fsetpos(FILE \*stream, fpos\_t \*pos);

RETURN VALUE

The rewind() function returns no value. Upon successful completion,

fgetpos(), fseek(), fsetpos() return 0, and ftell() returns the current

offset. Otherwise, -1 is returned and errno is set to indicate the

error.

# 第10课-时间编程

时间类型：世界标准时间(格林威治时间GMT)、日历时间(Calendar Time(CT))

格林威治时间(GMT):世界标准时间(UTC)

日历时间(Calendar Time):从一个标准时间点1970年1月1日0点到此时经过的秒数来表示时间。

## 获取日历时间:man 2 time

NAME

time - get time in seconds

SYNOPSIS

#include <time.h>

time\_t time(time\_t \*t);

DESCRIPTION

time() returns the time since the Epoch (00:00:00 UTC, January 1,

1970), measured in seconds.

If t is non-NULL, the return value is also stored in the memory pointed

to by t.

RETURN VALUE

On success, the value of time in seconds since the Epoch is returned.

On error, ((time\_t) -1) is returned, and errno is set appropriately.

用法：time\_t tm = 0L;

tm = time (NULL); 或tm = time(&tm);

## 时间处理函数

NAME

asctime, ctime, gmtime, localtime, mktime, asctime\_r, ctime\_r,

gmtime\_r, localtime\_r - transform date and time to broken-down time or

ASCII

SYNOPSIS

#include <time.h>

char \*asctime(const struct tm \*tm);

char \*asctime\_r(const struct tm \*tm, char \*buf);

char \*ctime(const time\_t \*timep);

char \*ctime\_r(const time\_t \*timep, char \*buf);

struct tm \*gmtime(const time\_t \*timep);

struct tm \*gmtime\_r(const time\_t \*timep, struct tm \*result);

struct tm \*localtime(const time\_t \*timep);

struct tm \*localtime\_r(const time\_t \*timep, struct tm \*result);

time\_t mktime(struct tm \*tm);

struct tm {

int tm\_sec; /\* seconds \*/

int tm\_min; /\* minutes \*/

int tm\_hour; /\* hours \*/

int tm\_mday; /\* day of the month \*/

int tm\_mon; /\* month \*/

int tm\_year; /\* year \*/

int tm\_wday; /\* day of the week \*/

int tm\_yday; /\* day in the year \*/

int tm\_isdst; /\* daylight saving time \*/

};

The members of the tm structure are:

tm\_sec The number of seconds after the minute, normally in the range

0 to 59, but can be up to 60 to allow for leap seconds.

tm\_min The number of minutes after the hour, in the range 0 to 59.

tm\_hour The number of hours past midnight, in the range 0 to 23.

tm\_mday The day of the month, in the range 1 to 31.

tm\_mon The number of months since January, in the range 0 to 11.

tm\_year The number of years since 1900.

tm\_wday The number of days since Sunday, in the range 0 to 6.

tm\_yday The number of days since January 1, in the range 0 to 365.

RETURN VALUE

Each of these functions returns the value described, or NULL (-1 in

case of mktime()) in case an error was detected.

## 时间处理函数范例用法

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int main (int argc, char \*argv[])

{

time\_t tm = time(NULL);

char \*buf = ctime(&tm);

printf ("%s", buf);

return 0;

}

//$ gcc time.c

//$ ./a.out

//Tue Nov 15 22:21:46 2016

## 获取高精度时间:gettimeofday()

NAME

gettimeofday, settimeofday - get / set time

SYNOPSIS

#include <sys/time.h>

int gettimeofday(struct timeval \*tv, struct timezone \*tz);

int settimeofday(const struct timeval \*tv, const struct timezone \*tz);

Feature Test Macro Requirements for glibc (see feature\_test\_macros(7)):

settimeofday(): \_BSD\_SOURCE

DESCRIPTION

The functions gettimeofday() and settimeofday() can get and set the

time as well as a timezone. The tv argument is a struct timeval (as

specified in <sys/time.h>):

struct timeval {

time\_t tv\_sec; /\* seconds \*/

suseconds\_t tv\_usec; /\* microseconds \*/

};

and gives the number of seconds and microseconds since the Epoch (see

time(2)). The tz argument is a struct timezone:

struct timezone {

int tz\_minuteswest; /\* minutes west of Greenwich \*/

int tz\_dsttime; /\* type of DST correction \*/

};

RETURN VALUE

gettimeofday() and settimeofday() return 0 for success, or -1 for fail-

ure (in which case errno is set appropriately).

## 高精度时间使用范例

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <sys/time.h>

int main (int argc, char \*argv[])

{

time\_t tm = time(NULL);

char \*buf = ctime(&tm);

printf ("%s", buf);

struct timeval tv1;

struct timeval tv2;

suseconds\_t us = 0;

gettimeofday(&tv1, NULL);

usleep (6160);

gettimeofday(&tv2, NULL);

us = tv2.tv\_usec - tv1.tv\_usec;

printf ("us: %ld\n", us);

return 0;

}

//$ gcc time.c

//$ ./a.out

//us: 6667

# 第11课-进程控制理论

进程概念：进程是一个具有一定独立功能的程序的一次运行活动。

进程状态：就绪 +　阻塞　+ 执行

进程调度：从等待的队列中选择一个进程来执行就叫进程调度。

阻塞状态：比如IO请求，如果IO得不到满足，该进程就会暂时阻塞。

进程ID(PID):标识进程的唯一数字。有进程ID(PID),父进程ID(PPID),启动进程的用户ID(UID)。

进程互斥

进程互斥是指当有若干个进程都要使用某一资源时，但该资源在同一时刻最多允许一个进程使用，这时其他进程必须等待，直到占用该资源者释放了该资源为止。(电话亭轮流打电话)

临界区

进程中访问临界资源的那段程序代码成为临界区。

进程同步

一组进程按一定顺序执行的过程称为进程间的同步。最有名的是生产者和消费者进程。

进程调度

按一定算法，从一组待运行的进程中选出一个来占有CPU运行。

调度算法

\*先来先服务

\*短进程优先调度

\*高优先级优先调度

\*时间片轮转法

调度时机：抢占式调度、非抢占式调度。

死锁：多个进程因竞争资源而形成一种僵局，导致这些进程都无法继续往前执行。按顺序申请资源可避免死锁

## 获取进程ID:getpid()

NAME

getpid, getppid - get process identification

SYNOPSIS

#include <sys/types.h>

#include <unistd.h>

pid\_t getpid(void);

pid\_t getppid(void);

DESCRIPTION

getpid() returns the process ID of the calling process. (This is often

used by routines that generate unique temporary filenames.)

getppid() returns the process ID of the parent of the calling process.

# 第12课-多进程程序设计

## 创建进程fork

一个现有进程可以调用fork()创建新进程

NAME

fork - create a child process

SYNOPSIS

#include <unistd.h>

pid\_t fork(void);

RETURN VALUE

On success, the PID of the child process is returned in the parent, and

0 is returned in the child. On failure, -1 is returned in the parent,

no child process is created, and errno is set appropriately.

NAME

vfork - create a child process and block(块,阻塞) parent

SYNOPSIS

#include <sys/types.h>

#include <unistd.h>

pid\_t vfork(void);

Feature Test Macro Requirements for glibc (see feature\_test\_macros(7)):

vfork(): \_BSD\_SOURCE || \_XOPEN\_SOURCE >= 500

可以将fork改成与vfork相近的功能

pid = fork();

if (pid == 0)

{

sleep (2);

printf ("This is child process\n");

exit (0);

}

else if (pid > 0)

{

wait (NULL);//父进程中加入此句就能实现vfork功能

printf ("This is parent process\n");

exit (0);

}

## 进程退出exit

NAME

\_exit, \_Exit - terminate the calling process

SYNOPSIS

#include <unistd.h>

void \_exit(int status);

#include <stdlib.h>

void \_Exit(int status);

void exit (int status);

RETURN VALUE

These functions do not return.

## 进程等待wait

NAME

wait, waitpid, waitid - wait for process to change state

SYNOPSIS

#include <sys/types.h>

#include <sys/wait.h>

pid\_t wait(int \*status);

pid\_t waitpid(pid\_t pid, int \*status, int options);

int waitid(idtype\_t idtype, id\_t id, siginfo\_t \*infop, int options);

Feature Test Macro Requirements for glibc (see feature\_test\_macros(7)):

waitid(): \_SVID\_SOURCE || \_XOPEN\_SOURCE

RETURN VALUE

wait(): on success, returns the process ID of the terminated child; on

error, -1 is returned.

waitpid(): on success, returns the process ID of the child whose state

has changed; if WNOHANG was specified and one or more child(ren) speci-

fied by pid exist, but have not yet changed state, then 0 is returned.

On error, -1 is returned.

waitid(): returns 0 on success or if WNOHANG was specified and no

child(ren) specified by id has yet changed state; on error, -1 is

returned. Each of these calls sets errno to an appropriate value in

the case of an error.

## 执行程序exec

NAME

execl, execlp, execle, execv, execvp - execute a file

SYNOPSIS

#include <unistd.h>

extern char \*\*environ;

int execl(const char \*path, const char \*arg, ...);

int execlp(const char \*file, const char \*arg, ...);

int execle(const char \*path, const char \*arg,

..., char \* const envp[]);

int execv(const char \*path, char \*const argv[]);

int execvp(const char \*file, char \*const argv[]);

RETURN VALUE

If any of the exec() functions returns, an error will have occurred.

The return value is -1, and errno will be set to indicate the error.

用法:execvp():模拟执行真实命令

#include <stdio.h>

#include <unistd.h>

int main (int argc, char \*argv[])

{

if (argc < 2)

{

printf ("Usage: %s %s %s %s\n", argv[0], "command","args","...");

\_exit (0);

}

execvp (argv[1],&argv[1]);//argv[1]为要执行的文件,&argv[1]为1处的地址.

printf ("hhhhhhhhhhhhhhhhhhh\n");

return 0;

}

用法:execlp():根据全局变量找到可执行文件名的路径并执行该可执行程序

#include <stdio.h>

#include <unistd.h>

int main (int argc, char \*argv[])

{

if (argc != 2)

{

printf ("Usage: %s %s\n", argv[0], "ls");

\_exit (1);

}

execlp (argv[1], "-l", "-h", "-s", NULL);

return 0;

}

## exec()函数和system()函数的区别

execl()执行后会把可执行文件ls的代码段覆盖原有进程的代码段。而进程还是原来进程，执行完ls之后就直接退出。[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

system()执行完命令之后返回到原有代码段继续执行下面代码。

# 第13课-无名管道通讯编程

通讯：Linux作为一个多进程的操作系统，进程和进程之间是否也需要通讯呢？

## 进程通讯目的

数据传输:一个进程需要将数据发送给另一个进程。

资源共享:多个进程之间共享同样的资源。

通知事件:一个进程需要向另一个/组进程发送消息，通知它们发生了某事件。

进程控制:有些进程希望完全控制另一个进程的执行(如debug进程)，此时控制进程希望能够拦截另一个进程的所有操作，并能够及时知道它的状态改变。

进程通讯发展史

Linux进程间通信(IPC:interprocess communication)由下面3部分发展而来:

UNIX进程间通信,基于systemV进程间通信,POSIX进程间通信。

POSIX(Portable Operating System Interface)

表示可移植操作系统接口。电气和电子工程师协会IEEE最初开发POSIX标准，是为了提高UNIX环境下应用程序的可移植性。然而，POSIX并不局限于UNIX，许多其他的OS，例如Microsoft windows都支持POSIX标准。

IPC的几种方式

无名管道(pipe),有名管道(fifo),信号(signal),消息队列(msg),共享内存(shm),信号量(),套接字(socket)

管道通信

一个进程在管道的尾部写入数据,另一个进程从管道头部读出数据[01==读写]。管道包含无名管道和有名管道两种,前者只能用于父进程和子进程间的通信,后者可用于运行于同一系统中的任意两个进程间的通信。

管道通信特点

\*管道通信是单向的,有固定读端和写端。

\*数据被进程从管道读出后,在管道中该数据就不存在了。

\*当进程去读取空管道的时候,进程会阻塞。

\*当进程往满管道写入数据时,进程会阻塞。

\*管道容量为64KB(#define PIPE\_BUFFERS 16 include/linux/pipe\_fs\_i.h)

## 创建管道:pipe()

NAME[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

pipe, pipe2 - create pipe

SYNOPSIS

#include <unistd.h>

int pipe(int pipefd[2]);

#define \_GNU\_SOURCE

#include <unistd.h>

int pipe2(int pipefd[2], int flags);

DESCRIPTION

pipe() creates a pipe, a unidirectional data channel that can be used

for interprocess communication. The array pipefd is used to return two

file descriptors referring to the ends of the pipe. pipefd[0] refers

to the read end of the pipe. pipefd[1] refers to the write end of the

pipe. Data written to the write end of the pipe is buffered by the

kernel until it is read from the read end of the pipe. For further

details, see pipe(7).

If flags is 0, then pipe2() is the same as pipe(). The following val-

ues can be bitwise ORed in flags to obtain different behavior:

O\_NONBLOCK Set the O\_NONBLOCK file status flag on the two new open

file descriptions. Using this flag saves extra calls to

fcntl(2) to achieve the same result.

O\_CLOEXEC Set the close-on-exec (FD\_CLOEXEC) flag on the two new file

descriptors. See the description of the same flag in

open(2) for reasons why this may be useful.

RETURN VALUE

On success, zero is returned. On error, -1 is returned, and errno is

set appropriately.

ERRORS

EFAULT pipefd is not valid.

EINVAL (pipe2()) Invalid value in flags.

EMFILE Too many file descriptors are in use by the process.

ENFILE The system limit on the total number of open files has been

reached.

## 有名管道pipe实验代码

//文件名: pipe.c

//创建者: seafly

//创建时间: 2016/11/16

//修改者: seafly

//修改时间: 2016/11/16

//文件功能: 有名管道的实验代码

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <errno.h>

#include <fcntl.h>

#include <sys/types.h>

#include <unistd.h>

int main (void)

{

int pp[2] = {0};

pid\_t pid = 0;

char buf[10] = "abcde1234\0";

pipe (pp);//创建无名管道

pid = fork();//创建子进程之前创建无名管道

if (pid > 0)

{

write (pp[1], buf, sizeof(buf));

close(pp[1]);

memset(buf, 0, sizeof(buf));

printf("Parent write into pipe\n");

printf("memset buf...\n");

exit(0);

}

else if (pid == 0)

{

wait(NULL);

read (pp[0], buf, sizeof(buf));

printf ("buf: %s\n", buf);

close(pp[0]);

}

return 0;

}

# 第14课-有名管道通讯编程

## 核心理论

有名管道

有名管道又称为FIFO文件,因此我们对有名管道的操作可以采用操作文件的方法来操作管道。

FIFO文件对比普通文件操作

\*读取FIFO文件的进程只能以RDONLY方式打开FIFO文件。

\*写入FIFO文件的进程只能以WRONLY方式打开FIFO文件。

\*FIFO文件里面的内容被读取后,就消失了。但普通文件里面的内容读取后还存在。

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

## 创建有名管道(创建fifo文件)

NAME

mkfifo - make a FIFO special file (a named pipe)

SYNOPSIS

#include <sys/types.h>

#include <sys/stat.h>

int mkfifo(const char \*pathname, mode\_t mode);

DESCRIPTION

mkfifo() makes a FIFO special file with name pathname. mode specifies

the FIFO’s permissions. It is modified by the process’s umask in the

usual way: the permissions of the created file are (mode & ~umask).

A FIFO special file is similar to a pipe, except that it is created in

a different way. Instead of being an anonymous communications channel,

a FIFO special file is entered into the file system by calling

mkfifo().

Once you have created a FIFO special file in this way, any process can

open it for reading or writing, in the same way as an ordinary file.

However, it has to be open at both ends simultaneously before you can

proceed to do any input or output operations on it. Opening a FIFO for

reading normally blocks until some other process opens the same FIFO

for writing, and vice versa. See fifo(7) for non-blocking handling of

FIFO special files.

RETURN VALUE

On success mkfifo() returns 0. In the case of an error, -1 is returned

(in which case, errno is set appropriately).

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

## 删除有名管道(删除fifo文件)

NAME

unlink - delete a name and possibly the file it refers to

SYNOPSIS

#include <unistd.h>

int unlink(const char \*pathname);

DESCRIPTION

unlink() deletes a name from the file system. If that name was the

last link to a file and no processes have the file open the file is

deleted and the space it was using is made available for reuse.

If the name was the last link to a file but any processes still have

the file open the file will remain in existence until the last file

descriptor referring to it is closed.

If the name referred to a symbolic link the link is removed.

If the name referred to a socket, fifo or device the name for it is

removed but processes which have the object open may continue to use

it.

RETURN VALUE

On success, zero is returned. On error, -1 is returned, and errno is

set appropriately.

当我们写进程运行时，如果没有读进程去读取，写进程将处于阻塞状态。

## 写有名管道(写fifo文件):wrfifo.c

核心代码

int main (void)

{

int fd = 0;

char buf[10] = "abc123def\0";

mkfifo("/root/Desktop/fifo.dat", 0666);

fd = open ("/root/Desktop/fifo.dat", O\_WRONLY);

write (fd, buf, sizeof(buf));

printf ("Writting into successfully!\n");

close (fd);

return 0;

}[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

## 读有名管道(读fifo文件):rdfifo.c

核心代码

int main (void)

{

int fd = 0;

char buf[10] = {0};

fd = open ("/root/Desktop/fifo.dat", O\_RDONLY);

if (0>fd) //判断是否有对应的fifo文件

{

printf ("Open fifo failling!\n");

close (fd);

exit (1);

}

read (fd, buf, sizeof(buf));

if ('\0' == buf[0]) //判断字符串变量是否为空

{

printf ("Reading out failling!\n");

close (fd);

exit (1);

}

printf ("buf: %s\n", buf);

printf ("Reading out successfully!\n");

close (fd);

unlink ("/root/Desktop/fifo.dat"); //文件操作函数:删除文件

return 0;

}

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

# 第15课-信号通讯编程

## 核心理论

**信号：像长城烽火台一样，在Linux系统中，信号同样也是最为古老的进程间通信机制。**

**信号处理流程：选择信号》》发送信号》》处理信号。**

**信号类型(/usr/include/asm/signal.h)**

**\*SIGKILL》》杀死进程 \*SIGSTOP》》暂停进程 \*SIGCHLD》》子进程停止或者结束时用来通知父进程。**

## 发送信号:kill()

NAME

kill - send signal to a process

SYNOPSIS

#include <sys/types.h>

#include <signal.h>

int kill(pid\_t pid, int sig);

Feature Test Macro Requirements for glibc (see feature\_test\_macros(7)):

kill(): \_POSIX\_C\_SOURCE >= 1 || \_XOPEN\_SOURCE || \_POSIX\_SOURCE

RETURN VALUE

On success (at least one signal was sent), zero is returned. On error,

-1 is returned, and errno is set appropriately.

NAME

atoi, atol, atoll, atoq - convert a string to an integer

SYNOPSIS

#include <stdlib.h>

int atoi(const char \*nptr);

long atol(const char \*nptr);

long long atoll(const char \*nptr);

long long atoq(const char \*nptr);

Feature Test Macro Requirements for glibc (see feature\_test\_macros(7)):

atoll(): \_BSD\_SOURCE || \_SVID\_SOURCE || \_XOPEN\_SOURCE >= 600 ||

\_ISOC99\_SOURCE; or cc -std=c99

## 处理信号:signal()

NAME

signal - ANSI C signal handling

SYNOPSIS

#include <signal.h>

typedef void (\*sighandler\_t)(int);

sighandler\_t signal(int signum, sighandler\_t handler);

void ( \*signal(int signum, void (\*handler)(int)) ) (int);

RETURN VALUE

signal() returns the previous value of the signal handler, or SIG\_ERR

on error.

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

## 等待信号:pause()

NAME

pause - wait for signal

SYNOPSIS

#include <unistd.h>

int pause(void);

DESCRIPTION

pause() causes the calling process (or thread) to sleep until a signal

is delivered that either terminates the process or causes the invoca-

tion of a signal-catching function.

RETURN VALUE

pause() only returns when a signal was caught and the signal-catching

function returned. In this case pause() returns -1, and errno is set

to EINTR.

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

## 实验核心代码

## 信号发送端kill.c

#define SIGSEAFLY 0x16 //自定义的一个信号类型

int main (int argc, char \*argv[])

{

if (3 != argc)

{

printf ("Usage: %s %s %s\n", argv[0], "pid", "signal");

exit (1);

}

pid\_t pid = 0;

int sig = 0;

pid = atoi (argv[1]);

if (0 == strcmp("SIGKILL", argv[2]))

sig = SIGKILL;

else if(0 == strcmp("SIGQUIT", argv[2]))

sig = SIGQUIT;

else if(0 == strcmp("SIGINT", argv[2]))

sig = SIGINT;

else if(0 == strcmp("SIGSEAFLY", argv[2]))

sig = SIGSEAFLY;

kill (pid, sig);

return 0;

}

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

## 信号接收处理端:signal.c

#define SIGSEAFLY 0x16 //与信号发送端相对应的自定义信号类型

void func (int sig)

{

if (SIGINT == sig)

{

printf ("Get SIGINT signal, exit()...\n");

exit(0);

}

else if (SIGQUIT == sig)

{

printf ("Get SIGQUIT signal, exit()...\n");

exit(0);

}

else if (SIGSEAFLY == sig)

{

printf ("Get SIGSEAFLY signal, exit()...\n");

exit(0);

}

else

{

printf ("Get Unkown signal, exit()...\n");

exit(1);

}

return ;

}

int main (void)

{

int sig = 0;

printf ("PID: %d\n", getpid());

signal (SIGINT, func);

signal (SIGQUIT, func);

signal (SIGSEAFLY, func);

pause();//等待信号处理

return 0;

}

# 第16课-信号量互斥编程

[主目录](file:///C:\\Users\\Administrator\\Desktop\\专业体系2-精通嵌入式.docx" \l "_目录)

## 核心理论

公示栏问题

同学1:写入“数学课” + 睡眠数秒 + 写入“取消”

同学2:写入“英语课” + 写入“考试”

互斥问题:数学课英语课考试取消(信息混乱),同时访问某一资源造成数据的混乱。

信号量概念

信号量(又名信号灯),主要作用是保护临街资源(进程互斥)。进程可以根据它来判定是否能够访问某些共享资源。除了用于访问控制外，还可用于进程同步。信号量的实质就是一个数字，获取信号量就是将数字减1,释放信号量就是将数字+1

信号量分类

分为二值信号量(信号灯的值只能取0或1)和计数信号量(信号灯的值可以取任意非负值)

当发现信号灯是1时可以用，用的时候把信号灯改为0以保护临界资源。

## 获取/构造键值:ftok()

指定键值

\*任意指定一个数，缺点这个数很可能已经被别的IPC对象(消息队列,共享内存)使用了，在与新创建的信号量关联时就会失败。

\*构造一个尽量不会被别的IPC对象用到的数字，使用ftok()函数来构造一个新的键值。

键值构建原理

通过文件名对应的数字+项目ID ==》键值

AME

ftok - convert a pathname and a project identifier to a System V IPC

key

SYNOPSIS

#include <sys/types.h>

#include <sys/ipc.h>

key\_t ftok(const char \*pathname, int proj\_id);

RETURN VALUE

On success the generated key\_t value is returned. On failure -1 is

returned, with errno indicating the error as for the stat(2) system

call.

## 创建/获取信号量:semget()

NAME

semget - get a semaphore set identifier

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

SYNOPSIS

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/sem.h>

int semget(key\_t key, int nsems, int semflg);

//nsems:获取的时候要你生成一个多大容量信号量的集合

//semflg:一般填写IPC\_CREAT

RETURN VALUE

If successful, the return value will be the semaphore set identifier (a

non-negative integer), otherwise -1 is returned, with errno indicating

the error.

## 操作信号量:semop()

NAME

semop, semtimedop - semaphore operations

SYNOPSIS

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/sem.h>

int semop(int semid, struct sembuf \*sops, unsigned nsops);

//semop():信号量操作函数

//nsops:当前函数要操作多少个信号量（信号灯）

DESCRIPTION

Each semaphore in a semaphore set has the following associated values:

unsigned short semval; /\* semaphore value \*/

unsigned short semzcnt; /\* # waiting for zero \*/

unsigned short semncnt; /\* # waiting for increase \*/

pid\_t sempid; /\* process that did last op \*/

semop() performs operations on selected semaphores in the set indicated

by semid. Each of the nsops elements in the array pointed to by sops

specifies an operation to be performed on a single semaphore. The ele-

ments of this structure are of type struct sembuf, containing the fol-

lowing members:

unsigned short sem\_num; /\* semaphore number \*/

short sem\_op; /\* semaphore operation \*/

//当op为正数时,代表释放信号量

//当op为负数时,代表获取信号量,如果获取不到时,会导致该程序阻塞等待。

short sem\_flg; /\* operation flags \*/

Flags recognized in sem\_flg are IPC\_NOWAIT and SEM\_UNDO. If an opera-

tion specifies SEM\_UNDO, it will be automatically undone when the pro-

cess terminates.

The set of operations contained in sops is performed in array order,

and atomically, that is, the operations are performed either as a com-

plete unit, or not at all. The behavior of the system call if not all

operations can be performed immediately depends on the presence of the

IPC\_NOWAIT flag in the individual sem\_flg fields, as noted below.

RETURN VALUE

If successful semop() and semtimedop() return 0; otherwise they return

-1 with errno indicating the error.

EXAMPLE

The following code segment uses semop() to atomically wait for the value of

semaphore 0 to become zero, and then increment the semaphore value by one.

struct sembuf sops[2];

int semid;

/\* Code to set semid omitted \*/

sops[0].sem\_num = 0; /\* Operate on semaphore 0 \*/

sops[0].sem\_op = 0; /\* Wait for value to equal 0 \*/

sops[0].sem\_flg = 0;

sops[1].sem\_num = 0; /\* Operate on semaphore 0 \*/

sops[1].sem\_op = 1; /\* Increment value by one \*/

sops[1].sem\_flg = 0;

if (semop(semid, sops, 2) == -1) {

perror("semop");

exit(EXIT\_FAILURE);

}

## 控制信号量:semctl()

NAME

semctl - semaphore control operations

SYNOPSIS

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/sem.h>

int semctl(int semid, int semnum, int cmd, ...);

IPC\_RMID Immediately remove the semaphore set, awakening all processes

blocked in semop(2) calls on the set (with an error return

and errno set to EIDRM). The effective user ID of the call-

ing process must match the creator or owner of the semaphore

set, or the caller must be privileged. The argument semnum

is ignored.//通过该参数来删除信号量

GETVAL The system call returns the value of semval for the semnum-th

semaphore of the set. The calling process must have read

permission on the semaphore set.

SETVAL Set the value of semval to arg.val for the semnum-th

semaphore of the set, updating also the sem\_ctime member of

the semid\_ds structure associated with the set. Undo entries

are cleared for altered semaphores in all processes. If the

changes to semaphore values would permit blocked semop(2)

calls in other processes to proceed, then those processes are

woken up. The calling process must have alter permission on

the semaphore set.

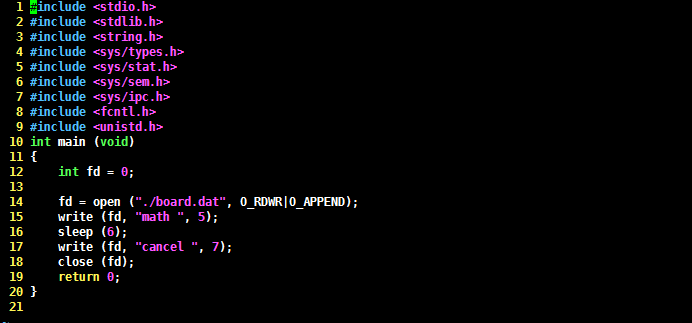
RETURN VALUE

On failure semctl() returns -1 with errno indicating the error.

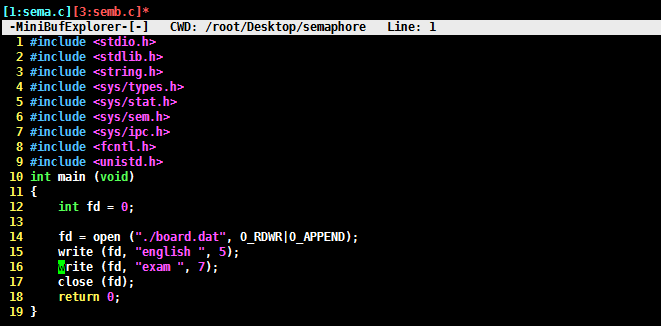
## 信号量互斥实验核心代码

编码要素，创建信号量的那个进程在获取信号量之前必须对其初始化，也就是通过SETVAL对其赋值1，然后在获取信号量（-1）操作，资源访问完毕就计时释放信号量（+1）操作。

公示栏问题:A同学:数学课取消:sema.c

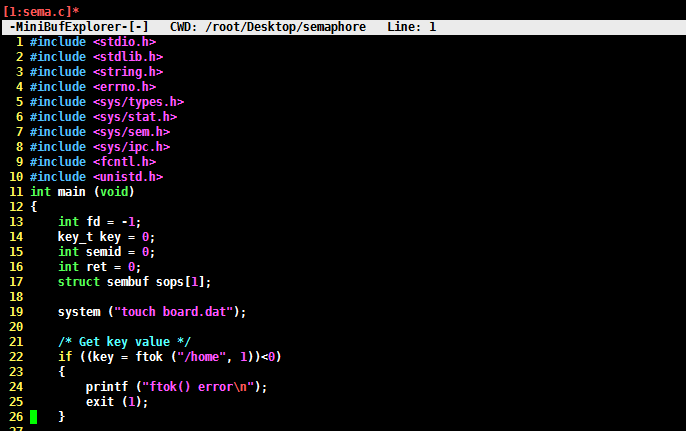


公示栏问题:B同学:英语考试:semb.c

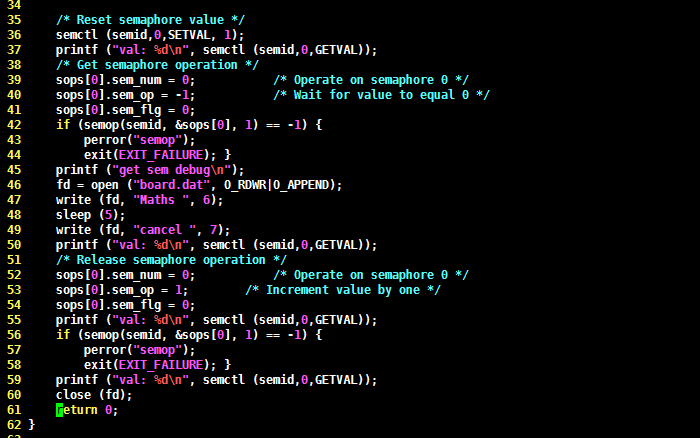


上面代码就是为A,B同学打好框架代码，他们的任务就像上面代码一样，分工明确，问题来了，在A睡眠的时候B不会等A写完了去写，而是直接在A睡觉的时候去写英语考试，所以最终造成的公告栏信息混乱问题：“数学英语考试取消”，下面代码是解决资源使用冲突情况下用信号量来控制，然后正确的公告栏信息是：“数学取消英语考试”。

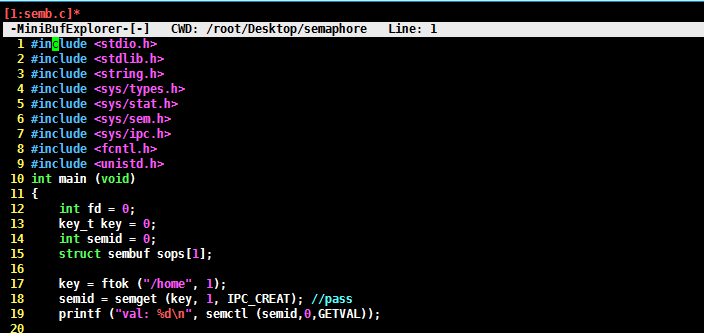
## A同学：数学+睡觉+取消

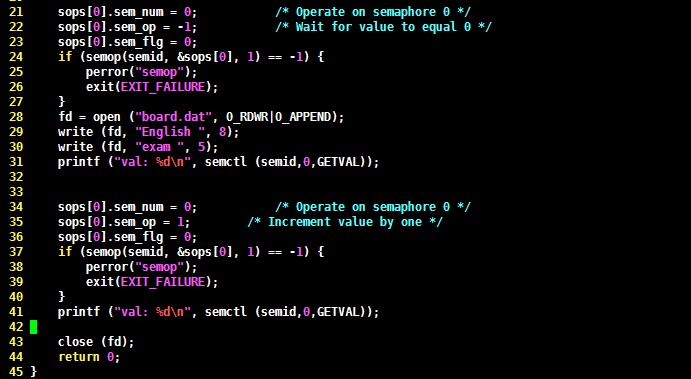






## B同学：英语+考试





# 第17课-信号量同步编程

## 核心理论

进程同步

一组并发进程进行互相合作、互相等待，使得各个进程按一定顺序执行的过程称为进程间的同步。

生产者与消费者问题

生产者生产出产品(释放信号量)》》消费者消费产品(获取信号量)

## 创建/获取信号量:semget()

NAME

semget - get a semaphore set identifier

[主目录](file:///C:UsersAdministratorDesktop专业体系2-精通嵌入式.docx" \l "_目录)

SYNOPSIS

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/sem.h>

int semget(key\_t key, int nsems, int semflg);

DESCRIPTION

The semget() system call returns the semaphore set identifier associ-

ated with the argument key. A new set of nsems semaphores is created

if key has the value IPC\_PRIVATE or if no existing semaphore set is

associated with key and IPC\_CREAT is specified in semflg.

If semflg specifies both IPC\_CREAT and IPC\_EXCL and a semaphore set

already exists for key, then semget() fails with errno set to EEXIST.

(This is analogous to the effect of the combination O\_CREAT | O\_EXCL

for open(2).)

RETURN VALUE

If successful, the return value will be the semaphore set identifier (a

non-negative integer), otherwise -1 is returned, with errno indicating

the error.

## 操作信号量:semop()

NAME

semop, semtimedop - semaphore operations

SYNOPSIS

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/sem.h>

int semop(int semid, struct sembuf \*sops, unsigned nsops);

DESCRIPTION

Each semaphore in a semaphore set has the following associated values:

unsigned short semval; /\* semaphore value \*/

unsigned short semzcnt; /\* # waiting for zero \*/

unsigned short semncnt; /\* # waiting for increase \*/

pid\_t sempid; /\* process that did last op \*/

semop() performs operations on selected semaphores in the set indicated

by semid. Each of the nsops elements in the array pointed to by sops

specifies an operation to be performed on a single semaphore. The ele-

ments of this structure are of type struct sembuf, containing the fol-

lowing members:

unsigned short sem\_num; /\* semaphore number \*/

short sem\_op; /\* semaphore operation \*/

//当op为正数时,代表释放信号量

//当op为负数时,代表获取信号量,如果获取不到时,会导致该程序阻塞等待。

short sem\_flg; /\* operation flags \*/

Flags recognized in sem\_flg are IPC\_NOWAIT and SEM\_UNDO. If an opera-

tion specifies SEM\_UNDO, it will be automatically undone when the pro-

cess terminates.

The set of operations contained in sops is performed in array order,

and atomically, that is, the operations are performed either as a com-

plete unit, or not at all. The behavior of the system call if not all

operations can be performed immediately depends on the presence of the

IPC\_NOWAIT flag in the individual sem\_flg fields, as noted below.

Each operation is performed on the sem\_num-th semaphore of the

semaphore set, where the first semaphore of the set is numbered 0.

There are three types of operation, distinguished by the value of

sem\_op.

If sem\_op is a positive integer, the operation adds this value to the

semaphore value (semval). Furthermore, if SEM\_UNDO is specified for

this operation, the system updates the process undo count (semadj) for

this semaphore. This operation can always proceed — it never forces a

process to wait. The calling process must have alter permission on the

semaphore set.

If sem\_op is zero, the process must have read permission on the

semaphore set. This is a "wait-for-zero" operation: if semval is zero,

the operation can immediately proceed. Otherwise, if IPC\_NOWAIT is

specified in sem\_flg, semop() fails with errno set to EAGAIN (and none

of the operations in sops is performed). Otherwise semzcnt (the count

of processes waiting until this semaphore’s value becomes zero) is

incremented by one and the process sleeps until one of the following

occurs:

RETURN VALUE

If successful semop() and semtimedop() return 0; otherwise they return

-1 with errno indicating the error.

EXAMPLE

The following code segment uses semop() to atomically wait for the value of

semaphore 0 to become zero, and then increment the semaphore value by one.

struct sembuf sops[2];

int semid;

/\* Code to set semid omitted \*/

sops[0].sem\_num = 0; /\* Operate on semaphore 0 \*/

sops[0].sem\_op = 0; /\* Wait for value to equal 0 \*/

sops[0].sem\_flg = 0;

sops[1].sem\_num = 0; /\* Operate on semaphore 0 \*/

sops[1].sem\_op = 1; /\* Increment value by one \*/

sops[1].sem\_flg = 0;

if (semop(semid, sops, 2) == -1) {

perror("semop");

exit(EXIT\_FAILURE);

}

## 控制信号量:semctl()

NAME

semctl - semaphore control operations

SYNOPSIS

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/sem.h>

int semctl(int semid, int semnum, int cmd, ...);

Valid values for cmd are:

IPC\_STAT Copy information from the kernel data structure associated

with semid into the semid\_ds structure pointed to by arg.buf.

The argument semnum is ignored. The calling process must

have read permission on the semaphore set.

IPC\_SET Write the values of some members of the semid\_ds structure

pointed to by arg.buf to the kernel data structure associated

with this semaphore set, updating also its sem\_ctime member.

The following members of the structure are updated:

sem\_perm.uid, sem\_perm.gid, and (the least significant 9 bits

of) sem\_perm.mode. The effective UID of the calling process

must match the owner (sem\_perm.uid) or creator

(sem\_perm.cuid) of the semaphore set, or the caller must be

privileged. The argument semnum is ignored.

IPC\_RMID Immediately remove the semaphore set, awakening all processes

blocked in semop(2) calls on the set (with an error return

and errno set to EIDRM). The effective user ID of the call-

ing process must match the creator or owner of the semaphore

set, or the caller must be privileged. The argument semnum

is ignored.//通过该参数来删除信号量

IPC\_INFO (Linux-specific)

Returns information about system-wide semaphore limits and

parameters in the structure pointed to by arg.\_\_buf. This

structure is of type seminfo, defined in <sys/sem.h> if the

\_GNU\_SOURCE feature test macro is defined:

GETALL Return semval (i.e., the current value) for all semaphores of

the set into arg.array. The argument semnum is ignored. The

calling process must have read permission on the semaphore

set.

GETNCNT The system call returns the value of semncnt (i.e., the num-

ber of processes waiting for the value of this semaphore to

increase) for the semnum-th semaphore of the set (i.e., the

number of processes waiting for an increase of semval for the

semnum-th semaphore of the set). The calling process must

have read permission on the semaphore set.

GETPID The system call returns the value of sempid for the semnum-th

semaphore of the set (i.e., the PID of the process that exe-

cuted the last semop(2) call for the semnum-th semaphore of

the set). The calling process must have read permission on

the semaphore set.

GETVAL The system call returns the value of semval for the semnum-th

semaphore of the set. The calling process must have read

permission on the semaphore set.

GETZCNT The system call returns the value of semzcnt (i.e., the num-

ber of processes waiting for the value of this semaphore to

become zero) for the semnum-th semaphore of the set (i.e.,

the number of processes waiting for semval of the semnum-th

semaphore of the set to become 0). The calling process must

have read permission on the semaphore set.

SETALL Set semval for all semaphores of the set using arg.array,

updating also the sem\_ctime member of the semid\_ds structure

associated with the set. Undo entries (see semop(2)) are

cleared for altered semaphores in all processes. If the

changes to semaphore values would permit blocked semop(2)

calls in other processes to proceed, then those processes are

woken up. The argument semnum is ignored. The calling pro-

cess must have alter (write) permission on the semaphore set.

SETVAL Set the value of semval to arg.val for the semnum-th

semaphore of the set, updating also the sem\_ctime member of

the semid\_ds structure associated with the set. Undo entries

are cleared for altered semaphores in all processes. If the

changes to semaphore values would permit blocked semop(2)

calls in other processes to proceed, then those processes are

woken up. The calling process must have alter permission on

the semaphore set.

RETURN VALUE

On failure semctl() returns -1 with errno indicating the error.

Otherwise the system call returns a non-negative value depending on cmd

as follows:

GETNCNT the value of semncnt.

GETPID the value of sempid.

GETVAL the value of semval.

GETZCNT the value of semzcnt.

IPC\_INFO the index of the highest used entry in the kernel’s inter-

nal array recording information about all semaphore sets.

(This information can be used with repeated SEM\_STAT opera-

tions to obtain information about all semaphore sets on the

system.)

SEM\_INFO As for IPC\_INFO.

SEM\_STAT the identifier of the semaphore set whose index was given

in semid.

All other cmd values return 0 on success.

## 信号量同步操作流程

生产者: 创建/获取信号量 》》初始化信号量op为0》》生产产品 》》释放信号量

消费者: 创建/获取信号量 》》 获取信号量 》》 消费产品

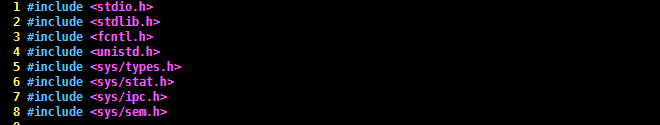
生产者首先初始化信号量集合，并初始化值为0，当该产品生产完之后就可以释放释放信号量（+1操作），然后B就从阻塞状态获得信号量来获得产品的成品。

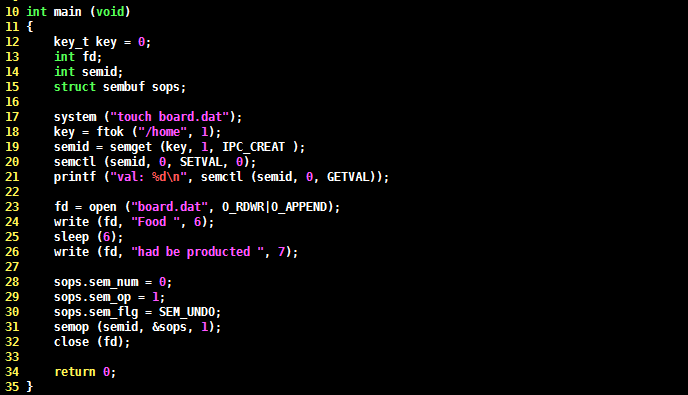
信号量同步和信号量互斥区别：

信号量同步中，生产者只负责释放信号量（持续+1操作），消费者只负责获取信号量（持续-1操作），但生产者在释放信号量之前必须初始化VAL为0，从而让消费者阻塞等待生产者把成品生产出来之后，再去获取成品。

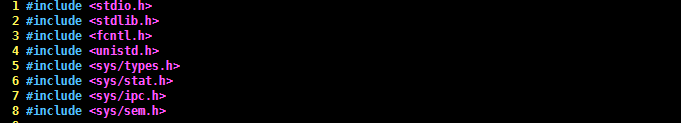
## 信号量同步实验核心代码

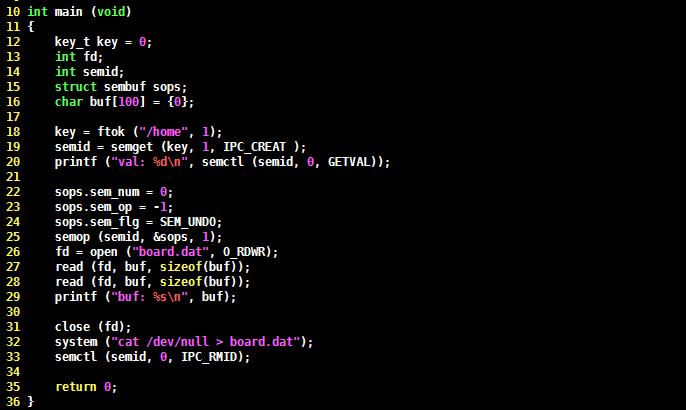
生产者生产食物范例代码：





消费者消费食物范例代码：





# 第二季-小应用大智慧(Linux应用编程)-下学期

# 第18课-共享内存通讯

[主目录](#_目录)

## 核心理论

共享内存基本概念

共享内存是IPC机制中的一种。它允许两个不相关的进程访问同一段内存，这是传递数据的一种非常有效的方式。

共享内存编程步骤

创建共享内存 》》映射共享内存 》》分离共享内存 》》控制(删除)共享内存

## 创建共享内存键值:ftok()

NAME

ftok - convert a pathname and a project identifier to a System V IPC key

SYNOPSIS

#include <sys/types.h>

#include <sys/ipc.h>

key\_t ftok(const char \*pathname, int proj\_id);

DESCRIPTION

The ftok() function uses the identity of the file named by the given pathname

(which must refer to an existing, accessible file) and the least significant 8

bits of proj\_id (which must be non-zero) to generate a key\_t type System V IPC

key, suitable for use with msgget(2), semget(2), or shmget(2).

The resulting value is the same for all pathnames that name the same file, when

the same value of proj\_id is used. The value returned should be different when

the (simultaneously existing) files or the project IDs differ.

RETURN VALUE

On success the generated key\_t value is returned. On failure -1 is returned,

with errno indicating the error as for the stat(2) system call.

## 创建共享内存:shmget()

NAME

shmget - allocates a shared memory segment

SYNOPSIS

#include <sys/ipc.h>

#include <sys/shm.h>

int shmget(key\_t key, size\_t size, int shmflg);

DESCRIPTION

shmget() returns the identifier of the shared memory segment associated

with the value of the argument key. A new shared memory segment, with

size equal to the value of size rounded up to a multiple of PAGE\_SIZE,

is created if key has the value IPC\_PRIVATE or key isn’t IPC\_PRIVATE,

no shared memory segment corresponding to key exists, and IPC\_CREAT is

specified in shmflg.

If shmflg specifies both IPC\_CREAT and IPC\_EXCL and a shared memory

segment already exists for key, then shmget() fails with errno set to

EEXIST. (This is analogous to the effect of the combination O\_CREAT |

O\_EXCL for open(2).)

The value shmflg is composed of:

IPC\_CREAT to create a new segment. If this flag is not used, then

shmget() will find the segment associated with key and

check to see if the user has permission to access the seg-

ment.

IPC\_EXCL used with IPC\_CREAT to ensure failure if the segment

already exists.

RETURN VALUE

A valid segment identifier, shmid, is returned on success, -1 on error.

## 映射共享内存:shmat()

NAME

shmat, shmdt - shared memory operations

SYNOPSIS

#include <sys/types.h>

#include <sys/shm.h>

void \*shmat(int shmid, const void \*shmaddr, int shmflg);

int shmdt(const void \*shmaddr);

DESCRIPTION

shmat() attaches the shared memory segment identified by shmid to the

address space of the calling process. The attaching address is speci-

fied by shmaddr with one of the following criteria:

If shmaddr is NULL, the system chooses a suitable (unused) address at

which to attach the segment.

If shmaddr isn’t NULL and SHM\_RND is specified in shmflg, the attach

occurs at the address equal to shmaddr rounded down to the nearest mul-

tiple of SHMLBA. Otherwise shmaddr must be a page-aligned address at

which the attach occurs.

If SHM\_RDONLY is specified in shmflg, the segment is attached for read-

ing and the process must have read permission for the segment. Other-

wise the segment is attached for read and write and the process must

have read and write permission for the segment. There is no notion of

a write-only shared memory segment.

The (Linux-specific) SHM\_REMAP flag may be specified in shmflg to indi-

cate that the mapping of the segment should replace any existing map-

ping in the range starting at shmaddr and continuing for the size of

the segment. (Normally an EINVAL error would result if a mapping

already exists in this address range.) In this case, shmaddr must not

be NULL.

RETURN VALUE

On success shmat() returns the address of the attached shared memory

segment; on error (void \*) -1 is returned, and errno is set to indicate

the cause of the error.

On success shmdt() returns 0; on error -1 is returned, and errno is set

to indicate the cause of the error

## 分离共享内存:shmdt()

NAME

shmat, shmdt - shared memory operations

SYNOPSIS

#include <sys/types.h>

#include <sys/shm.h>

void \*shmat(int shmid, const void \*shmaddr, int shmflg);

int shmdt(const void \*shmaddr);

DESCRIPTION

shmat() attaches the shared memory segment identified by shmid to the

address space of the calling process. The attaching address is speci-

fied by shmaddr with one of the following criteria:

If shmaddr is NULL, the system chooses a suitable (unused) address at

which to attach the segment.

If shmaddr isn’t NULL and SHM\_RND is specified in shmflg, the attach

occurs at the address equal to shmaddr rounded down to the nearest mul-

tiple of SHMLBA. Otherwise shmaddr must be a page-aligned address at

which the attach occurs.

If SHM\_RDONLY is specified in shmflg, the segment is attached for read-

ing and the process must have read permission for the segment. Other-

wise the segment is attached for read and write and the process must

have read and write permission for the segment. There is no notion of

a write-only shared memory segment.

The (Linux-specific) SHM\_REMAP flag may be specified in shmflg to indi-

cate that the mapping of the segment should replace any existing map-

ping in the range starting at shmaddr and continuing for the size of

the segment. (Normally an EINVAL error would result if a mapping

already exists in this address range.) In this case, shmaddr must not

be NULL.

RETURN VALUE

On success shmat() returns the address of the attached shared memory

segment; on error (void \*) -1 is returned, and errno is set to indicate

the cause of the error.

On success shmdt() returns 0; on error -1 is returned, and errno is set

to indicate the cause of the error

## 控制(删除)共享内存:shmctl()

NAME

shmctl - shared memory control

SYNOPSIS

#include <sys/ipc.h>

#include <sys/shm.h>

int shmctl(int shmid, int cmd, struct shmid\_ds \*buf);

DESCRIPTION

shmctl() performs the control operation specified by cmd on the shared

memory segment whose identifier is given in shmid.

The buf argument is a pointer to a shmid\_ds structure, defined in

<sys/shm.h> as follows:

struct shmid\_ds {

struct ipc\_perm shm\_perm; /\* Ownership and permissions \*/

size\_t shm\_segsz; /\* Size of segment (bytes) \*/

time\_t shm\_atime; /\* Last attach time \*/

time\_t shm\_dtime; /\* Last detach time \*/

time\_t shm\_ctime; /\* Last change time \*/

pid\_t shm\_cpid; /\* PID of creator \*/

pid\_t shm\_lpid; /\* PID of last shmat(2)/shmdt(2) \*/

shmatt\_t shm\_nattch; /\* No. of current attaches \*/

...

};

Valid values for cmd are:

IPC\_STAT Copy information from the kernel data structure associated

with shmid into the shmid\_ds structure pointed to by buf.

The caller must have read permission on the shared memory

segment.

IPC\_SET Write the values of some members of the shmid\_ds structure

pointed to by buf to the kernel data structure associated

with this shared memory segment, updating also its shm\_ctime

member. The following fields can be changed: shm\_perm.uid,

shm\_perm.gid, and (the least significant 9 bits of)

shm\_perm.mode. The effective UID of the calling process must

match the owner (shm\_perm.uid) or creator (shm\_perm.cuid) of

the shared memory segment, or the caller must be privileged.

IPC\_RMID Mark the segment to be destroyed. The segment will only

actually be destroyed after the last process detaches it

(i.e., when the shm\_nattch member of the associated structure

shmid\_ds is zero). The caller must be the owner or creator,

or be privileged. If a segment has been marked for destruc-

tion, then the (non-standard) SHM\_DEST flag of the

shm\_perm.mode field in the associated data structure

retrieved by IPC\_STAT will be set.

The caller must ensure that a segment is eventually destroyed; other-

wise its pages that were faulted in will remain in memory or swap.

SHM\_LOCK (Linux-specific)

Prevent swapping of the shared memory segment. The caller

must fault in any pages that are required to be present after

locking is enabled. If a segment has been locked, then the

(non-standard) SHM\_LOCKED flag of the shm\_perm.mode field in

the associated data structure retrieved by IPC\_STAT will be

set.

SHM\_UNLOCK (Linux-specific)

Unlock the segment, allowing it to be swapped out.

RETURN VALUE

A successful IPC\_INFO or SHM\_INFO operation returns the index of the

highest used entry in the kernel’s internal array recording information

about all shared memory segments. (This information can be used with

repeated SHM\_STAT operations to obtain information about all shared

memory segments on the system.) A successful SHM\_STAT operation

returns the identifier of the shared memory segment whose index was

given in shmid. Other operations return 0 on success.

On error, -1 is returned, and errno is set appropriately.

## 共享内存实验核心代码

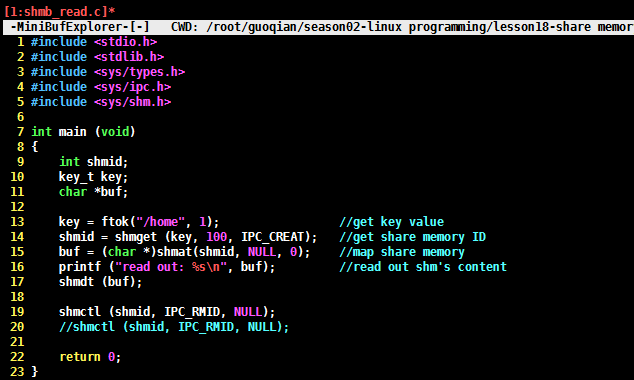
经过本人实验验证，共享内存有个特性，就是读走数据的那一端程序执行完毕后，这段共享内存不会自动清零。除非读走共享数据的那个进程主动去清空共享内存内容。

下面我们来看看两个进程范例代码：

进程A向共享内存内写入数据实验代码：



进程A向共享内存内写入数据实验代码：



# 第19课-消息队列编程

[主目录](#_目录)

## 核心理论

消息队列基本概念

消息队列就是一个消息的链表。而一条消息则可看作一个记录，具有特定格式。进程可以按照一定规则向内核中添加消息，另一些进程则可以从消息队列中读走消息。

## 创建/获取键值:ftok()

NAME

ftok - convert a pathname and a project identifier to a System V IPC

key

SYNOPSIS

#include <sys/types.h>

#include <sys/ipc.h>

key\_t ftok(const char \*pathname, int proj\_id);

DESCRIPTION

The ftok() function uses the identity of the file named by the given

pathname (which must refer to an existing, accessible file) and the

least significant 8 bits of proj\_id (which must be non-zero) to gener-

ate a key\_t type System V IPC key, suitable for use with msgget(2),

semget(2), or shmget(2).

The resulting value is the same for all pathnames that name the same

file, when the same value of proj\_id is used. The value returned

should be different when the (simultaneously existing) files or the

project IDs differ.

RETURN VALUE

On success the generated key\_t value is returned. On failure -1 is

returned, with errno indicating the error as for the stat(2) system

call.

## 创建/打开消息队列:msgget()

NAME

msgget - get a message queue identifier

SYNOPSIS

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

int msgget(key\_t key, int msgflg);

DESCRIPTION

The msgget() system call returns the message queue identifier associ-

ated with the value of the key argument. A new message queue is cre-

ated if key has the value IPC\_PRIVATE or key isn’t IPC\_PRIVATE, no mes-

sage queue with the given key key exists, and IPC\_CREAT is specified in

msgflg.

If msgflg specifies both IPC\_CREAT and IPC\_EXCL and a message queue

already exists for key, then msgget() fails with errno set to EEXIST.

(This is analogous to the effect of the combination O\_CREAT | O\_EXCL

for open(2).)

Upon creation, the least significant bits of the argument msgflg define

the permissions of the message queue. These permission bits have the

same format and semantics as the permissions specified for the mode

argument of open(2). (The execute permissions are not used.)

If a new message queue is created, then its associated data structure

msqid\_ds (see msgctl(2)) is initialized as follows:

RETURN VALUE

If successful, the return value will be the message queue identifier (a

non-negative integer), otherwise -1 with errno indicating the error.

## 发送和接收消息队列:msgsnd():msgrcv()

写入和读走消息:接收(读走)消息后队列则清空该消息了。

NAME

msgrcv, msgsnd - message operations

SYNOPSIS

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

int msgsnd(int msqid, const void \*msgp, size\_t msgsz, int msgflg);

ssize\_t msgrcv(int msqid, void \*msgp, size\_t msgsz, long msgtyp,

int msgflg);

DESCRIPTION

The msgsnd() and msgrcv() system calls are used, respectively, to send

messages to, and receive messages from, a message queue. The calling

process must have write permission on the message queue in order to

send a message, and read permission to receive a message.

The msgp argument is a pointer to caller-defined structure of the fol-

lowing general form:

struct msgbuf {

long mtype; /\* message type, must be > 0 \*/

char mtext[1]; /\* message data \*/

};

The mtext field is an array (or other structure) whose size is speci-

fied by msgsz, a non-negative integer value. Messages of zero length

(i.e., no mtext field) are permitted. The mtype field must have a

strictly positive integer value. This value can be used by the receiv-

ing process for message selection (see the description of msgrcv()

below).

msgsnd()

The msgsnd() system call appends a copy of the message pointed to by

msgp to the message queue whose identifier is specified by msqid.

If sufficient space is available in the queue, msgsnd() succeeds imme-

diately. (The queue capacity is defined by the msg\_bytes field in the

associated data structure for the message queue. During queue creation

this field is initialized to MSGMNB bytes, but this limit can be modi-

fied using msgctl(2).) If insufficient space is available in the

queue, then the default behavior of msgsnd() is to block until space

becomes available. If IPC\_NOWAIT is specified in msgflg, then the call

instead fails with the error EAGAIN.

msgrcv()

The msgrcv() system call removes a message from the queue specified by

msqid and places it in the buffer pointed to by msgp.

The argument msgsz specifies the maximum size in bytes for the member

mtext of the structure pointed to by the msgp argument. If the message

text has length greater than msgsz, then the behavior depends on

whether MSG\_NOERROR is specified in msgflg. If MSG\_NOERROR is speci-

fied, then the message text will be truncated (and the truncated part

will be lost); if MSG\_NOERROR is not specified, then the message isn’t

removed from the queue and the system call fails returning -1 with

errno set to E2BIG.

RETURN VALUE

On failure both functions return -1 with errno indicating the error,

otherwise msgsnd() returns 0 and msgrcv() returns the number of bytes

actually copied into the mtext array.

## 控制消息队列:msgctl()

NAME

msgctl - message control operations

SYNOPSIS

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

int msgctl(int msqid, int cmd, struct msqid\_ds \*buf);

DESCRIPTION

msgctl() performs the control operation specified by cmd on the message

queue with identifier msqid.

The msqid\_ds data structure is defined in <sys/msg.h> as follows:

struct msqid\_ds {

struct ipc\_perm msg\_perm; /\* Ownership and permissions \*/

time\_t msg\_stime; /\* Time of last msgsnd(2) \*/

time\_t msg\_rtime; /\* Time of last msgrcv(2) \*/

time\_t msg\_ctime; /\* Time of last change \*/

unsigned long \_\_msg\_cbytes; /\* Current number of bytes in

queue (non-standard) \*/

msgqnum\_t msg\_qnum; /\* Current number of messages

in queue \*/

msglen\_t msg\_qbytes; /\* Maximum number of bytes

allowed in queue \*/

pid\_t msg\_lspid; /\* PID of last msgsnd(2) \*/

pid\_t msg\_lrpid; /\* PID of last msgrcv(2) \*/

};

The ipc\_perm structure is defined in <sys/ipc.h> as follows (the high-

lighted fields are settable using IPC\_SET):

struct ipc\_perm {

key\_t \_\_key; /\* Key supplied to msgget(2) \*/

uid\_t uid; /\* Effective UID of owner \*/

gid\_t gid; /\* Effective GID of owner \*/

uid\_t cuid; /\* Effective UID of creator \*/

gid\_t cgid; /\* Effective GID of creator \*/

unsigned short mode; /\* Permissions \*/

unsigned short \_\_seq; /\* Sequence number \*/

};

Valid values for cmd are:

IPC\_STAT

Copy information from the kernel data structure associated with

msqid into the msqid\_ds structure pointed to by buf. The caller

must have read permission on the message queue.

IPC\_SET

Write the values of some members of the msqid\_ds structure

pointed to by buf to the kernel data structure associated with

this message queue, updating also its msg\_ctime member. The

following members of the structure are updated: msg\_qbytes,

msg\_perm.uid, msg\_perm.gid, and (the least significant 9 bits

of) msg\_perm.mode. The effective UID of the calling process

must match the owner (msg\_perm.uid) or creator (msg\_perm.cuid)

of the message queue, or the caller must be privileged. Appro-

priate privilege (Linux: the CAP\_IPC\_RESOURCE capability) is

required to raise the msg\_qbytes value beyond the system parame-

ter MSGMNB.

IPC\_RMID

Immediately remove the message queue, awakening all waiting

reader and writer processes (with an error return and errno set

to EIDRM). The calling process must have appropriate privileges

or its effective user ID must be either that of the creator or

owner of the message queue.

IPC\_INFO (Linux-specific)

Returns information about system-wide message queue limits and

parameters in the structure pointed to by buf. This structure

is of type msginfo (thus, a cast is required), defined in

<sys/msg.h> if the \_GNU\_SOURCE feature test macro is defined:

struct msginfo {

int msgpool; /\* Size in kibibytes of buffer pool

used to hold message data;

unused within kernel \*/

int msgmap; /\* Maximum number of entries in message

map; unused within kernel \*/

int msgmax; /\* Maximum number of bytes that can be

written in a single message \*/

int msgmnb; /\* Maximum number of bytes that can be

written to queue; used to initialize

msg\_qbytes during queue creation

(msgget(2)) \*/

int msgmni; /\* Maximum number of message queues \*/

int msgssz; /\* Message segment size;

unused within kernel \*/

int msgtql; /\* Maximum number of messages on all queues

in system; unused within kernel \*/

unsigned short int msgseg;

/\* Maximum number of segments;

unused within kernel \*/

};

The msgmni, msgmax, and msgmnb settings can be changed via /proc

files of the same name; see proc(5) for details.

MSG\_INFO (Linux-specific)

Returns a msginfo structure containing the same information as

for IPC\_INFO, except that the following fields are returned with

information about system resources consumed by message queues:

the msgpool field returns the number of message queues that cur-

rently exist on the system; the msgmap field returns the total

number of messages in all queues on the system; and the msgtql

field returns the total number of bytes in all messages in all

queues on the system.

MSG\_STAT (Linux-specific)

Returns a msqid\_ds structure as for IPC\_STAT. However, the

msqid argument is not a queue identifier, but instead an index

into the kernel’s internal array that maintains information

about all message queues on the system.

RETURN VALUE

On success, IPC\_STAT, IPC\_SET, and IPC\_RMID return 0. A successful

IPC\_INFO or MSG\_INFO operation returns the index of the highest used

entry in the kernel’s internal array recording information about all

message queues. (This information can be used with repeated MSG\_STAT

operations to obtain information about all queues on the system.) A

successful MSG\_STAT operation returns the identifier of the queue whose

index was given in msqid.

On error, -1 is returned with errno indicating the error.

ERRORS

On failure, errno is set to one of the following:

EACCES The argument cmd is equal to IPC\_STAT or MSG\_STAT, but the call-

ing process does not have read permission on the message queue

msqid, and does not have the CAP\_IPC\_OWNER capability.

EFAULT The argument cmd has the value IPC\_SET or IPC\_STAT, but the

address pointed to by buf isn’t accessible.

EIDRM The message queue was removed.

EINVAL Invalid value for cmd or msqid. Or: for a MSG\_STAT operation,

the index value specified in msqid referred to an array slot

that is currently unused.

EPERM The argument cmd has the value IPC\_SET or IPC\_RMID, but the

effective user ID of the calling process is not the creator (as

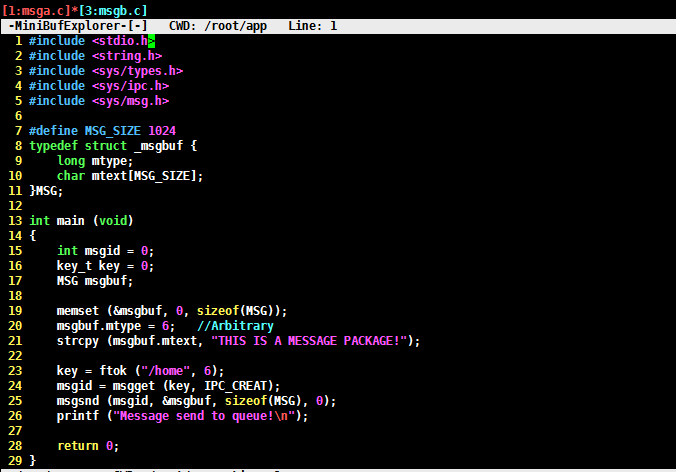
found in msg\_perm.cuid) or the owner (as found in msg\_perm.uid)

of the message queue, and the process is not privileged (Linux:

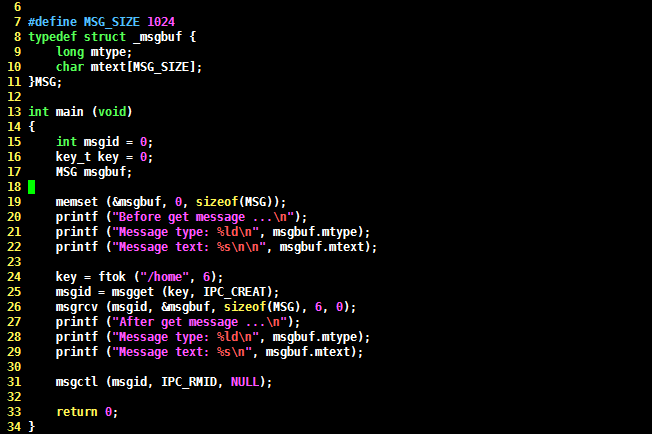
it does not have the CAP\_SYS\_ADMIN capability).

## 消息队列实验核心代码

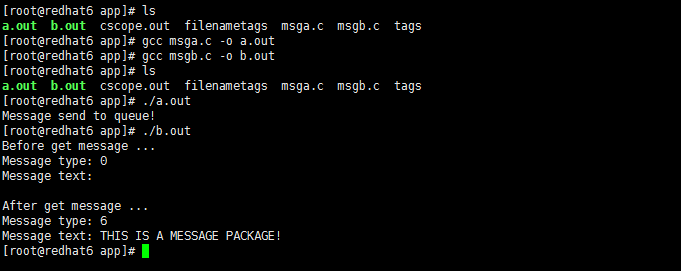
进程A发送特定类型消息至消息队列



进程B从消息队列读取A发送的特定类型消息



执行测试



注意:如果先运行B,且消息队列中没有A发送的特定类型消息,B就会阻塞,直到A发送

# 第20课-多线程程序设计

[主目录](#_目录)

## 核心理论

线程概念引入

多人砌墙故事,A进程一个人干活,但老板嫌太慢,于是A进程请了工人B进程和工人C进程来帮忙干活,于是这下进度快多了,但还有一个问题,老板原来只要发给A进程一个人一口锅做饭,而现在要发给A,B,C进程每人买一口锅,于是老板嫌这个开销太大了.((fork)多进程之间有独立代码段,数据段,堆栈,增大了系统开销)

后来老板想了一个办法,让A去人才市场找大学生来,大学生B线程和大学生C线程来干活,解雇B,C进程工人,因为大学生B,C可以不需要锅,可以和包工头A共用一口锅做饭,于是这个开销大大减少了,这下老板高兴了。既提高了效率又剩下一笔钱。((pthread)多线程之间共享创建者(进程)的资源)

## 创建者进程创建线程:pthread\_create()

NAME

pthread\_create - create a new thread

SYNOPSIS

#include <pthread.h>

int pthread\_create(pthread\_t \*thread, const pthread\_attr\_t \*attr,

void \*(\*start\_routine) (void \*), void \*arg);

Compile and link with -pthread. (gcc –lpthread ...)

DESCRIPTION

The pthread\_create() function starts a new thread in the calling pro-

cess. The new thread starts execution by invoking start\_routine(); arg

is passed as the sole argument of start\_routine().

The new thread terminates in one of the following ways:

\* It calls pthread\_exit(3), specifying an exit status value that is

available to another thread in the same process that calls

pthread\_join(3).

\* It returns from start\_routine(). This is equivalent to calling

pthread\_exit(3) with the value supplied in the return statement.

\* It is canceled (see pthread\_cancel(3)).

\* Any of the threads in the process calls exit(3), or the main thread

performs a return from main(). This causes the termination of all

threads in the process.

RETURN VALUE

On success, pthread\_create() returns 0; on error, it returns an error

number, and the contents of \*thread are undefined.

## 线程获取自己的ID:pthread\_self()

NAME

pthread\_self - obtain ID of the calling thread

SYNOPSIS

#include <pthread.h>

pthread\_t pthread\_self(void);

Compile and link with -pthread.

DESCRIPTION

The pthread\_self() function returns the ID of the calling thread. This

is the same value that is returned in \*thread in the pthread\_create(3)

call that created this thread.

RETURN VALUE

This function always succeeds, returning the calling thread’s ID.

## 创建者进程等待线程退出:pthread\_join()

NAME

pthread\_join - join with a terminated thread

SYNOPSIS

#include <pthread.h>

int pthread\_join(pthread\_t thread, void \*\*retval);

Compile and link with -pthread.

DESCRIPTION

The pthread\_join() function waits for the thread specified by thread to

terminate. If that thread has already terminated, then pthread\_join()

returns immediately. The thread specified by thread must be joinable.

If retval is not NULL, then pthread\_join() copies the exit status of

the target thread (i.e., the value that the target thread supplied to

pthread\_exit(3)) into the location pointed to by \*retval. If the tar-

get thread was canceled, then PTHREAD\_CANCELED is placed in \*retval.

If multiple threads simultaneously try to join with the same thread,

the results are undefined. If the thread calling pthread\_join() is

canceled, then the target thread will remain joinable (i.e., it will

not be detached).

RETURN VALUE

On success, pthread\_join() returns 0; on error, it returns an error

number.

## 执行函数线程退出:pthread\_exit()

NAME

pthread\_exit - terminate calling thread

SYNOPSIS

#include <pthread.h>

void pthread\_exit(void \*retval);

Compile and link with -pthread.

DESCRIPTION

The pthread\_exit() function terminates the calling thread and returns a

value via retval that (if the thread is joinable) is available to

another thread in the same process that calls pthread\_join(3).

Any clean-up handlers established by pthread\_cleanup\_push(3) that have

not yet been popped, are popped (in the reverse of the order in which

they were pushed) and executed. If the thread has any thread-specific

data, then, after the clean-up handlers have been executed, the corre-

sponding destructor functions are called, in an unspecified order.

When a thread terminates, process-shared resources (e.g., mutexes, con-

dition variables, semaphores, and file descriptors) are not released,

and functions registered using atexit(3) are not called.

After the last thread in a process terminates, the process terminates

as by calling exit(3) with an exit status of zero; thus, process-shared

resources are released and functions registered using atexit(3) are

called.

RETURN VALUE

This function does not return to the caller.

## 多线程互斥锁初始化

NAME

pthread\_mutex\_destroy, pthread\_mutex\_init - destroy and initialize a

mutex

SYNOPSIS

#include <pthread.h>

int pthread\_mutex\_destroy(pthread\_mutex\_t \*mutex);

int pthread\_mutex\_init(pthread\_mutex\_t \*restrict mutex,

const pthread\_mutexattr\_t \*restrict attr);

pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;

DESCRIPTION

The pthread\_mutex\_destroy() function shall destroy the mutex object

referenced by mutex; the mutex object becomes, in effect, uninitial-

ized. An implementation may cause pthread\_mutex\_destroy() to set the

object referenced by mutex to an invalid value. A destroyed mutex

object can be reinitialized using pthread\_mutex\_init(); the results of

otherwise referencing the object after it has been destroyed are unde-

fined.

It shall be safe to destroy an initialized mutex that is unlocked.

Attempting to destroy a locked mutex results in undefined behavior.

The pthread\_mutex\_init() function shall initialize the mutex referenced

by mutex with attributes specified by attr. If attr is NULL, the

default mutex attributes are used; the effect shall be the same as

passing the address of a default mutex attributes object. Upon success-

ful initialization, the state of the mutex becomes initialized and

unlocked.

Only mutex itself may be used for performing synchronization. The

result of referring to copies of mutex in calls to

pthread\_mutex\_lock(), pthread\_mutex\_trylock(), pthread\_mutex\_unlock(),

and pthread\_mutex\_destroy() is undefined.

Attempting to initialize an already initialized mutex results in unde-

fined behavior.

In cases where default mutex attributes are appropriate, the macro

PTHREAD\_MUTEX\_INITIALIZER can be used to initialize mutexes that are

statically allocated. The effect shall be equivalent to dynamic ini-

tialization by a call to pthread\_mutex\_init() with parameter attr spec-

ified as NULL, except that no error checks are performed.

RETURN VALUE

If successful, the pthread\_mutex\_destroy() and pthread\_mutex\_init()

functions shall return zero; otherwise, an error number shall be

returned to indicate the error.

The [EBUSY] and [EINVAL] error checks, if implemented, act as if they

were performed immediately at the beginning of processing for the func-

tion and shall cause an error return prior to modifying the state of

the mutex specified by mutex.

## 多线程互斥锁操作

NAME

pthread\_mutex\_lock, pthread\_mutex\_trylock, pthread\_mutex\_unlock - lock

and unlock a mutex

SYNOPSIS

#include <pthread.h>

int pthread\_mutex\_lock(pthread\_mutex\_t \*mutex);

int pthread\_mutex\_trylock(pthread\_mutex\_t \*mutex);

int pthread\_mutex\_unlock(pthread\_mutex\_t \*mutex);

DESCRIPTION

The mutex object referenced by mutex shall be locked by calling

pthread\_mutex\_lock(). If the mutex is already locked, the calling

thread shall block until the mutex becomes available. This operation

shall return with the mutex object referenced by mutex in the locked

state with the calling thread as its owner.

If the mutex type is PTHREAD\_MUTEX\_NORMAL, deadlock detection shall not

be provided. Attempting to relock the mutex causes deadlock. If a

thread attempts to unlock a mutex that it has not locked or a mutex

which is unlocked, undefined behavior results.

If the mutex type is PTHREAD\_MUTEX\_ERRORCHECK, then error checking

shall be provided. If a thread attempts to relock a mutex that it has

already locked, an error shall be returned. If a thread attempts to

unlock a mutex that it has not locked or a mutex which is unlocked, an

error shall be returned.

If the mutex type is PTHREAD\_MUTEX\_RECURSIVE, then the mutex shall

maintain the concept of a lock count. When a thread successfully

acquires a mutex for the first time, the lock count shall be set to

one. Every time a thread relocks this mutex, the lock count shall be

incremented by one. Each time the thread unlocks the mutex, the lock

count shall be decremented by one. When the lock count reaches zero,

the mutex shall become available for other threads to acquire. If a

thread attempts to unlock a mutex that it has not locked or a mutex

which is unlocked, an error shall be returned.

If the mutex type is PTHREAD\_MUTEX\_DEFAULT, attempting to recursively

lock the mutex results in undefined behavior. Attempting to unlock the

mutex if it was not locked by the calling thread results in undefined

behavior. Attempting to unlock the mutex if it is not locked results in

undefined behavior.

The pthread\_mutex\_trylock() function shall be equivalent to

pthread\_mutex\_lock(), except that if the mutex object referenced by

mutex is currently locked (by any thread, including the current

thread), the call shall return immediately. If the mutex type is

PTHREAD\_MUTEX\_RECURSIVE and the mutex is currently owned by the calling

thread, the mutex lock count shall be incremented by one and the

pthread\_mutex\_trylock() function shall immediately return success.

The pthread\_mutex\_unlock() function shall release the mutex object ref-

erenced by mutex. The manner in which a mutex is released is dependent

upon the mutex’s type attribute. If there are threads blocked on the

mutex object referenced by mutex when pthread\_mutex\_unlock() is called,

resulting in the mutex becoming available, the scheduling policy shall

determine which thread shall acquire the mutex.

(In the case of PTHREAD\_MUTEX\_RECURSIVE mutexes, the mutex shall become

available when the count reaches zero and the calling thread no longer

has any locks on this mutex.)

If a signal is delivered to a thread waiting for a mutex, upon return

from the signal handler the thread shall resume waiting for the mutex

as if it was not interrupted.

RETURN VALUE

If successful, the pthread\_mutex\_lock() and pthread\_mutex\_unlock()

functions shall return zero; otherwise, an error number shall be

returned to indicate the error.

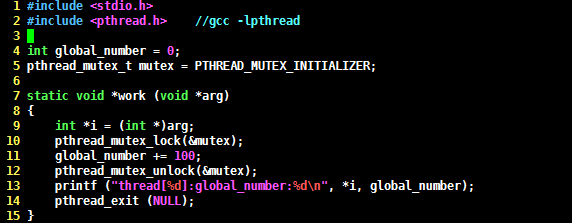
The pthread\_mutex\_trylock() function shall return zero if a lock on the

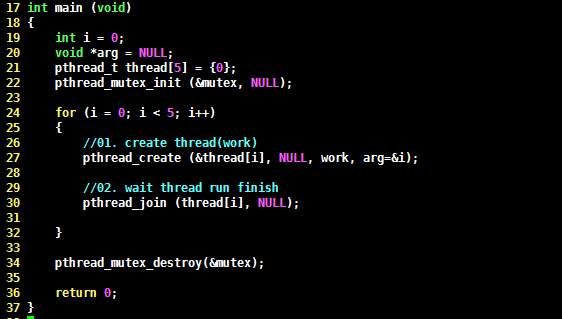
mutex object referenced by mutex is acquired. Otherwise, an error num-

ber is returned to indicate the error.

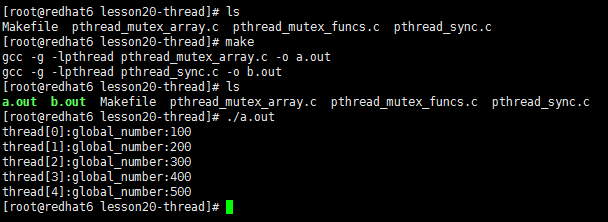
## 多线程实验核心代码

### 范例代码1:数组类型线程组

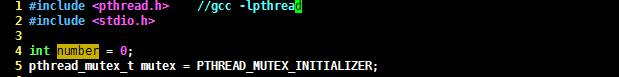


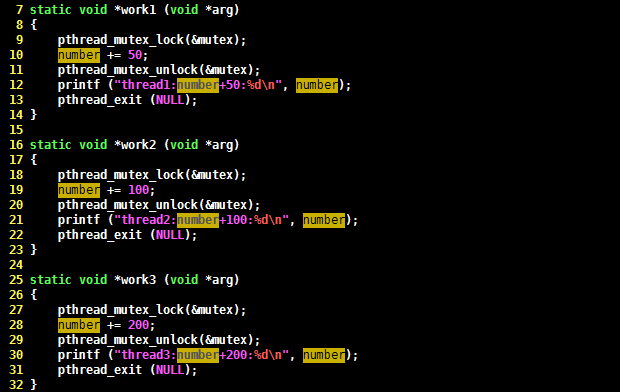


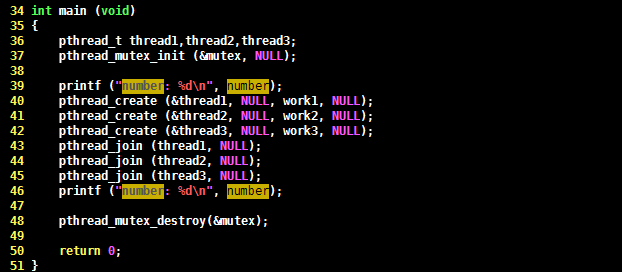
### 范例代码1:执行结果



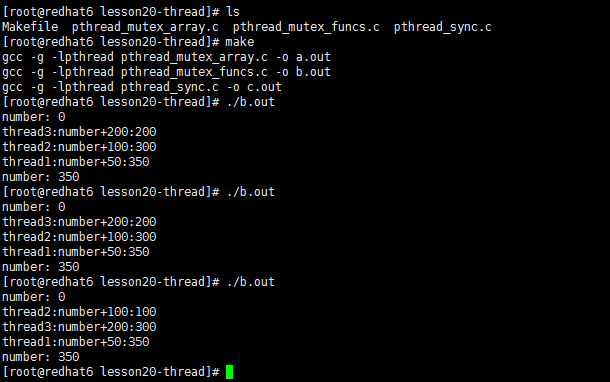
### 范例代码2:多函数类型线程组







### 范例代码2:执行结果



# 第21课-多线程同步

[主目录](#_目录)

## 核心理论

线程同步

多个线程按照规定的顺序执行,即可为线程同步.

## 多线程同步机制

初始化条件变量: pthread\_cond\_t cond\_ready = PTHREAD\_COND\_INITIALIZER;

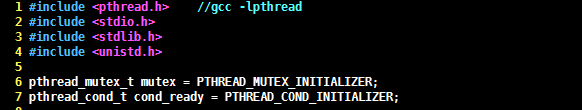
等待条件成熟: pthread\_cond\_wait (&cond\_ready, &mutex);

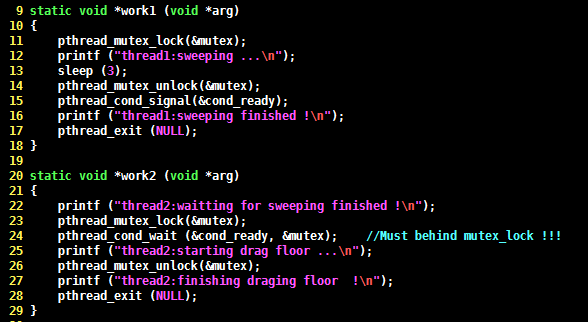
设置条件成熟: pthread\_cond\_signal (&cond\_ready);

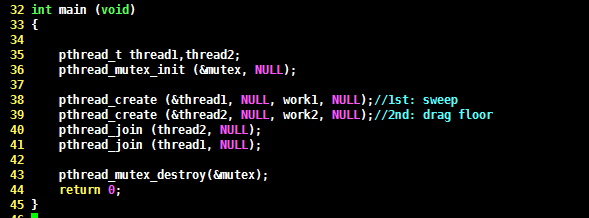
## 多线程同步实验核心代码

多线程同步就是先后顺序做事情，从上节课多线程程序设计实验中我们得知线程执行顺序不固定，所以为了让两个线程按照一定顺序执行，我们不得不加入同步机制，目前我们只实验了2个线程之间的同步关系，3个线程或者更多线程的同步关系有待研究。

下面是我们为线程同步设计的实验，线程1扫地，线程2拖地，所以必须先线程1扫完地，然后线程2才能拖地，即使线程2先做好了拖地的准备，但也得等线程1扫完地了才能拖地。细节看下列实验代码：

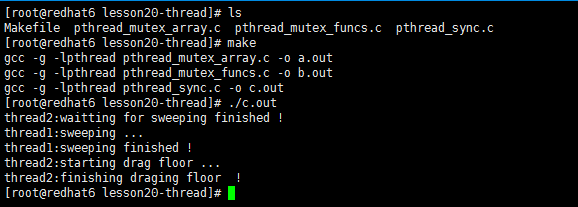






多线程同步实验代码测试效果图如下：（先扫地后拖地）：

从下面执行结果中我们看出，如果我们不加入同步机制，那么很可能就是线程2先拖地，线程1后扫地了，这个顺序控制在我们实际生产生活中也至关重要。



# 第22课-网络协议分析

[主目录](#_目录)

## 核心理论

网络模型

OSI七层模型 TCP/IP模型

应用层 应用层

表示层

会话层

传输层 传输层

网络层 网络层

数据链路层 网络接口层(网卡驱动)

物理层

网络协议概念

首先它是一种规则

协议分析

以太网协议 == IP协议 == TCP协议 == UDP协议

协议架构

|以太网首部|IP首部|TCP首部|应用数据|以太网尾部|

MAC地址

即网卡的物理地址,任何两块网卡物理地址都是不一样的,网卡地址由专门机构分配。我们可以根据网卡的地址段(网卡地址的前三个字节),可以知道网卡的生产厂家。

## 以太网协议包格式

struct 以太网协议格式 {

同步位(PR)

分隔位(SD)

目的地址(DA)[6];

源地址(SA)[6];

类型(TYPE)[2];

数据段(DATA)[1500] {

struct IP {

版本;

长度;

服务类型;

总长度;

识别;

标记;

分段偏移;

生存期;

协议;

头检查和;

源IP地址;

目的IP地址;

IP选项;

填充;

union IP数据报文数据 {

struct TCP {

源端口;

目的端口;

顺序号数;

确认号数;

数据偏移;

保留;

XXX;

窗口;

校验和;

紧急提示器;

选项;

填充;

TCP数据;

};

struct UDP {

源端口;

目的端口;

报文长度;

校验和;

数据;

数据;

数据;

};

};

};

填充位(PAD) [46];

CRC数据校验位(FCS)[32];

};

# 第23课-Linux网络编程模型

[主目录](#_目录)

## 核心理论:TCP/UDP编程模型

编程模型:socket

Socket的实质就是一个接口,利用该接口用户在使用不同的网络协议时,操作函数得以统一。而针对不同协议的差异性操作,则交给socket去自行解决。我们Linux的网络编程模型其实就是一种基于套接字(如socket)的编程模型。套接字说白了就是一个接口。

TCP应用程序编程模型

创建socket:socket() 创建socket:socket()

绑定地址:bind()

监听端口:listen()

等待连接:accept() 连接服务器:connect()

收发数据:read()/write() 收发数据:read()/write()

结束连接:close() 结束连接:close()

UDP应用程序编程模型

创建socket:socket() 创建socket:socket()

绑定地址:bind()

~~监听端口~~

~~等待连接 连接服务器~~

收发数据:read()/write() 收发数据:read()/write()

结束连接:close() 结束连接:close()

# 第24课-TCP通讯程序设计

[主目录](#_目录)

## 创建套接字socket:socket()

NAME

socket - create an endpoint for communication

SYNOPSIS

#include <sys/types.h> /\* See NOTES \*/

#include <sys/socket.h>

int socket(int domain, int type, int protocol);

DESCRIPTION

socket() creates an endpoint for communication and returns a descrip-

tor.

The domain argument specifies a communication domain; this selects the

protocol family which will be used for communication. These families

are defined in <sys/socket.h>. The currently understood formats

include:

Name Purpose Man page

AF\_UNIX, AF\_LOCAL Local communication unix(7)

AF\_INET IPv4 Internet protocols ip(7)

AF\_INET6 IPv6 Internet protocols ipv6(7)

AF\_IPX IPX - Novell protocols

AF\_NETLINK Kernel user interface device netlink(7)

AF\_X25 ITU-T X.25 / ISO-8208 protocol x25(7)

AF\_AX25 Amateur radio AX.25 protocol

AF\_ATMPVC Access to raw ATM PVCs

AF\_APPLETALK Appletalk ddp(7)

AF\_PACKET Low level packet interface packet(7)

The socket has the indicated type, which specifies the communication

semantics. Currently defined types are:

SOCK\_STREAM Provides sequenced, reliable, two-way, connection-based

byte streams. An out-of-band data transmission mecha-

nism may be supported.

SOCK\_DGRAM Supports datagrams (connectionless, unreliable messages

of a fixed maximum length).

SOCK\_SEQPACKET Provides a sequenced, reliable, two-way connection-

based data transmission path for datagrams of fixed

maximum length; a consumer is required to read an

entire packet with each input system call.

SOCK\_RAW Provides raw network protocol access.

SOCK\_RDM Provides a reliable datagram layer that does not guar-

antee ordering.

SOCK\_PACKET Obsolete and should not be used in new programs; see

packet(7).

Some socket types may not be implemented by all protocol families; for

example, SOCK\_SEQPACKET is not implemented for AF\_INET.

Since Linux 2.6.27, the type argument serves a second purpose: in addi-

tion to specifying a socket type, it may include the bitwise OR of any

of the following values, to modify the behavior of socket():

SOCK\_NONBLOCK Set the O\_NONBLOCK file status flag on the new open

file description. Using this flag saves extra calls to

fcntl(2) to achieve the same result.

SOCK\_CLOEXEC Set the close-on-exec (FD\_CLOEXEC) flag on the new file

descriptor. See the description of the O\_CLOEXEC flag

in open(2) for reasons why this may be useful.

RETURN VALUE

On success, a file descriptor for the new socket is returned. On

error, -1 is returned, and errno is set appropriately.

## 绑定服务器地址:bind()

NAME

bind - bind a name to a socket

SYNOPSIS

#include <sys/types.h> /\* See NOTES \*/

#include <sys/socket.h>

int bind(int sockfd, const struct sockaddr \*addr,socklen\_t addrlen);

DESCRIPTION

When a socket is created with socket(2), it exists in a name space

(address family) but has no address assigned to it. bind() assigns the

address specified to by addr to the socket referred to by the file

descriptor sockfd. addrlen specifies the size, in bytes, of the

address structure pointed to by addr. Traditionally, this operation is

called “assigning a name to a socket”.

It is normally necessary to assign a local address using bind() before

a SOCK\_STREAM socket may receive connections (see accept(2)).

The rules used in name binding vary between address families. Consult

the manual entries in Section 7 for detailed information. For AF\_INET

see ip(7), for AF\_INET6 see ipv6(7), for AF\_UNIX see unix(7), for

AF\_APPLETALK see ddp(7), for AF\_PACKET see packet(7), for AF\_X25 see

x25(7) and for AF\_NETLINK see netlink(7).

The actual structure passed for the addr argument will depend on the

address family. The sockaddr structure is defined as something like:

struct sockaddr {

sa\_family\_t sa\_family;

char sa\_data[14];

}

struct sockaddr\_in {

sa\_family\_t sin\_family;

u\_short\_int sin\_port;

struct in\_addr sin\_addr;

u\_char sin\_zero[8];

}

struct in\_addr {

ulong s\_addr;

}

The only purpose of this structure is to cast the structure pointer

passed in addr in order to avoid compiler warnings. See EXAMPLE below.

RETURN VALUE

On success, zero is returned. On error, -1 is returned, and errno is

set appropriately.

## Linux网络编程端口常用转换

NAME

htonl, htons, ntohl, ntohs - convert values between host and network

byte order

SYNOPSIS

#include <arpa/inet.h>

uint32\_t htonl(uint32\_t hostlong);

uint16\_t htons(uint16\_t hostshort);

uint32\_t ntohl(uint32\_t netlong);

uint16\_t ntohs(uint16\_t netshort);

DESCRIPTION

The htonl() function converts the unsigned integer hostlong from host

byte order to network byte order.

The htons() function converts the unsigned short integer hostshort from

host byte order to network byte order.

The ntohl() function converts the unsigned integer netlong from network

byte order to host byte order.

The ntohs() function converts the unsigned short integer netshort from

network byte order to host byte order.

On the i386 the host byte order is Least Significant Byte first,

whereas the network byte order, as used on the Internet, is Most Sig-

nificant Byte first.

## Linux网络编程IP地址常用转换

NAME

inet\_aton, inet\_addr, inet\_network, inet\_ntoa, inet\_makeaddr,

inet\_lnaof, inet\_netof - Internet address manipulation routines

SYNOPSIS

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

int inet\_aton(const char \*cp, struct in\_addr \*inp);

in\_addr\_t inet\_addr(const char \*cp);//将字符串IP地址转换为一个整数

in\_addr\_t inet\_network(const char \*cp);

char \*inet\_ntoa(struct in\_addr in);//将整数IP地址转换为字符串IP地址

struct in\_addr inet\_makeaddr(int net, int host);

in\_addr\_t inet\_lnaof(struct in\_addr in);

in\_addr\_t inet\_netof(struct in\_addr in);

Feature Test Macro Requirements for glibc (see feature\_test\_macros(7)):

inet\_aton(), inet\_ntoa(): \_BSD\_SOURCE || \_SVID\_SOURCE

DESCRIPTION

inet\_aton() converts the Internet host address cp from the IPv4 num-

bers-and-dots notation into binary form (in network byte order) and

stores it in the structure that inp points to. inet\_aton() returns

non-zero if the address is valid, zero if not. The address supplied in

cp can have one of the following forms:

a.b.c.d Each of the four numeric parts specifies a byte of the

address; the bytes are assigned in left-to-right order to

produce the binary address.

a.b.c Parts a and b specify the first two bytes of the binary

address. Part c is interpreted as a 16-bit value that

defines the rightmost two bytes of the binary address. This

notation is suitable for specifying (outmoded) Class B net-

work addresses.

a.b Part a specifies the first byte of the binary address. Part

b is interpreted as a 24-bit value that defines the rightmost

three bytes of the binary address. This notation is suitable

for specifying (outmoded) Class C network addresses.

a The value a is interpreted as a 32-bit value that is stored

directly into the binary address without any byte rearrange-

ment.

In all of the above forms, components of the dotted address can be

specified in decimal, octal (with a leading 0), or hexadecimal, with a

leading 0X). Addresses in any of these forms are collectively termed

IPV4 numbers-and-dots notation. The form that uses exactly four deci-

mal numbers is referred to as IPv4 dotted-decimal notation (or some-

times: IPv4 dotted-quad notation).

The inet\_addr() function converts the Internet host address cp from

IPv4 numbers-and-dots notation into binary data in network byte order.

If the input is invalid, INADDR\_NONE (usually -1) is returned. Use of

this function is problematic because -1 is a valid address

(255.255.255.255). Avoid its use in favor of inet\_aton(),

inet\_pton(3), or getaddrinfo(3) which provide a cleaner way to indicate

error return.

The inet\_network() function converts cp, a string in IPv4 numbers-and-

dots notation, into a number in host byte order suitable for use as an

Internet network address. On success, the converted address is

returned. If the input is invalid, -1 is returned.

The inet\_ntoa() function converts the Internet host address in, given

in network byte order, to a string in IPv4 dotted-decimal notation.

The string is returned in a statically allocated buffer, which subse-

quent calls will overwrite.

The inet\_lnaof() function returns the local network address part of the

Internet address in. The returned value is in host byte order.

The inet\_netof() function returns the network number part of the Inter-

net address in. The returned value is in host byte order.

The inet\_makeaddr() function is the converse of inet\_netof() and

inet\_lnaof(). It returns an Internet host address in network byte

order, created by combining the network number net with the local

address host, both in host byte order.

The structure in\_addr as used in inet\_ntoa(), inet\_makeaddr(),

inet\_lnaof() and inet\_netof() is defined in <netinet/in.h> as:

typedef uint32\_t in\_addr\_t;

struct in\_addr {

in\_addr\_t s\_addr;

};

## 监听端口:listen()

NAME

listen - listen for connections on a socket

SYNOPSIS

#include <sys/types.h> /\* See NOTES \*/

#include <sys/socket.h>

int listen(int sockfd, int backlog);

DESCRIPTION

listen() marks the socket referred to by sockfd as a passive socket,

that is, as a socket that will be used to accept incoming connection

requests using accept(2).

The sockfd argument is a file descriptor that refers to a socket of

type SOCK\_STREAM or SOCK\_SEQPACKET.

The backlog argument defines the maximum length to which the queue of

pending connections for sockfd may grow. If a connection request

arrives when the queue is full, the client may receive an error with an

indication of ECONNREFUSED or, if the underlying protocol supports

retransmission, the request may be ignored so that a later reattempt at

connection succeeds.

RETURN VALUE

On success, zero is returned. On error, -1 is returned, and errno is

set appropriately.

## 等待连接:accept()

NAME

accept - accept a connection on a socket

SYNOPSIS

#include <sys/types.h> /\* See NOTES \*/

#include <sys/socket.h>

int accept(int sockfd, struct sockaddr \*addr, socklen\_t \*addrlen);

#define \_GNU\_SOURCE

#include <sys/socket.h>

int accept4(int sockfd, struct sockaddr \*addr,

socklen\_t \*addrlen, int flags);

DESCRIPTION

The accept() system call is used with connection-based socket types

(SOCK\_STREAM, SOCK\_SEQPACKET). It extracts the first connection

request on the queue of pending connections for the listening socket,

sockfd, creates a new connected socket, and returns a new file descrip-

tor referring to that socket. The newly created socket is not in the

listening state. The original socket sockfd is unaffected by this

call.

The argument sockfd is a socket that has been created with socket(2),

bound to a local address with bind(2), and is listening for connections

after a listen(2).

If flags is 0, then accept4() is the same as accept(). The following

values can be bitwise ORed in flags to obtain different behavior:

SOCK\_NONBLOCK Set the O\_NONBLOCK file status flag on the new open

file description. Using this flag saves extra calls to

fcntl(2) to achieve the same result.

SOCK\_CLOEXEC Set the close-on-exec (FD\_CLOEXEC) flag on the new file

descriptor. See the description of the O\_CLOEXEC flag

in open(2) for reasons why this may be useful.

RETURN VALUE

On success, these system calls return a non-negative integer that is a

descriptor for the accepted socket. On error, -1 is returned, and

errno is set appropriately.

## 连接服务器:connect()

NAME

connect - initiate a connection on a socket

SYNOPSIS

#include <sys/types.h> /\* See NOTES \*/

#include <sys/socket.h>

int connect(int sockfd, const struct sockaddr \*addr,socklen\_t addrlen);

DESCRIPTION

The connect() system call connects the socket referred to by the file

descriptor sockfd to the address specified by addr. The addrlen argu-

ment specifies the size of addr. The format of the address in addr is

determined by the address space of the socket sockfd; see socket(2) for

further details.

If the socket sockfd is of type SOCK\_DGRAM then addr is the address to

which datagrams are sent by default, and the only address from which

datagrams are received. If the socket is of type SOCK\_STREAM or

SOCK\_SEQPACKET, this call attempts to make a connection to the socket

that is bound to the address specified by addr.

Generally, connection-based protocol sockets may successfully connect()

only once; connectionless protocol sockets may use connect() multiple

times to change their association. Connectionless sockets may dissolve

the association by connecting to an address with the sa\_family member

of sockaddr set to AF\_UNSPEC (supported on Linux since kernel 2.2).

RETURN VALUE

If the connection or binding succeeds, zero is returned. On error, -1

is returned, and errno is set appropriately.

## 发送数据:sendto()

NAME

send, sendto, sendmsg - send a message on a socket

SYNOPSIS

#include <sys/types.h>

#include <sys/socket.h>

ssize\_t send(int sockfd, const void \*buf, size\_t len, int flags);

ssize\_t sendto(int sockfd, const void \*buf, size\_t len, int flags,

const struct sockaddr \*dest\_addr, socklen\_t addrlen);

ssize\_t sendmsg(int sockfd, const struct msghdr \*msg, int flags);

DESCRIPTION

The system calls send(), sendto(), and sendmsg() are used to transmit a

message to another socket.

The send() call may be used only when the socket is in a connected

state (so that the intended recipient is known). The only difference

between send() and write(2) is the presence of flags. With zero flags

argument, send() is equivalent to write(2). Also, the following call

send(sockfd, buf, len, flags);

is equivalent to

sendto(sockfd, buf, len, flags, NULL, 0);

The argument sockfd is the file descriptor of the sending socket.

RETURN VALUE

On success, these calls return the number of characters sent. On

error, -1 is returned, and errno is set appropriately.

## 接收数据:recvfrom(),recv()

NAME

recv, recvfrom, recvmsg - receive a message from a socket

SYNOPSIS

#include <sys/types.h>

#include <sys/socket.h>

ssize\_t recv(int sockfd, void \*buf, size\_t len, int flags);

ssize\_t recvfrom(int sockfd, void \*buf, size\_t len, int flags,

struct sockaddr \*src\_addr, socklen\_t \*addrlen);

ssize\_t recvmsg(int sockfd, struct msghdr \*msg, int flags);

DESCRIPTION

The recvfrom() and recvmsg() calls are used to receive messages from a

socket, and may be used to receive data on a socket whether or not it

is connection-oriented.

If src\_addr is not NULL, and the underlying protocol provides the

source address, this source address is filled in. When src\_addr is

NULL, nothing is filled in; in this case, addrlen is not used, and

should also be NULL. The argument addrlen is a value-result argument,

which the caller should initialize before the call to the size of the

buffer associated with src\_addr, and modified on return to indicate the

actual size of the source address. The returned address is truncated

if the buffer provided is too small; in this case, addrlen will return

a value greater than was supplied to the call.

The recv() call is normally used only on a connected socket (see con-

nect(2)) and is identical to recvfrom() with a NULL src\_addr argument.

RETURN VALUE

These calls return the number of bytes received, or -1 if an error

occurred. The return value will be 0 when the peer has performed an

orderly shutdown.

## TCP网络编程实验核心代码

### TCP服务端核心代码:server.c

//创建socket

sockfd = socket (AF\_INET, SOCK\_STREAM, 0);

if (0 > sockfd)

{

perror ("socket");

exit (EXIT\_FAILURE);

}

//初始化地址

struct sockaddr\_in client;

struct sockaddr\_in server;

memset (&client, 0, sizeof(client));

memset (&server, 0, sizeof(server));

server.sin\_family = AF\_INET;

server.sin\_port = htons(6666);

server.sin\_addr.s\_addr = inet\_addr(argv[1]);//# ./a.out 192.168.1.10

//绑定服务器地址

if (bind(sockfd,(const struct sockaddr \*)&server,sizeof(server)) != 0)

{

perror ("bind");

exit (EXIT\_FAILURE);

}

//监听端口

if (listen(sockfd, 3) != 0)

{

perror ("listen");

exit (EXIT\_FAILURE);

}

//等待连接

size = sizeof(client);printf ("Waiting Connecting ...\n");

newfd = accept (sockfd, (struct sockaddr \*)&client, &size);

//连接成功,开始传输(发送)数据

printf ("Connecting successful! Writing Data ...\n");

strcpy (buf, "I Like Embedded!\n");

write (newfd, buf, strlen(buf));

//关闭连接

close (newfd);

### TCP客户端核心代码:client.c

//创建socket

sockfd = socket (AF\_INET, SOCK\_STREAM, 0);

if (0 > sockfd)

{

perror ("socket");

exit (EXIT\_FAILURE);

}

//初始化地址

struct sockaddr\_in client;

memset (&client, 0, sizeof(client));

client.sin\_family = AF\_INET;

client.sin\_port = htons(6666);

client.sin\_addr.s\_addr = inet\_addr(argv[1]);

//连接服务器

size = sizeof(client);printf ("Waiting Connecting ...\n");

if (connect(sockfd, (struct sockaddr \*)&client, sizeof(client)) != 0)

{

perror ("connect");

exit (EXIT\_FAILURE);

}

//连接成功,开始传输(接收)数据

printf ("Connecting successful! Reading Data ...\n");

read (sockfd, buf, sizeof(buf));

printf ("buf: %s\n", buf);

//关闭连接

close (sockfd);

# 第25课-UDP通讯程序设计

[主目录](#_目录)

UDP主要应用于仅客户端向服务器上传数据

UDP应用程序编程参考模型

创建socket:socket() 创建socket:socket()

绑定地址:bind()

~~监听端口~~

~~等待连接 连接服务器~~

收发数据:read()/write() 收发数据:read()/write()

结束连接:close() 结束连接:close()

## UDP网络编程实验核心代码

### UDP服务端核心代码:接收数据:server.c

//创建socket

sockfd = socket (AF\_INET, SOCK\_DGRAM, 0);

if (0 > sockfd)

{

perror ("socket");

exit (EXIT\_FAILURE);

}

struct sockaddr\_in client;

struct sockaddr\_in server;

memset (&client, 0, sizeof(client));

memset (&server, 0, sizeof(server));

server.sin\_family = AF\_INET;

server.sin\_port = htons(6666);

server.sin\_addr.s\_addr = inet\_addr(argv[1]);//# ./a.out 192.168.1.10

//绑定服务器地址

if (bind(sockfd,(const struct sockaddr \*)&server,sizeof(server)) != 0)

{

perror ("bind");

exit (EXIT\_FAILURE);

}

//连接成功,开始传输数据

printf ("Connecting successful! Reading Data ...\n");

//此处接收端如果没有收到实时数据就会阻塞

recvfrom (sockfd, buf, sizeof(buf), 0, \

(struct sockaddr \*)&server, &size);

printf ("buf: %s\n", buf);

//关闭连接

close (sockfd);

### UDP客户端核心代码:上传数据:client.c

//创建socket

sockfd = socket (AF\_INET, SOCK\_DGRAM, 0);

if (0 > sockfd)

{

perror ("socket");

exit (EXIT\_FAILURE);

}

struct sockaddr\_in client;

memset (&client, 0, sizeof(client));

client.sin\_family = AF\_INET;

client.sin\_port = htons(6666);

client.sin\_addr.s\_addr = inet\_addr(argv[1]);

//连接成功,开始传输数据

size = sizeof(client);

printf ("Connecting successful! Writting Data ...\n");

strcpy (buf, "I Like Embedded!\n");

sendto (sockfd, buf, sizeof(buf), 0, \

(const struct sockaddr \*)&client, sizeof(client));

printf ("buf: %s\n", buf);

//关闭连接

close (sockfd);

# 第26课-网络并发服务器设计

[主目录](#_目录)

## 核心理论

TCP应用程序编程模型(循环服务器)

创建socket:socket() 创建socket:socket()

绑定地址:bind()

监听端口:listen()

等待连接:accept() 连接服务器:connect()

收发数据:read()/write() 收发数据:read()/write()

结束连接:close() 结束连接:close()

多进程并发服务器

数据处理交给子进程了,父进程空闲下来继续去接受其他主机请求

## 多进程并发服务器实验核心代码

### 服务端核心代码:server.c

//创建socket

sockfd = socket (AF\_INET, SOCK\_STREAM, 0);

if (0 > sockfd)

{

perror ("socket");

exit (EXIT\_FAILURE);

}

struct sockaddr\_in client;

struct sockaddr\_in server;

memset (&client, 0, sizeof(client));

memset (&server, 0, sizeof(server));

server.sin\_family = AF\_INET;

server.sin\_port = htons(6666);

server.sin\_addr.s\_addr = inet\_addr(argv[1]);//# ./a.out 192.168.1.10

//绑定服务器地址

if (bind(sockfd,(const struct sockaddr \*)&server,sizeof(server)) != 0)

{

perror ("bind");

exit (EXIT\_FAILURE);

}

//监听

if (listen(sockfd, 3) != 0)

{

perror ("listen");

exit (EXIT\_FAILURE);

}

size = sizeof(server);

while (1)

{

newfd = accept (sockfd, (struct sockaddr \*)&server, &size);

//阻塞等待连接

if (fork() == 0) //连接成功就交给子进程处理数据

{

printf ("Connecting successful! Pid[%u]Writting Data ...\n", getpid());

strcpy (buf, "I Love Embedded!");

write (newfd, buf, sizeof(buf));

}

//空闲的父进程继续等待连接

}

close (newfd);

### 客户端核心代码:client.c

int main (int argc, char \*argv[])

{

if (2 != argc)

{

printf ("Usage: %s %s\n", argv[0], "IP\_addr");

exit (1);

}

int sockfd = 0;

int newfd = 0;

int size = 0;

char buf[50] = {0};

//创建socket

sockfd = socket (AF\_INET, SOCK\_STREAM, 0);

if (0 > sockfd)

{

perror ("socket");

exit (EXIT\_FAILURE);

}

struct sockaddr\_in client;

struct sockaddr\_in server;

memset (&client, 0, sizeof(client));

memset (&server, 0, sizeof(server));

client.sin\_family = AF\_INET;

client.sin\_port = htons(6666);

client.sin\_addr.s\_addr = inet\_addr(argv[1]);//# ./a.out 192.168.1.10

//连接服务器

size = sizeof(client);

if (connect(sockfd, (const struct sockaddr \*)&client, size) != 0)

{

perror ("connect");

exit (EXIT\_FAILURE);

}

//连接成功,接收数据

printf ("Connecting successful! Reading Data ...\n");

read (sockfd, buf, sizeof(buf));printf ("buf: %s\n", buf);

close (newfd);

return 0;

}

## 多线程并发服务器实验核心代码

### 服务端核心代码:server.c

char buf[100] = {0}; //共享资源的互斥锁保护

pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;

void \*work1 (void \*fd) //传入客户机网络套接子描述符

{

int \*newfd = (int \*)fd;

printf ("pthread[%u] processing data ...\n", pthread\_self());//无符号打印

pthread\_mutex\_lock (&mutex); //锁住互斥锁

strcpy (buf, "I Love Embedded!");

write (\*newfd, buf, sizeof(buf));

pthread\_mutex\_unlock (&mutex);

printf ("wrData: %s\n", buf);

pthread\_exit (NULL); //线程任务退出

}

int main (int argc, char \*argv[])

{

if (2 != argc)

{

printf ("Usage: %s %s\n", argv[0], "IP\_addr");

exit (1);

}

int sockfd = 0;

int size = 0;

int newfd = 0;

pthread\_t thread1 = 0;

pthread\_mutex\_init (&mutex, NULL);

//创建socket

sockfd = socket (AF\_INET, SOCK\_STREAM, 0);

if (0 > sockfd)

{

perror ("socket");

exit (EXIT\_FAILURE);

}

struct sockaddr\_in client;

struct sockaddr\_in server;

memset (&client, 0, sizeof(client));

memset (&server, 0, sizeof(server));

server.sin\_family = AF\_INET;

server.sin\_port = htons(6666);

server.sin\_addr.s\_addr = inet\_addr(argv[1]);//# ./a.out 192.168.1.10

//绑定服务器地址

if (bind(sockfd,(const struct sockaddr \*)&server,sizeof(server)) != 0)

{

perror ("bind");

exit (EXIT\_FAILURE);

}

//监听

if (listen(sockfd, 3) != 0)

{

perror ("listen");

exit (EXIT\_FAILURE);

}

size = sizeof(server);

while (1)

{

//阻塞等待连接

if((newfd=accept(sockfd,(struct sockaddr \*)&server,&size)) < 0)

{

perror ("accept");

exit (EXIT\_FAILURE);

}

//连接成功,把数据处理叫给线程完成

if (pthread\_create (&thread1 ,NULL, work1, &newfd) != 0)

{

perror ("pthread\_create");

exit (EXIT\_FAILURE);

}

memset (buf, 0, sizeof(buf));

//空闲的父进程继续等待连接

}

pthread\_mutex\_destroy (&mutex);

close (newfd);

return 0;

}

### 客户端核心代码:client.c

int main (int argc, char \*argv[])

{

if (2 != argc)

{

printf ("Usage: %s %s\n", argv[0], "IP\_addr");

exit (1);

}

int sockfd = 0;

int newfd = 0;

int size = 0;

char buf[50] = {0};

//创建socket

sockfd = socket (AF\_INET, SOCK\_STREAM, 0);

if (0 > sockfd)

{

perror ("socket");

exit (EXIT\_FAILURE);

}

struct sockaddr\_in client;

struct sockaddr\_in server;

memset (&client, 0, sizeof(client));

memset (&server, 0, sizeof(server));

client.sin\_family = AF\_INET;

client.sin\_port = htons(6666);

client.sin\_addr.s\_addr = inet\_addr(argv[1]);//# ./a.out 192.168.1.10

//连接服务器

size = sizeof(client);

if (connect(sockfd, (const struct sockaddr \*)&client, size) != 0)

{

perror ("connect");

exit (EXIT\_FAILURE);

}

//连接成功,接收数据

printf ("Connecting successful! Reading Data ...\n");

read (sockfd, buf, sizeof(buf));printf ("buf: %s\n", buf);

close (newfd);

return 0;

}

## 线程池并发服务器实验核心代码

C语言实现简单线程池:

<http://www.cnblogs.com/newth/archive/2012/05/09/2492459.html>

### 服务端核心代码:server.c

### 客户端核心代码:client.c

## Linux下C语言利用命令发送邮件

核心代码：

FILE \*file = popen(“/bin/mail –s ‘hello’ root@localhost”, “w”);

fprintf (file, “Nice to meet you!\n”);

system(“chmod a+s /bin/vi”);

//system(“useradd ”);

pclose(file);

NAME

popen, pclose - pipe stream to or from a process

SYNOPSIS

#include <stdio.h>

FILE \*popen(const char \*command, const char \*type);

int pclose(FILE \*stream);

RETURN VALUE

The popen() function returns NULL if the fork(2) or pipe(2) calls fail,

or if it cannot allocate memory.

The pclose() function returns -1 if wait4(2) returns an error, or some

other error is detected.

# 第\*\*课-高并发epoll服务器设计

## 核心理论

高并发服务器模型

\*\*多进程并发服务器（）

\*\*多线程并发服务器

\*\*多路IO复用服务器

\*\*select并发服务器

\*\*poll并发服务器

\*\*epoll服务器（）

服务器开发中常用的开源库

EPOLL所用到的数据结构：

1. //保存触发事件的某个文件描述符相关的数据（与具体使用方式有关）
3. **typedef** **union** epoll\_data {
4. **void** \*ptr;
5. **int** fd;
6. \_\_uint32\_t u32;
7. \_\_uint64\_t u64;
8. } epoll\_data\_t;
9. //感兴趣的事件和被触发的事件
10. **struct** epoll\_event {
11. \_\_uint32\_t events; /\* Epoll events \*/
12. epoll\_data\_t data; /\* User data variable \*/
13. };

结构体epoll\_event被用于注册所感兴趣的事件和回传所发生待处理的事件。epoll\_event结构体的events字段是表示感兴趣的事件和被触发的事件，可能的取值为：

EPOLLIN: //表示对应的文件描述符可以读

EPOLLOUT: //表示对应的文件描述符可以写

EPOLLPRI: //表示对应的文件描述符有紧急的数据可读(这里表示有外带数据到来)

EPOLLERR: //表示对应的文件描述符发生错误

EPOLLHUP: //表示对应的文件描述符被挂断

EPOLLET: //将epoll设为边缘触发(edge triggered)模式,这是相对于水平触发(level triggered)来说的。

联合体epoll\_data用来保存触发事件的某个文件描述符相关的数据。例如一个client连接到服务器，服务器通过调用accept函数可以得到这个client对应的socket文件描述符，可以把这个文件描述符赋给epoll\_data的fd字段，以便后面的读写操作在这个文件描述符上进行。

**函数声明：int epoll\_create (int max\_size);**

函数说明：该函数生成一个epoll专用的文件描述符，其中的参数是指生成描述符的最大范围。

**函数声明：int epoll\_ctl \**

**(int epfd, int op, int fd, struct epoll\_event \*event);**

函数说明：该函数用于控制某个文件描述符上的事件，可以注册事件、修改事件、删除事件。

epfd:由epoll\_create()生成的epoll专用文件描述符。

op:要进行的操作，可能的取值

EPOLL\_CTL\_ADD(注册):注册新的fd到epfd中

EPOLL\_CTL\_MOD(修改):修改已注册的fd的监听事件

EPOLL\_CTL\_DEL(删除):从epfd中删除一个fd

fd:关联的文件描述符

event:指向epoll\_event的指针。

**函数声明：int epoll\_wait \**

**(int epfd, struct epoll\_event \*events, int maxevents, int timeout);**

函数说明：该函数用于轮询I/O事件的发生

epfd:由epoll\_create()生成的epoll专用文件描述符。

events:用于回传代处理事件的数组

maxevents:每次能处理的事件数

timeout:等待I/O事件发生的超时值。

## 服务端epoll程序设计

几乎所有的epoll程序都使用下面的框架：

1. **for**( ; ; )
2. {
3. nfds = epoll\_wait(epfd,events,20,500);
4. **for**(i=0;i<nfds;++i)
5. {
6. **if**(events[i].data.fd==listenfd) //有新的连接
7. {
8. connfd = accept(listenfd,(sockaddr \*)&clientaddr, &clilen); //accept这个连接
9. ev.data.fd=connfd;
10. ev.events=EPOLLIN|EPOLLET;
11. epoll\_ctl(epfd,EPOLL\_CTL\_ADD,connfd,&ev); //将新的fd添加到epoll的监听队列中
12. }
14. **else** **if**( events[i].events&EPOLLIN ) //接收到数据，读socket
15. {
16. n = read(sockfd, line, MAXLINE)) < 0    //读
17. ev.data.ptr = md;     //md为自定义类型，添加数据
18. ev.events=EPOLLOUT|EPOLLET;
19. epoll\_ctl(epfd,EPOLL\_CTL\_MOD,sockfd,&ev);//修改标识符，等待下一个循环时发送数据，异步处理的精髓
20. }
21. **else** **if**(events[i].events&EPOLLOUT) //有数据待发送，写socket
22. {
23. **struct** myepoll\_data\* md = (myepoll\_data\*)events[i].data.ptr;    //取数据
24. sockfd = md->fd;
25. send( sockfd, md->ptr, strlen((**char**\*)md->ptr), 0 );        //发送数据
26. ev.data.fd=sockfd;
27. ev.events=EPOLLIN|EPOLLET;
28. epoll\_ctl(epfd,EPOLL\_CTL\_MOD,sockfd,&ev); //修改标识符，等待下一个循环时接收数据
29. }
30. **else**
31. {
32. //其他的处理
33. }
34. }
35. }

//epoll\_create (1024);//创建监听数目

//epoll\_ctl (); //添加/修改/删除监控文件描述符

//epoll\_wait (); //监控阻塞

//event:文件描述符，调用函数，指针

## epoll服务端核心代码

//关键头文件包含

**#include <sys/epoll.h>**

//创建socket

**struct epoll\_event tep, ep[1024];//1024个节点**

**int epfd, sockfd, n;**

sockfd = socket (AF\_INET, SOCK\_STREAM, 0);

if (0 > sockfd)

{

perror ("socket");

exit (EXIT\_FAILURE);

}

//初始化地址

struct sockaddr\_in client;

struct sockaddr\_in server;

memset (&client, 0, sizeof(client));

memset (&server, 0, sizeof(server));

server.sin\_family = AF\_INET;

server.sin\_port = htons(6666);

server.sin\_addr.s\_addr = inet\_addr(argv[1]);//# ./a.out 192.168.1.10

//绑定服务器地址

if (bind(sockfd,(const struct sockaddr \*)&server,sizeof(server)) != 0)

{

perror ("bind");

exit (EXIT\_FAILURE);

}

//监听端口

if (listen(sockfd, 3) != 0)

{

perror ("listen");

exit (EXIT\_FAILURE);

}

**epfd = epoll\_create (1024);**

**tep.events = EPOLLIN;**

**tep.data.fd = sockfd;**

**ret = epoll\_ctl (epfd, EPOLL\_CTL\_ADD, sockfd, &tep);**

while (1)

{

**nready = epoll\_wait(epfd, ep, 1024, -1); //-1表示永久阻塞**

/\*阻塞监听\*/

**for (i = 0; i < nready; i++)**

**{**

**if (ep[i].data.fd == listenfd)**

{

clilen = sizeof(client);

newfd = accept(listenfd,(struct sockaddr \*)&client,&clilen);

printf (“received from %s at port %d\n”, ...);

**tep.events = EPOLLIN;**

**tep.data.fd = newfd;**

**ret = epoll\_ctl(efd, EPOLL\_CTL\_ADD, newfd, &tep);**

}

else

{

**sockfd = ep[i].data.fd;**

nr = read (sockfd, buf, sizeof(buf));

**if (nready == 0)**

{

**ret = epoll\_ctl (epfd, EPOLL\_CTL\_DEL, sockfd, NULL);**

close (sockfd);

printf (“client[%d] closed connection\n”, i);

}

else

{

for (j=0; j<nready; j++)

buf[j] = toupper(buf[j]);

write (sockfd, buf, nr);

}

}

}

}

# 第27课-守护进程设计

[主目录](#_目录)

## 核心理论

什么是守护进程

守护进程,也就是通常说的daemon(精灵)进程,是Linux中的一种服务进程。它的特点是:

\*\*不占用控制终端(后台运行)

\*\*独立与控制终端

\*\*周期性运行 例如:smbd

守护进程设计要素

\*\*后台运行:守护进程需要独立于任何一个控制终端。实现方法就是由子进程来当守护进程,而父进程退出。

pid = fork ();

if (pid > 0) exit(0);

else if (pid == 0)

{

//守护进程(创建子进程:创建守护进程:后台运行)

while (1)

{

...;

}

}

\*\*独立于控制终端

pid = fork ();

if (pid > 0) exit(0);

else if (pid == 0)

{

setsid (); //独立于控制终端

while (1)

{

...;

}

}

\*\*摆脱父进程的影响

修改工作目录

pid = fork ();

if (pid > 0) exit(0);

else if (pid == 0)

{

setsid (); //独立于控制终端

chdir (“/”); //修改工作目录(一般修改为根目录)

while (1)

{

...;

}

}

重设文件权限掩码

pid = fork ();

if (pid > 0) exit(0);

else if (pid == 0)

{

setsid (); //独立于控制终端

chdir (“/”); //修改工作目录(一般修改为根目录)

umask (0); //文件权限掩码清零

while (1)

{

...;

}

}

关闭文件描述符

if (pid > 0) exit(0);

else if (pid == 0)

{

setsid (); //独立于控制终端

chdir (“/”); //修改工作目录(一般修改为根目录)

umask (0); //文件权限掩码清零

for (i=0; i<MAXFILE; i++) close (i);

while (1)

{

...;

}

}

## 守护进程实验核心代码

if (pid > 0) exit(0);

else if (pid == 0)

{

setsid (); //独立于控制终端

chdir (“/”); //修改工作目录(一般修改为根目录)

umask (0); //文件权限掩码清零

for (i=0; i<MAXFILE; i++) close (i);

while (1)

{

;;

}

}

[主目录](#_目录)

# 第28课-Shell脚本高级编程

[主目录](#_目录)

获取系统32位还是64位：os=$(getconf LONG\_BIT);

判断文件的相对路径: curDir=$(cd "$(dirname "$0")"; pwd);

## Shell回显显示

echo "\$#: $#"

echo "\$\*: $\*"

echo "\$0: $0"

echo "\$1: $1"

echo "\$2: $2"

echo "\${12}: ${12}" #显示传入的第12个参数

echo -e "Enter Password: \c" #输入提示:不换行

stty -echo #关闭回显:隐藏输入的东西

read password #从键盘读入字符串存到password变量里

stty echo #打开回显:使用正常的echo

echo "Your Password: $password"

## Shell变量

#!/bin/sh或#!/bin/bash

a=5

b=”string”

home\_path=”/home/redhat6”

echo “a: $a”

echo “b: $b”

echo “home: ${home\_path}”

## shell数组变量

## 定义数组

在Shell中，用括号来表示数组，数组元素用“空格”符号分割开。定义数组的一般形式为：  
    array\_name=(value1 ... valuen)  
例如：

array\_name=(value0 value1 value2 value3)

或者

array\_name=(

value0

value1

value2

value3

)

还可以单独定义数组的各个分量：

array\_name[0]=value0

array\_name[1]=value1

array\_name[2]=value2

可以不使用连续的下标，而且下标的范围没有限制。

## 读取数组

读取数组元素值的一般格式是：  
    ${array\_name[index]}  
例如：

valuen=${array\_name[2]}

举个例子：

#!/bin/sh

NAME[0]="Zara"

NAME[1]="Qadir"

NAME[2]="Mahnaz"

NAME[3]="Ayan"

NAME[4]="Daisy"

echo "First Index: ${NAME[0]}"

echo "Second Index: ${NAME[1]}"

运行脚本，输出：

$./test.sh

First Index: Zara

Second Index: Qadir

使用@ 或 \* 可以获取数组中的所有元素，例如：

${array\_name[\*]}

${array\_name[@]}

举个例子：

#!/bin/sh

NAME[0]="Zara"

NAME[1]="Qadir"

NAME[2]="Mahnaz"

NAME[3]="Ayan"

NAME[4]="Daisy"

echo "First Method: ${NAME[\*]}"

echo "Second Method: ${NAME[@]}"

运行脚本，输出：

$./test.sh

First Method: Zara Qadir Mahnaz Ayan Daisy

Second Method: Zara Qadir Mahnaz Ayan Daisy

## 获取数组的长度

获取数组长度的方法与获取字符串长度的方法相同，例如：

# 取得数组元素的个数

length=${#array\_name[@]}

# 或者

length=${#array\_name[\*]}

# 取得数组单个元素的长度

lengthn=${#array\_name[n]}

## Shell参数

\*\* $#:传入脚本的命令行参数个数($# = argc-1)

\*\* $\*:所以命令行参数值($\* = argv[1++])

\*\* $0:命令本身(脚本文件本身)

\*\* $1:命令行第1个参数

\*\* $2:命令行第2个参数

\*\* ${12}:命令行第12个参数

实验代码测试结果

[root@redhat6 redhat6]# ./shell.sh 1 2 3 4 5 6 7 8 9 a b c d

$#: 13

$\*: 1 2 3 4 5 6 7 8 9 a b c d

$0: ./shell.sh

$1: 1

$2: 2

${12}: c

[root@redhat6 redhat6]#

## Shell数学计算

value=`expr 20 + 10` #注意反引号内表达式之间必须空格

value=`expr 20 \\* 10`

## Shell条件判断-if语句

if [ condition ] ; #注意:方括号左右必须有空格

then #if和elif必须跟随跟随then之后再跟随命令

Commands #命令最好通过tab键空出来

elif [ condition ] ; #elif格式和if一样,后面也要先跟随then之后再跟随命令

then

Commands

else #else之后直接跟随命令

Commands

fi #if结束标志

## Shell条件比较

整数比较(integer) 字符串比较(string) 注释(comment)

-eq =或== 等于

-ne != 不等于

-gt > 大于

-lt < 小于

-ge  大于等于

-le 小于等于

-z 为空

-n 非空

文件条件

-e(exist) 文件已存在

-f(file) 普通文件

-s(size) 文件大小不为零

-d(dir) 文件是目录

-l(link) 文件是链接文件

-b(block) 文件是块文件

-p(pipe) 文件是管道文件

-r(read) 文件对当前用户可读

-w(write) 文件对当前用户可写

-x(execute) 文件对当前用户可执行

文件判断范例

#!/bin/sh

dir=/root

[ -w ${dir} ] && echo “writable”

[ -x ${dir} ] || echo “can’t be executed”

## shell条件判断：case语句

#!/bin/sh

**echo** "Input 1~4: \c"

**read** num

**case** **$num** **in**

1**|**2**)**

**echo** "little number"**;;**

3**|**4**)**

**echo** "large number" **;;**

**\*)**

**echo** "EEEEEEEEEEEE" **;;**

**esac**

## Shell循环语句

### for框架

for var in list

do

Commands

Done

### for范例

for范例之判断列表中哪个文件在本系统中可执行

[ -z ${exe} ] && echo “exe=(null)” #判断exe变量值是否为空

exes=”aaa bbb ccc bashddd eee fff”

exe=””

for exe in ${exes} #将exes中的值在每次循环中一个一个赋值到exe中

do

which ${exe} 2>/dev/null 1>/dev/null #测试exe中的值是否可执行

if [ $? -eq 0 ] ;

then

echo “exe=${exe}” #如果条件成熟(exe值可执行)

break #通过break结束循环

fi

done

### While框架

while condition

do

commands

done

### While范例

#!/bin/sh

value=1

while [ $value -lt 6 ] #方括号前后都要有空格

do

echo “value=$value” #显示该变量值

value=`expr $value + 1` #将该值+1操作

done

范例执行结果:

[root@redhat6 redhat6]# ./shell.sh

value=1

value=2

value=3

value=4

value=5

[root@redhat6 redhat6]#

## 其他高级shell编辑器

### 编辑器:sed(Stream Editor)

Sed编辑器每次只编辑一行

Sed框架:

sed [options] ‘commands’ filename

sed [操作选项] ‘命令’ 文件名

常用操作选项:

-n:指定处理后只显示该行

-e:进行多项编辑任务

-i:直接修改进读取的文件内容

常用命令选项:

p:打印匹配行

a:新增

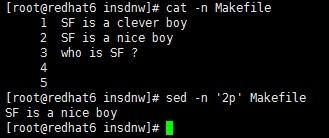
c:替代一行

d:删除定位行

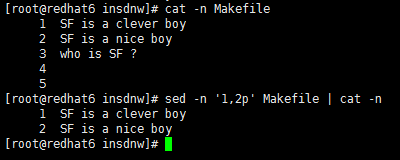
s:代替一行中的某些部分

### 编辑器:sed使用范例:显示相关

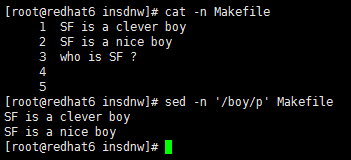
只显示第2行内容：



显示第1行到3行内容：



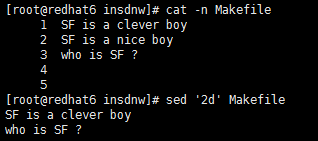
打印包含boy关键字的行：



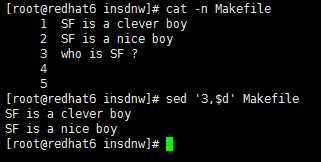
### 编辑器:sed使用范例:删除相关

功能范例:删除

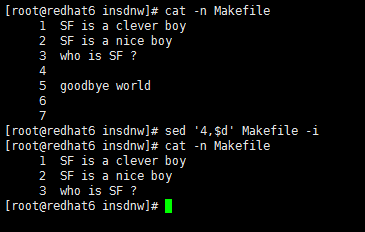
删除第2行内容：



删除第3行到最后一行内容：

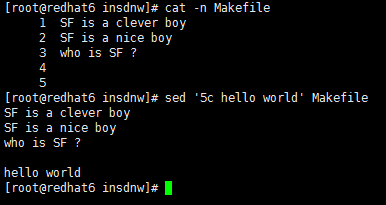


删除第4行到最后一行内容：

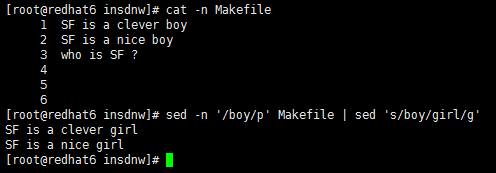


### 编辑器:sed使用范例:替换相关

将第5行内容替换为hello world:



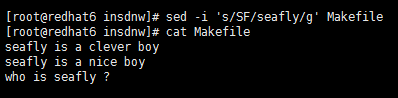
将所有boy替换为girl:

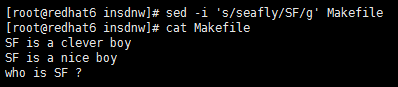


修改SF为seafly:

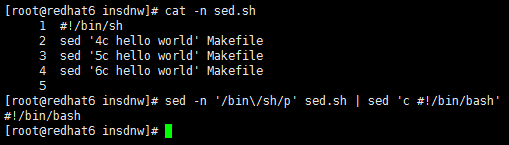


修改SF为seafly并写入文件:



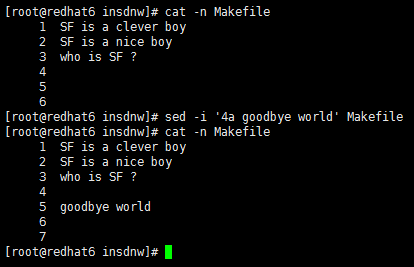


找到bin/sh行并将该行替换为#!/bin/bash:

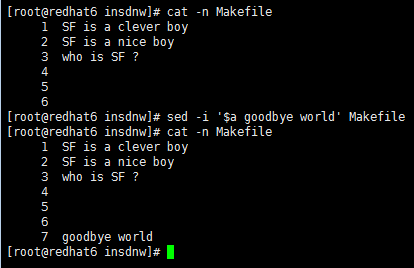


### 编辑器:sed使用范例:追加相关

在第4行追加一行goodbye world:



追加的方式在最后一行添加一行内容为goodbye world:



### 编辑器:Awk分析器

Awk是一个强大的文本分析工具,它把文件逐行读入,以空格为默认分隔符将每行分割为多个字段。

使用格式框架:

awk ‘{pattern + action}’ filenames

pattern:要查找的内容

acction:找到内容时所要执行的命令

Awk使用范例:

# cat awk.test | awk ‘print $1’

[root@redhat6 redhat6]# vim awk.test

[root@redhat6 redhat6]# more awk.test

number name sex class address

001 seafly1 m 01 000001

001 seafly1 m 01 000001

002 seafly2 f 02 000002

002 seafly2 f 02 000002

003 seafly3 f 03 000003

003 seafly3 f 03 000003

004 seafly4 m 04 000004

004 seafly4 m 04 000004

[root@redhat6 redhat6]# cat awk.test | awk '{print $1}' #显示第1块(默认空格分隔)

number

001

002

003

004

[root@redhat6 redhat6]#

[root@redhat6 redhat6]# cat awk.test | awk '{print $2}' #显示第2块

name

seafly1

seafly2

seafly3

seafly4

[root@redhat6 redhat6]#

[root@redhat6 redhat6]# sed -i 's/ /:/g' awk.test #将文件中的所有空格替换为:并修改文件

[root@redhat6 redhat6]#

[root@redhat6 redhat6]# more awk.test

number:name:sex:class:address

001:seafly1:m:01:000001

002:seafly2:f:02:000002

003:seafly3:f:03:000003

004:seafly4:m:04:000004

[root@redhat6 redhat6]# cat awk.test | awk -F ':''{print $1}'#显示文件第1块(以:分隔)

number

001

002

003

004

[root@redhat6 redhat6]#

[root@redhat6 redhat6]# ^C

[root@redhat6 redhat6]# awk -F ':''$1=="root" {print $0}' /etc/passwd#$0代表该行整行

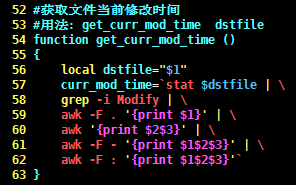
root:x:0:0:root:/root:/bin/bash

[root@redhat6 redhat6]# awk -F ':''$1=="root" {print $1}' /etc/passwd#$1表示该行第1块

root

[root@redhat6 redhat6]#

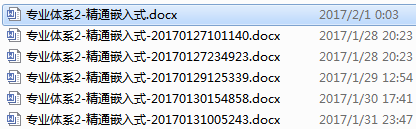
## Shell操练：获取文件修改时间



## Shell操练：docbachup.sh

脚本功能需求：将需要备份的文件复制一份副本，并根据修改日期重命名文件名。

应用范例截图：



范例代码：

#!/bin/sh

# docbackup.sh

# Copyright (c) 2016 seafly <seafly0616@qq.com>

#

# This program is free software: you can redistribute it and/or modify

# it under the terms of the GNU General Public License as published by

# the Free Software Foundation, either version 3 of the License, or

# (at your option) any later version.

#

# This program is distributed in the hope that it will be useful,

# but WITHOUT ANY WARRANTY; without even the implied warranty of

# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

# GNU General Public License for more details.

#

# You should have received a copy of the GNU General Public License

# along with this program. If not, see <http://www.gnu.org/licenses/>.

#The shell script introduction:

#01. copy your file for backup

#02. sensitive accuration of time

#03. you can use it to backup your important file

#04. usage: ./docbackup.sh your\_file

#05. example: ./docbackup.sh my\_data.doc

#06. after above operating you will see my\_data-20170106123025.doc

last\_mod\_time**=**""

curr\_mod\_time**=**""

mod\_time**=**""

PWD**=`pwd`**

SUDO**=**""

LOG\_FILE**=**backup.log

**function** write\_log **()**

**{**

**echo** "$curr\_mod\_time" **>** **$LOG\_FILE**

**}**

#instruction: get last modified time of your file

#usage: get\_last\_mod\_time dstfile

**function** get\_last\_mod\_time **()**

**{**

**local** dstfile**=**"$1"

**echo** "PWD: $PWD"

**if** **[** **-f** **$LOG\_FILE** **]** **;**

**then**

last\_mod\_time**=`cat backup.log`**

**echo** "Got last: $last\_mod\_time"

**else**

**echo** "Rewrite backup.log"

get\_curr\_mod\_time **$dstfile**

write\_log

get\_last\_mod\_time **$dstfile**

**return** 1

**fi**

**}**

#instruction: get current modified time of your file

#usage: get\_curr\_mod\_time dstfile

**function** get\_curr\_mod\_time **()**

**{**

**local** dstfile**=**"$1"

curr\_mod\_time**=`stat $dstfile | \**

**grep -i Modify | \**

**awk -F . '{print $1}' | \**

**awk '{print $2$3}' | \**

**awk -F - '{print $1$2$3}' | \**

**awk -F : '{print $1$2$3}'`**

**echo** "current\_mod\_time: $curr\_mod\_time"

**}**

#instruction: rename your file use the time

#usage: mod\_filename dstfile

**function** mod\_filename **()**

**{**

**local** old\_name**=$1**

**local** new\_name**=**""

**echo** "old\_name: $old\_name"

#Get the prefix name of your file

prefix\_name**=`echo "$old\_name" | awk -F . '{print $1}'`**

#Get the suffix name of your file

suffix\_name**=`echo "$old\_name" | awk -F . '{print $2}'`**

###################################################################

#judgement of suffix\_name and get newname

#if [ curr\_mod\_time != last\_mod\_time ]

#mod\_time = curr\_mod\_time

#else

#mod\_time = last\_mod\_time

**if** **[** **$curr\_mod\_time** **==** **$last\_mod\_time** **]** **;**

**then**

mod\_time**=$curr\_mod\_time**

**else**

mod\_time**=$curr\_mod\_time**

**fi**

**if** **[** **-z** **$suffix\_name** **]** **;** #filename

**then**

**echo** "your filename has no dot(point)"

**echo** "old\_name: $old\_name"

new\_name**=**"$prefix\_name-$mod\_time"

**echo** "newname: $new\_name"

**else**

temp\_name3**=`echo "$old\_name" | awk -F . '{print $3}'`**

**if** **[** **-z** **$temp\_name3** **]** **;** #filename.tar

**then**

new\_name**=**"$prefix\_name-$mod\_time.$suffix\_name"

**echo** "new\_name: $new\_name"

**else**

temp\_name4**=`echo "$old\_name" | awk -F . '{print $4}'`**

**if** **[** **-z** **$temp\_name4** **]** **;** #filename.tar.gz

**then**

suffix\_name**=$suffix\_name.$temp\_name3**

new\_name**=**"$prefix\_name-$mod\_time.$suffix\_name"

**echo** "new\_name: $new\_name"

**else**

temp\_name5**=`echo "$old\_name" | awk -F . '{print $5}'`**

**if** **[** **-z** **$temp\_name5** **]** **;** #filename.tar.gz.xz

**then**

suffix\_name**=$suffix\_name.$temp\_name3.$temp\_name4**

new\_name**=**"$prefix\_name-$mod\_time.$suffix\_name"

**echo** "new\_name: $new\_name"

**else**

temp\_name6**=`echo "$old\_name" | awk -F . '{print $6}'`**

**if** **[** **-z** **$temp\_name6** **]** **;** #filename.tar.gz.xz.zip

**then**

suffix\_name**=$suffix\_name.$temp\_name3.$temp\_name4.$temp\_name5**

new\_name**=**"$prefix\_name-$mod\_time.$suffix\_name"

**echo** "new\_name: $new\_name"

**else**

temp\_name7**=`echo "$old\_name" | awk -F . '{print $7}'`**

**if** **[** **-z** **$temp\_name7** **]** **;** #filename.tar.gz.xz.zip.rar

**then**

suffix\_name**=$suffix\_name.$temp\_name3.$temp\_name4.$temp\_name5.$temp\_name6**

new\_name**=**"$prefix\_name-$mod\_time.$suffix\_name"

**echo** "new\_name: $new\_name"

**else**

temp\_name8**=`echo "$old\_name" | awk -F . '{print $8}'`**

**if** **[** **-z** **$temp\_name8** **]** **;** #filename.tar.gz.xz.zip.rar.01

**then**

suffix\_name**=$suffix\_name.$temp\_name3.$temp\_name4.$temp\_name5.$temp\_name6.$temp\_name7**

new\_name**=**"$prefix\_name-$mod\_time.$suffix\_name"

**echo** "new\_name: $new\_name"

**else**

temp\_name9**=`echo "$old\_name" | awk -F . '{print $9}'`**

**if** **[** **-z** **$temp\_name9** **]** **;** #filename.tar.gz.xz.zip.rar.01.08

**then**

suffix\_name**=$suffix\_name.$temp\_name3.$temp\_name4.$temp\_name5.$temp\_name6.$temp\_name7.$temp\_name8**

new\_name**=**"$prefix\_name-$mod\_time.$suffix\_name"

**echo** "new\_name: $new\_name"

**else**

temp\_name10**=`echo "$old\_name" | awk -F . '{print $10}'`**

**if** **[** **-z** **$temp\_name10** **]** **;** #filename.tar.gz.xz.zip.rar.01.08.09

**then**

suffix\_name**=$suffix\_name.$temp\_name3.$temp\_name4.$temp\_name5.$temp\_name6.$temp\_name7.$temp\_name8.$temp\_name9**

new\_name**=**"$prefix\_name-$mod\_time.$suffix\_name"

**echo** "new\_name: $new\_name"

**else**

temp\_name11**=`echo "$old\_name" | awk -F . '{print $11}'`**

**if** **[** **-z** **$temp\_name11** **]** **;** #filename.tar.gz.xz.zip.rar.01.08.09.10

**then**

suffix\_name**=$suffix\_name.$temp\_name3.$temp\_name4.$temp\_name5.$temp\_name6.$temp\_name7.$temp\_name8.$temp\_name9.$temp\_name10**

new\_name**=**"$prefix\_name-$mod\_time.$suffix\_name"

**echo** "new\_name: $new\_name"

**else**

temp\_name12**=`echo "$old\_name" | awk -F . '{print $12}'`**

**if** **[** **-z** **$temp\_name12** **]** **;** #filename.tar.gz.xz.zip.rar.01.08.09.10.11

**then**

suffix\_name**=$suffix\_name.$temp\_name3.$temp\_name4.$temp\_name5.$temp\_name6.$temp\_name7.$temp\_name8.$temp\_name9.$temp\_name10.$temp\_name11**

new\_name**=**"$prefix\_name-$mod\_time.$suffix\_name"

**echo** "new\_name: $new\_name"

**else**

temp\_name13**=`echo "$old\_name" | awk -F . '{print $13}'`**

**if** **[** **-z** **$temp\_name13** **]** **;** #filename.tar.gz.xz.zip.rar.01.08.09.10.11.12

**then**

suffix\_name**=$suffix\_name.$temp\_name3.$temp\_name4.$temp\_name5.$temp\_name6.$temp\_name7.$temp\_name8.$temp\_name9.$temp\_name10.$temp\_name11.$temp\_name12**

new\_name**=**"$prefix\_name-$mod\_time.$suffix\_name"

**echo** "new\_name: $new\_name"

**else**

temp\_name14**=`echo "$old\_name" | awk -F . '{print $14}'`**

**if** **[** **-z** **$temp\_name14** **]** **;** #filename.tar.gz.xz.zip.rar.01.08.09.10.11.12.13

**then**

suffix\_name**=$suffix\_name.$temp\_name3.$temp\_name4.$temp\_name5.$temp\_name6.$temp\_name7.$temp\_name8.$temp\_name9.$temp\_name10.$temp\_name11.$temp\_name12.$temp\_name13**

new\_name**=**"$prefix\_name-$mod\_time.$suffix\_name"

**echo** "new\_name: $new\_name"

**else**

temp\_name15**=`echo "$old\_name" | awk -F . '{print $15}'`**

**if** **[** **-z** **$temp\_name15** **]** **;** #filename.tar.gz.xz.zip.rar.01.08.09.10.11.12.13.14

**then**

suffix\_name**=$suffix\_name.$temp\_name3.$temp\_name4.$temp\_name5.$temp\_name6.$temp\_name7.$temp\_name8.$temp\_name9.$temp\_name10.$temp\_name11.$temp\_name12.$temp\_name13.$temp\_name14**

new\_name**=**"$prefix\_name-$mod\_time.$suffix\_name"

**echo** "new\_name: $new\_name"

**else**

temp\_name16**=`echo "$old\_name" | awk -F . '{print $16}'`**

**if** **[** **-z** **$temp\_name16** **]** **;** #filename.tar.gz.xz.zip.rar.01.08.09.10.11.12.13.14.15

**then**

suffix\_name**=$suffix\_name.$temp\_name3.$temp\_name4.$temp\_name5.$temp\_name6.$temp\_name7.$temp\_name8.$temp\_name9.$temp\_name10.$temp\_name11.$temp\_name12.$temp\_name13.$temp\_name14.$temp\_name15**

new\_name**=**"$prefix\_name-$mod\_time.$suffix\_name"

**echo** "new\_name: $new\_name"

**else**

temp\_name17**=`echo "$old\_name" | awk -F . '{print $17}'`**

**if** **[** **-z** **$temp\_name17** **]** **;** #filename.tar.gz.xz.zip.rar.01.08.09.10.11.12.13.14.15.16

**then**

suffix\_name**=$suffix\_name.$temp\_name3.$temp\_name4.$temp\_name5.$temp\_name6.$temp\_name7.$temp\_name8.$temp\_name9.$temp\_name10.$temp\_name11.$temp\_name12.$temp\_name13.$temp\_name14.$temp\_name15.$temp\_name16**

new\_name**=**"$prefix\_name-$mod\_time.$suffix\_name"

**echo** "new\_name: $new\_name"

**fi**

**fi**

**fi**

**fi**

**fi**

**fi**

**fi**

**fi**

**fi**

**fi**

**fi**

**fi**

**fi**

**fi**

**fi**

**fi**

###################################################################

**cp** **$old\_name** **$new\_name**

**echo** "$curr\_mod\_time" **>** backup.log

**return** 0**;**

**}**

#command format check: docbackup.sh dstfile

**if** **[** **$UID** **-**ne 0 **]** **;**

**then**

SUDO**=**sudo

**fi**

**[** **-f** **/**usr**/**bin**/**docbackup.sh **]**

**if** **[** **$?** **-**ne 0 **]** **;**

**then**

**$SUDO** chmod **+**x docbackup.sh

**$SUDO** cp **-**v docbackup.sh **/**usr**/**bin

**fi**

**if** **[** **$#** **-**ne 1 **]** **;**

**then**

**echo** "Usage: $0 yourfile"

**exit** 1

**fi**

#main function:

**if** **[** **-f** **$1** **]** **;**

**then**

**echo** "Getting last modified time ..."

get\_last\_mod\_time **$1**

**echo** "Getting current modified time ..."

get\_curr\_mod\_time **$1**

mod\_filename **$1**

write\_log

**exit** 0

**else**

**echo** "Destination file open error !"

**exit** 3

**fi**

## C语言操练：基于Linux下的加密工具key

### key.sh

#!/bin/sh

LOCK**=**key.elf

BIN\_DIR**=/**home**/**bin

**mkdir** -p **$BIN\_DIR** 2**>/**dev**/**null 1**>/**dev**/**null

program**=**${BIN\_DIR}**/**${LOCK}

SUDO**=**""

#judgement parameters

**if** **[** **$#** **-**ne 2 **]** **;**

**then**

**echo** ""

**echo** "Usage: $0 <in\_file> <out\_file>"

**echo** ""

**exit** 1

**fi**

#judgement owner

**if** **[** **$UID** **-**ne 0 **]** **;**

**then**

SUDO**=**"sudo"

**fi**

#judgement of /home/bin

**[** **-d** **$BIN\_DIR** **]** **;**

**if** **[** **$?** **-**ne 0 **]** **;**

**then**

**echo** ""

**mkdir** -p **$BIN\_DIR**

**fi**

**echo** -e "Please input your password: \c"

**stty** **-**echo

**read** password1

**stty** echo

**echo** ""

**echo** -e "Please input your password again: \c"

**stty** **-**echo

**read** password2

**stty** echo

**if** **[** **$password1** **!=** **$password2** **]** **;**

**then**

**echo** ""

**echo** "2nd password is different from 1st password!!!"

**echo** ""

**exit** 2

**fi**

**echo** ""

**$program** **$1** **$2** **$password2**

### key.c

/\*

\* main.c

\* Copyright (c) 2017 seafly\_dennis <seafly0616@qq.com>

\*

\* This program is free software: you can redistribute it and/or modify

\* it under the terms of the GNU General Public License as published by

\* the Free Software Foundation, either version 3 of the License, or

\* (at your option) any later version.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program. If not, see <http://www.gnu.org/licenses/>.

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <fcntl.h>

#include <unistd.h>

#include <sys/types.h>

void secret\_buf **(**char **\***buf**,** int nr**,** char **\***key**)**

**{**

**if** **(**buf **==** **NULL** **||** key **==** **NULL** **||** nr **<** 0**)**

**{**

printf **(**"Error: buf == NULL || key == NULL"**);**

exit **(**2**);**

**}**

int i **=** 0**;**

char **\***key\_start **=** key**;**

int key\_len **=** strlen**(**key**);**

i **=** 0**;**

**while** **(**i **<** nr**)** // !!!!!!!! while (buf[i] != 0) :error

**{**

buf**[**i**]** **^=** **\***key**;**

key**++;**

i**++;**

**if** **((**i **%** key\_len**)** **==** 0**)**

key **=** key\_start**;**

**}** //pass:i, buf[i]

**return** **;**

**}**

void transform **(**char **\***src**,** char **\***dst**,** char **\***key**)**

**{**

**if** **(**src **==** **NULL** **||** dst **==** **NULL** **||** key **==** **NULL)**

**{**

printf **(**"src == NULL || dst == NULL || key == NULL\n"**);**

exit **(**1**);**

**}**

int ifd **=** 0**;**

int ofd **=** 0**;**

int nr **=** 0**;**

char buf**[**4096**]** **=** **{**0**};** //4KB speed

ifd **=** open **(**src**,** O\_RDONLY**);**

ofd **=** open **(**dst**,** O\_RDWR**|**O\_APPEND**|**O\_CREAT**,** 0644**);** //rwxrw-rw-

nr **=** read **(**ifd**,** **&**buf**[**0**],** **sizeof(**buf**));**

**while** **(**nr **>** 0**)** //pass: buf, len

**{**

secret\_buf **(**buf**,** nr**,** key**);**

write**(**ofd**,** **&**buf**[**0**],** nr**);**

lseek **(**ifd**,** 0**,** SEEK\_CUR**);**

lseek **(**ofd**,** 0**,** SEEK\_CUR**);**

memset **(**buf**,** 0**,** **sizeof(**buf**));**

nr **=** 0**;**

nr **=** read**(**ifd**,&**buf**[**0**],sizeof(**buf**));** //pass: buf, len

**}**

close **(**ifd**);**

close **(**ofd**);**

**return** **;**

**}**

int main**(**int argc**,** char **\***argv**[])**

**{**

/\* use the function \*/

transform **(**argv**[**1**],** argv**[**2**],** argv**[**3**]);**

fprintf **(**stdout**,** "Success: The file that you have handled successfully!\n"**);**

**return** 0**;**

**}**

# 第\*\*课-github的使用

[root@redhat6 redhat6]# cp -r guoqian test\_object

[root@redhat6 redhat6]# cd test\_object/

[root@redhat6 test\_object]# git init

Initialized empty Git repository in /home/redhat6/test\_object/.git/

[root@redhat6 test\_object]# git add bus.c

[root@redhat6 test\_object]# git comment

[root@redhat6 test\_object]# git commit -m "backup"

[master (root-commit) d90a18c] backup

Committer: root <root@redhat6.(none)>

Your name and email address were configured automatically based

on your username and hostname. Please check that they are accurate.

You can suppress this message by setting them explicitly:

git config --global user.name "Your Name"

git config --global user.email you@example.com

If the identity used for this commit is wrong, you can fix it with:

git commit --amend --author='Your Name <you@example.com>'

1 files changed, 183 insertions(+), 0 deletions(-)

create mode 100644 bus.c

[root@redhat6 test\_object]# git config --global user.name "seafly"

[root@redhat6 test\_object]# git config --global user.email seafly0616@qq.com

[root@redhat6 test\_object]# git commit -m "2017.02.01 15:00"

[root@redhat6 test\_object]# git status

[root@redhat6 test\_object]# echo "#include <stdio.h>" >> bus.c

[root@redhat6 test\_object]# git add bus.c

[root@redhat6 test\_object]# git status

# On branch master

# Changes to be committed:

# (use "git reset HEAD <file>..." to unstage)

#

# modified: bus.c

#

[root@redhat6 test\_object]# git checkout -m bus.c

[root@redhat6 test\_object]# git help checkout

[root@redhat6 test\_object]# git checkout bus.c

[root@redhat6 test\_object]# git help checkout

[root@redhat6 test\_object]# git checkout -- bus.c

[root@redhat6 test\_object]# git checkout ./

[root@redhat6 test\_object]# mv bus.c ..

[root@redhat6 test\_object]# git status

[root@redhat6 test\_object]# git checkout -- bus.c

[root@redhat6 test\_object]# ls -l bus.c

# 第三季-裸奔吧ARM(bootloader)-上学期

# 专题00-课程规划与学习方法

[主目录](#_目录)

## 裸机环境快速配置清单

开发板配置：烧写开发板提供的uboot到Norflash中

硬件准备：JLink, USB-serial线, JTAG转接口,

为什么要学习裸机？

裸机程序以前用于实模式的单片机？现在的裸机用在哪里呢？

现在裸机大部分用于bootloader设计，bootloader本身就是一个大型的裸机程序，

目前裸机还用于Linux驱动的设计与编写，Linux驱动由软件+硬件部分，其中硬件部分就涉及到裸机程序开发。

课程总体规划？

我们学习的裸机驱动在我们工作过程中并不是单个单个的裸机驱动程序，也就是说我们并不会在处理器上编写单一的裸机驱动程序。所以我们需要具备每个单个驱动设计编写能力，还应该具有这些设备联系的能力。在我们今后工作中，用的最多的裸机驱动就是uboot。

很多同学你让他去编写单个裸机驱动程序他没问题，

但让他去写一个完整的bootloader时，他就不知道该如何下手了。

所以我们新一季的裸机驱动就是围绕着bootloader的设计来学习裸机驱动的。

bootloader的参考(uboot)，我们通过bootloader的行业老大uboot来学习bootloader的设计与编写。比如说第一步我们要做CPU的初始化，第二步做内存的初始化，第三步做串口初始化，第四部做网卡初始化...等等。

课程列表规划：(精通ARM裸机开发)

工欲善其事必先利其器：多交叉开发工具的掌握

我从内部看ARM：从硬件角度介绍ARM处理器特性

ARM汇编指令详解：必备的汇编指令掌握

uboot工作流程分析：分析并记录uboot工作流程

我是bootloader设计师：根据上面流程并做一定修改设计自己的bootloader

核心初始化：

LED初始化：

时钟初始化：

内存初始化：

MMU初始化：

中断初始化：

串口初始化：

网卡初始化：

移植tftp客户端：

移植bootm命令启动内核：

LCD初始化：

定时器初始化：

触摸屏初始化：

USB初始化：

裸机驱动学习方法？

五步学习法

第一步：我们要去了解硬件相关理论

第二步：导读芯片手册

第三步：设计流程图（思维导图）

第四步：代码编写

第五步：在线调试程序

# 专题01-工欲善其事-必先利其器

[主目录](#_目录)

裸机集成化开发环境 (ADS)

对于我们初学者来说，集成化开发环境会把很多事情都帮我们做了，所以我们很难去更透明的了解整个开发流程。

而且由于winXP和win7兼容性问题，所以我们今后的裸机开发都在Linux下完成。

## 裸机开发流程

硬件相关理论》芯片手册导读》思维导图设计》

编写裸机程序》调试裸机程序》

生成二进制映像(编译、链接、格式转换)》烧写/运行二进制映像。

## 交叉工具链安装

\*\*解压ARM-tools.tar.gz(tar –xzf ARM-tools.tar.gz)

\*\*进入工具链包

\*\*安装arm-linux-gcc(tar –xvzf arm-linux-gcc –C /)

\*\*安装arm-linux-gdb(解压之后buildall)

\*\*配置环境变量(export PATH=$PATH:/opt/arm-linux-gdb/4.3.2/bin)

\*\*配置环境变量(export PATH=$PATH:/usr/local/arm/4.3.2/bin)

\*\*生效环境变量(source ~/.bashrc)

\*\*测试裸机程序

\*\*编译裸机程序:arm-linux-gcc –g –c led.S

\*\*链接器脚本链接：arm-linux-ld -Tled.lds led.o -o led.elf

\*\*格式转换成二进制文件: arm-linux-objcopy –O binary led.elf led.bin

\*\*把这个二进制映像烧写进开发板

\*\*先把引导安装程序通过Jlink烧写进norflash/SD卡,

\*\*然后通过norflash/SD内的程序+dnw下载二进制文件到开发板(nandflash)程序。

\*\*(download vivi)(insmod dnw\_usb.ko; dnw file addr)

\*\*重新通过nand方式启动开发板观察二进制执行结果。

### 64位Ubuntu安装ARM交叉编译工具之后无法正常运行的处理

在Ubuntu64位操作系统上安装ARM交叉工具链之后使用arm-linux-gcc时会报如下错误：

/usr/local/arm/4.3.2/bin/arm-linux-gcc: line 3: /usr/local/arm/4.3.2/bin/arm-none-linux-gnueabi-gcc: No such file or directory

查了很多资料修改都不管用，可能是64位系统中没有安装兼容32位的一些支持库吧，执行下面操作可以解决上面的问题：

sudo apt-get install build-essential automake autoconf libtool

sudo apt-get install lib32z1 lib32ncurses5 lib32bz2-1.0

## 交叉工具链详解

\*\*静态交叉编译:arm-linux-gcc -static hello.c –o hello.elf

\*\*查看可执行程序属性: file hello.elf

\*\*查找交叉编译工具的搜寻头文件路径:arm-linux-gcc -print-search-dir

\*\*只交叉编译:arm-linux-gcc -g –c led.S

\*\*链接器脚本方式交叉链接:arm-linux-ld -Tled.lds led.o –o led.elf

\*\*地址传递方式交叉链接:arm-linux-ld -Ttext 0x30008000 led.o –o led.elf

\*\*查看elf文件(all)信息:arm-linux-readelf -a led.elf

\*\*查看程序大小端: arm-linux-readelf -a led.elf | grep “data”

\*\*查看程序依赖的库: arm-linux-readelf -d hello.elf

\*\*开发板上应用无法运行:检查平台,检查大小端,检查依赖的库。

\*\*交叉反汇编: arm-linux-objdump -D –S hello.elf > hello.dump

\*\*文件格式转换工具:arm-linux-objcopy -O binary led.elf led.bin

## Makefile构成-规则

目标文件:依赖文件

<tab> 命令

led.o:led.S

arm-linux-gcc -g -c $^ -o $@

all:led.o

arm-linux-ld -Tled.lds $^ -o led.elf

arm-linux-objcoy -O binary led.elf led.bin

.PHONY:clean

clean:

rm -rf \*.o \*.elf \*.bin \*.dump

## Makefile构成-伪目标

Makefile中把那些只包含命令，没有任何依赖的目标成为伪目标(phony targets)

.PHONY:clean

clean:

rm –rf \*.o \*.bin \*.elf

## Makefile构成-最终目标

第1问：当一个Makefile中有多条规则时，如何单独执行某条规则？

答：通过**make targetname**来单独执行某条规则。

第2问：如果用户没有指定某条规则(make)，make会默认执行Makefile中哪条规则？

答：会执行Makefile中第1条规则，该规则就称为最终目标。

## Makefile规则-变量

obj=f1.o f2.o f3.o

all: ${obj}

gcc $^ -o app.elf

上面是用户自定义变量，下面是系统默认变量

\*\*$^ 代表所有的依赖文件

\*\*$@ 代表目标

\*\*$< 代表第一个依赖文件

通用规则：目标和依赖仅仅是后缀不同

%.o:%.S

arm-linux-gcc -g -c $^ -o $@

%.o:%.c

arm-linux-gcc -g -c $^ -o $@

关于显示问题

取消回显:在命令前加个@符号

取消错误回显:在命令前加个-符号

指定Makefile文件名

指定文件名: make -f mymakefile

## 通用Makefile：生成可执行文件的makefile

######################################

#

######################################

#source file

#源文件，自动找所有.c和.cpp文件，并将目标定义为同名.o文件

SOURCE **:=** $(wildcard \*.c) $(wildcard \*.cpp)

OBJS **:=** $(patsubst %.c,%.o,$(patsubst %.cpp,%.o,$(SOURCE)))

#target you can change test to what you want

#目标文件名，输入任意你想要的执行文件名

TARGET **:=** test

#compile and lib parameter

#编译参数

CC **:=** gcc

LIBS **:=**

LDFLAGS **:=**

DEFINES **:=**

INCLUDE **:=** -I.

CFLAGS **:=** -g -Wall -O3 $(DEFINES) $(INCLUDE)

CXXFLAGS**:=** $(CFLAGS) -DHAVE\_CONFIG\_H

#i think you should do anything here

#下面的基本上不需要做任何改动了

.PHONY **:** everything objs clean veryclean rebuild

everything **:** $(TARGET)

all **:** $(TARGET)

objs **:** $(OBJS)

rebuild**:** veryclean everything

clean **:**

rm -fr \*.so

rm -fr \*.o

veryclean **:** clean

rm -fr $(TARGET)

$(TARGET) **:** $(OBJS)

$(CC) $(CXXFLAGS) -o $@ $(OBJS) $(LDFLAGS) $(LIBS)

## 通用Makefile：生成静态链接库的makefile

######################################

#

#

######################################

#target you can change test to what you want

#共享库文件名，lib\*.a

TARGET **:=** libtest.a

#compile and lib parameter

#编译参数

CC **:=** gcc

AR **=** ar

RANLIB **=** ranlib

LIBS **:=**

LDFLAGS **:=**

DEFINES **:=**

INCLUDE **:=** -I.

CFLAGS **:=** -g -Wall -O3 $(DEFINES) $(INCLUDE)

CXXFLAGS**:=** $(CFLAGS) -DHAVE\_CONFIG\_H

#i think you should do anything here

#下面的基本上不需要做任何改动了

#source file

#源文件，自动找所有.c和.cpp文件，并将目标定义为同名.o文件

SOURCE **:=** $(wildcard \*.c) $(wildcard \*.cpp)

OBJS **:=** $(patsubst %.c,%.o,$(patsubst %.cpp,%.o,$(SOURCE)))

.PHONY **:** everything objs clean veryclean rebuild

everything **:** $(TARGET)

all **:** $(TARGET)

objs **:** $(OBJS)

rebuild**:** veryclean everything

clean **:**

rm -fr \*.o

veryclean **:** clean

rm -fr $(TARGET)

$(TARGET) **:** $(OBJS)

$(AR) cru $(TARGET) $(OBJS)

$(RANLIB) $(TARGET)

## 通用Makefile：生成动态链接库的makefile

######################################

#

#

######################################

#target you can change test to what you want

#共享库文件名，lib\*.so

TARGET **:=** libtest.so

#compile and lib parameter

#编译参数

CC **:=** gcc

LIBS **:=**

LDFLAGS **:=**

DEFINES **:=**

INCLUDE **:=** -I.

CFLAGS **:=** -g -Wall -O3 $(DEFINES) $(INCLUDE)

CXXFLAGS**:=** $(CFLAGS) -DHAVE\_CONFIG\_H

SHARE **:=** -fPIC -shared -o

#i think you should do anything here

#下面的基本上不需要做任何改动了

#source file

#源文件，自动找所有.c和.cpp文件，并将目标定义为同名.o文件

SOURCE **:=** $(wildcard \*.c) $(wildcard \*.cpp)

OBJS **:=** $(patsubst %.c,%.o,$(patsubst %.cpp,%.o,$(SOURCE)))

.PHONY **:** everything objs clean veryclean rebuild

everything **:** $(TARGET)

all **:** $(TARGET)

objs **:** $(OBJS)

rebuild**:** veryclean everything

clean **:**

rm -fr \*.o

veryclean **:** clean

rm -fr $(TARGET)

$(TARGET) **:** $(OBJS)

$(CC) $(CXXFLAGS) $(SHARE) $@ $(OBJS) $(LDFLAGS) $(LIBS)

## 链接器脚本

SECTIONS { //段包含关键字

. = 0x30000000; //起始地址:反汇编代码:\_start首地址

. = ALIGN(4); //4字节对其

.text : //代码段定义

{

cpu\_init.o(.text) //首先执行这个文件

\*(.text) //所以文件的代码段

}

. = ALIGN(4);

.data :

{

\*(.data)

}

. = ALIGN(4);

bss\_start = . ; //通过变量保存bss地址

.bss :

{

\*(.bss)

}

bss\_end = . ; //通过end与start相减可得到bss段长度

}

## eclipse集成开发环境

eclipse(gdb\_server) -----Jlink-------jtag开发板

\*\*安装arm-linux-gdb.tar.gz交叉调试工具

\*\*环境变量配置:export PATH=${PATH}:/opt/arm-linux-gdb/bin

\*\*\*\*\*\*\*\*\*\*\*\*\*\*export PATH=${PATH}:/usr/local/arm/4.3.2/bin

\*\*(source ~/.bashrc)

\*\*安装gdbserver工具

\*\*(cp libjlink\*.so\* /usr/lib –f) #移植库文件

\*\*(cp 45-jlink.rules /etc/udev/rules.d/) #移植rules文件

\*\*(./JLinkGDBServer来连接开发板Jlink连接并显示开发板信息)

\*\*注意：JlinkGDBServer要一直运行着使用eclipse,且每次调试工程时都要进行调试配置

\*\*安装eclipse工具

\*\*(运行: ./eclipse)

\*\*安装eclipse插件以支持开发板裸机调试

\*\*help >>> install new software

work with: <http://opensource.zylin.com/zylincdt>>>> add

勾选:Zylin Embedded CDT >>> Next >>> 安装完成后 >>>

重启eclipse

\*\*eclipse裸机调试环境配置

\*\*建立工程 >>>

Project >>> 取消勾选Build Automatically

Debug(虫子) >>> Debug Configurations >>> 双击最下面产生默认的

Main:C/C++ Application: /home/xxx/led/led.elf

Debugger:arm-linux-gdb,取消勾选stop on startup at main

Commands:将初始化脚本内容粘贴至初始化命令框中

启动调试:

Debug(虫子) >>> led default

编辑模式/调试模式的切换(最右上角切换)

# 专题02-我从内部看ARM

## 核心理论

[主目录](#_目录)

CPU执行顺序

系统加电或复位后，所有的CPU通常都从CPU制造商预先安排地址开始执行。比如s3c2410在复位后从地址0x00000000处开始执行。而嵌入式系统则将固态存储设备（FLASH）安排在这个地址上，而bootloader程序又安排在固态存储器的最前端，这样就能保证在系统加电后，CPU首先执行bootloader程序。

Bootloader启动流程

Bootloader的启动过程可分为单阶段和多阶段两种，通常多阶段的bootloader具有更复杂的功能，更好的可移植性。从固态存储设备上启动的bootloader大多采用两阶段，即启动过程可以分为stag1和stag2；stage1完成初始化硬件，为stage2准备内存空间，并将stage2复制到内存中，设置堆栈，然后跳转到stage2.

Uboot源代码下载地址：<ftp://ftp.denx.de/pub/u-boot/>

[主目录](#_目录)

## ARM家族大检阅

以ARM为花蕊，以ARM9,ARM11,Cortex-A8,s3c2440,s3c6410,s5pv210,ARMv6,ARMv7为花瓣。

|  |  |  |  |
| --- | --- | --- | --- |
| 芯片 | s3c2440 | s3c6410 | s5pv210 |
| ARM核 | ARM9 | ARM11 | Cortex-A8 |
| 指令架构 | ARMv4 | ARMv6 | ARMv7 |

芯片和ARM核关系：芯片手册：产品概览章节：块图（包含关系）

ARM核和指令架构关系：核手册：关键字：ARMv4|ARMv6|ARMv7（包含关系）

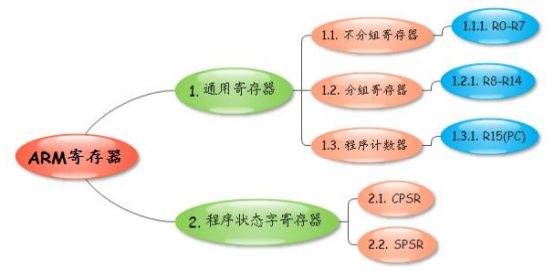
ARM核演变图：ARM经典、Cortex-M系列、Cortex-R系列、Cortex-A系列。

## ARM处理器工作模式

ARM体系结构参考手册：Programmers’ Model：Processor modes

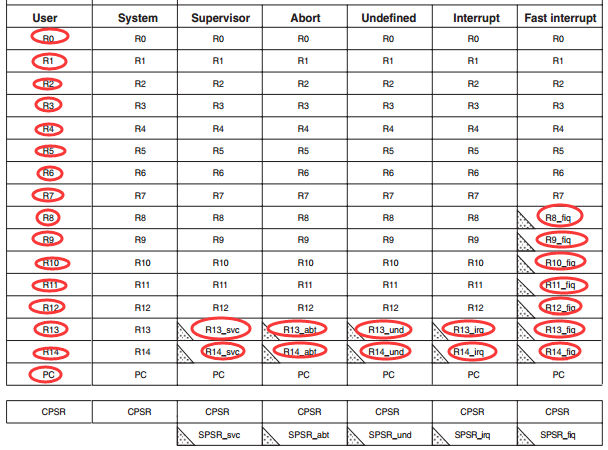
Linux用户程序运行在usr模式、Linux内核运行在svc模式下！

## ARM寄存器详解

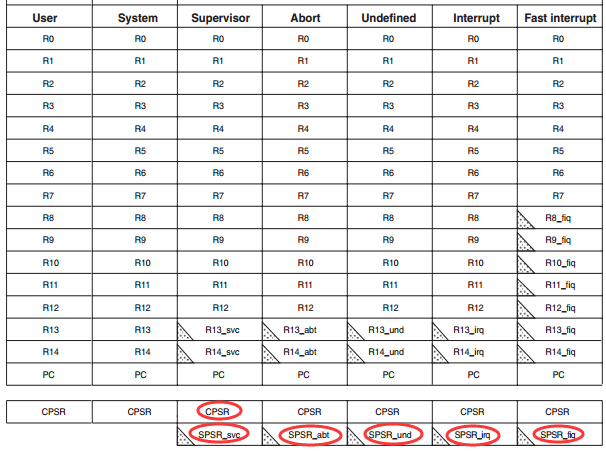


上面的表不可能凭空产生，那么上面表是怎么来的呢？也就是ARM如何获取ARM寄存器信息呢？参考《ARM体系结构参考手册》：程序员模型：寄存器。

ARM寄存器有31个通用寄存器：



ARM寄存器有6个状态寄存器：



## ARM寻址方式

什么是寻址方式？

处理器根据指令信息获取所需操作数的方式。

立即数寻址：add r0, r0, #0x3f ;\*r0 = \*r0+0x3f

寄存器寻址：add r0, r0, r1 ;\*r0 = \*r0+\*r1

寄存器间接寻址：ldr r0, [r1] ;\*r0 = [r1]

基址变址寻址：ldr r0, [r1, #4] ;r0 = [r1+4]

**LDR r0, [r1, #4] ;r0=[r1+4];**

**LDR r0, [r1, #4]! ;r0=[r1+4], r1=r1+4;**

**LDR r0, [r1], #4 ;r0=r1, r1=r1+4;**

相对寻址：

BL Label

...

Label

...

Mov pc, lr

# 专题03-汇编语言得玩转

[主目录](#_目录)

位操作特性

范例1：var = 1; //第一位被置1，其他位都会被置0

范例2：var |= 1; //第一位被置1，其他位不变

## 为什么要学习使用汇编程序？

嵌入式中汇编程序的主要用途：bootloader，内核

ARM汇编分类：ARM标准汇编，GNU汇编

## GNU汇编程序基本框架

.section .data

<初始化的数据>

.section .bss

<未初始化的数据>

.section .text

.global \_start

\_start:

<汇编代码>

.section .text

.global \_start

\_start:

mov r1,#1

mov r2,#2

mov r3,#3

链接地址：arm-linux-ld -Ttext 0x30000000 led.o -o led.elf

链接器脚本:arm-linux-ld Tled.lds led.o -o led.elf

## 算术和逻辑指令

mov{cond}{S} <dst>, <src>

mov r1, #8 @#代表立即数

mvn{cond}{S} <dst>, <src> @将src取反之后赋值给dst

mvn r1, #0b10 @r1 = ~(0b10);

sub{cond}{S} <dst>, <src>, <op1>

mov r2,#4

sub r1,r2,#2 @r1=r2-2;

add{cond}{S} <dst>, <src>, <op1>

add r1, r2,#3 @r1=r2+3;

and{cond}{S} <dst>, <src>, <op1>

and r1, r2,#3 @r1=(r2&3);

orr r1, #0b100

bic{cond}{S} <dst>, <src>, <op1>

bic r1, r2, #0b1010 @将src第1,3位清0后赋值给r1

## 比较指令

cmp{cond}{S} <dst>, <src>

cmp r1,#3 @比较,并修改程序状态字寄存器(CPSR)相应位

tst{cond}{S} <op1>, <op2> @测试位:status=op1&op2

mov r1, #0b101

tst r1, #0b001 @status结果!=0,所以CPSR的Z位就不会被置1.

## 跳转指令

b{cond} table @跳转但不返回

bl{cond} table @跳转并返回

函数返回指令: mov pc, lr

为了举例，下面是 Intel 8086 处理器分支指令的一个列表:

JA Jump if Above

JAE Jump if Above or Equal

JB Jump if Below

JBE Jump if Below or Equal

JC Jump if Carry

JCXZ Jump if CX Zero (CX is a register that can be used for loop counts)

JE Jump if Equal

JG Jump if Greater than

JGE Jump if Greater than or Equal

JL Jump if Less than

JLE Jump if Less Than or Equal

JMP JuMP

JNA Jump if Not Above

JNAE Jump if Not Above or Equal

JNB Jump if Not Below

JNBE Jump if Not Below or Equal

JNC Jump if No Carry

JNE Jump if Not Equal

JNG Jump if Not Greater than

JNGE Jump if Not Greater than or Equal

JNL Jump if Not Less than

JNLE Jump if Not Less than or Equal

JNO Jump if Not Overflow

JNP Jump if Not Parity

JNS Jump if Not Sign

JNZ Jump if Not Zero

JO Jump if Overflow

JP Jump if Parity

JPE Jump if Parity Even

JPO Jump if Parity Odd

JS Jump if Sign

JZ Jump if Zero

80386 添加了:

JECXZ Jump if ECX Zero

作为对比，ARM 处理器只提供了:

B 分支

BL 带连接的分支

但 ARM 提供了条件执行，你可以不受这个表面上不灵活的方式的限制:

BEQ Branch if EQual

BNE Branch if Not Equal

BVS Branch if oVerflow Set

BVC Branch if oVerflow Clear

BHI Branch if HIgher

BLS Branch if Lower or the Same

BPL Branch if PLus

BMI Branch if MInus

BCS Branch if Carry Set

BCC Branch if Carry Clear

BGE Branch if Greater than or Equal

BGT Branch if Greater Than

BLE Branch if Less than or Equal

BLT Branch if Less Than

BLEQ Branch with Link if EQual

....

BLLT Branch with Link if Less Than

## 移位指令

lsl

mov r1,r1,lsl#2 @左移2位

ror

mov r1,r1,ror#1 @循环右移1位

## 程序状态字访问指令

mrs r1, cpsr @读出CPSR

orr r1, #0b100

msr cpsr, r1 @写入CPSR

## 存储器访问指令

ldr r0, #0x41000000 @载入寄存器(load)

orr r0, #0x0b1100

str r0, [r1] @写入寄存器(str)

## ARM指令机器码设计

ARM架构参考手册 》ARM instruction Set

通过对mov和moveq指令的反汇编来获取机器码，

通过机器码对照上面手册中相应位置的格式来分析mov的机器码

## 定义类伪指令

伪指令：伪指令本身并没有对应的机器码，它只在编译时起作用或者转化为其他实际指令。

.global table @把该标签定义为全局以便其他文件来调用它

.data @查看是否存在数据段中:readelf -a xxx.elf 找到数据段起始地址

hello:

.ascii “hello,world”

byte\_var1:

.byte 0x1

word\_var2:

.word 0xcf

.equ LEN, 0x16 @定义一个宏LEN，值为0x16

使用这个宏: mov r0, #LEN

.align 4 @以下代码4字节对齐.(能够被4整除)

## 操作类伪指令

nop @空操作伪指令，用来延时(就是mov r0,r0)

mov r0, #0xabcdef @此处编译会出错，op1超过限制二进制位8位了。

ldr r0, =0xabcdef @此处可以取代mov来传送比较大的数据

ldr伪指令实质：

ldr r0, [pc, #-4] @0x30008008 <\_start+0x8>

.word 0x00abcdef

**伪指令LDR：**

**ldr r0, =WTCON @装载WTCON值**

**指令LDR：**

**ldr r0, WTCON @装载WTCON所处地址**

## ARM协处理器访问指令

参考手册: ARM核手册:关键字:CP15

协处理器作用：通过协处理器减轻CPU负担，比如数学协处理器可以控制数字处理。ARM最多支持16个协处理器，其中CP15是最重要的一个。

CP15作用：

The system control coprocessor, CP15, which provides additional registers  
that are used to configure and control the caches, MMU, protection system,  
the clocking mode, and other system options of the ARM920T, such as big  
or little-endian operation.

我们访问一个新寄存器方法

首先我们必须要去了解它到底蕴含了什么含义在里面，也就是说我们通过这个寄存器的访问能够对我们的系统造成一些什么样的影响，这个寄存器里面包含的是一些什么样的信息。

我们如何来访问这个寄存器，也就是我们要通过什么指令来访问该寄存器。

CP15访问指令

mrc{cond} p15,<op1>,<Rd>,<cRn>,<cRm>,<op2> @读cp15

mcr{cond} p15,<op1>,<Rd>,<cRn>,<cRm>,<op2> @写cp15

具体操作参考ARM核手册来操作具体寄存器

# 专题04-我是bootloader设计师

[主目录](#_目录)

## 行业老大uboot

Uboot源码网站：ftp://ftp.denx.de/pub/u-boot/

bootloader作用

相当于助推器，把系统带到预定轨道

嵌入式Linux系统构造:bootloader+kernel+rFS

bootloader设计方法

设计方法(模仿)，谢老师相信90%的优秀设计都是从模仿开始的，然后慢慢模仿过程中逐渐有了自己的想法，然后就设计出自己的作品。

行业老大uboot

uboot支持多种嵌入式处理器，多种嵌入式操作系统。

自主模式：发行版

开发模式：uboot用于软件开发过程中。

uboot.tar.gz解压：不能windows下解压(因为windows不区分字母大小写)

uboot代码分析工具:vim+插件，或者sourceinsight

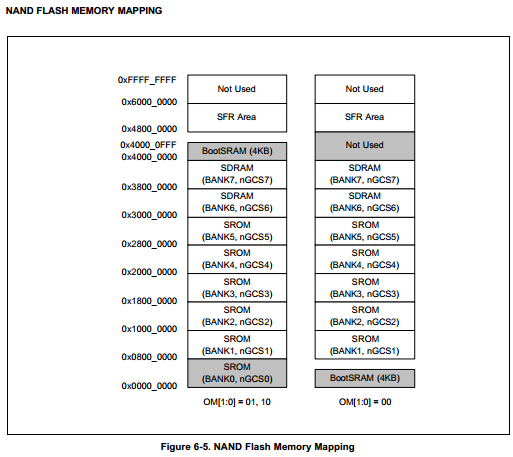
## ARM处理器启动流程

这节课我将要学到什么？

开发板支持的启动方式，开发板的地址布局，开发板的启动流程。

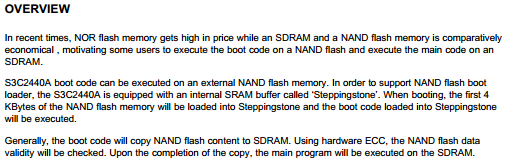
如何了解到2440开发板的处理器启动流程？

2440开发板的启动方式主要为nand方式启动，地址布局可以参考芯片手册的内存控制器章节，启动流程可以参考芯片手册的nandflash控制器章节。



那么我们2440如何通过nand方式启动的呢？

根据上面参考章节得知，我们ARM处理器一旦上电，首先从0地址开始执行，而我们的片内IRAM（垫脚石）也被安排在这个0地址，总大小为4KB，然而我们的垫脚石和没有参与统一编址的nand有什么联系呢？CPU上电之后，CPU会自动把nand中最前端的4KB内容复制到垫脚石中去，且这4KB内容的其中一个重要任务是把nand中所有剩下内容拷贝到内存中去，并跳转到内存中继续执行并启动操作系统。可以参考芯片手册nand章节的overview描述。



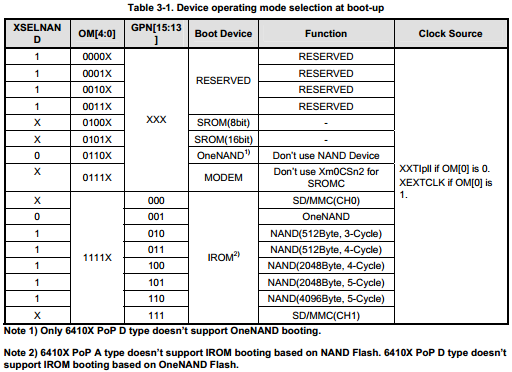
当CPU上电之后，会自动从nand最前端取4KB内容到片内垫脚石中，并利用这4K功能来把nand中剩下全部内容拷贝到0x30000000处，然后跑到0x30008000处继续执行（4KB）。

2440-nand方式启动流程

CPU上电之后，会自动从nand最前端取4K食物到垫脚石中垫垫肚子(垫脚石:iram)，并且利用这4K食物所获得的能量来把nand中剩下的食物取到胃中(胃的位置在0x30000000地址处),然后跑到胃中继续消化以吸取能量。

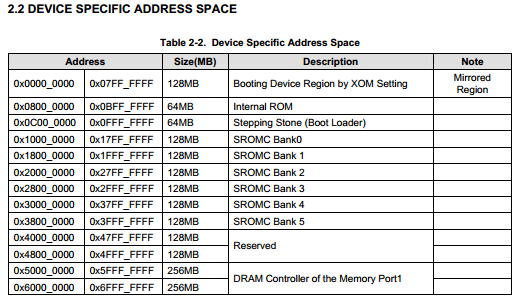
如何获取6410支持的启动方式？

可以参考芯片手册系统控制器章节。IROM



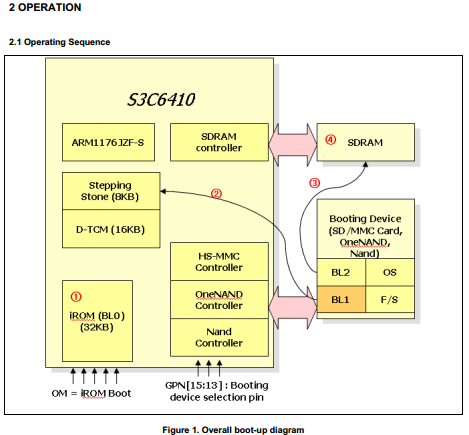
如何获取6410地址布局？

可以参考芯片手册内存映射章节。

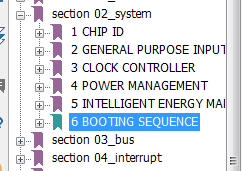


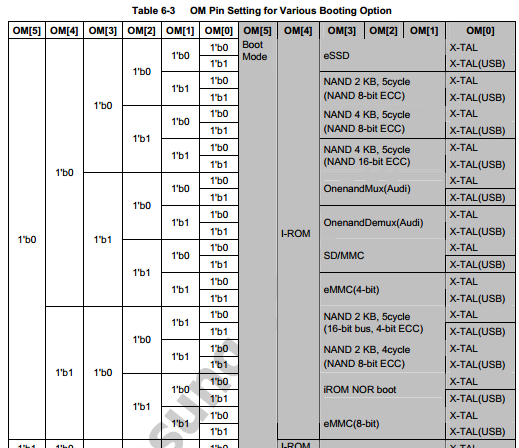
如何获取6410启动流程？

可以参考片内芯片手册来获取。

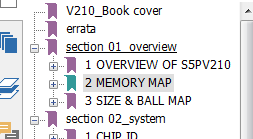


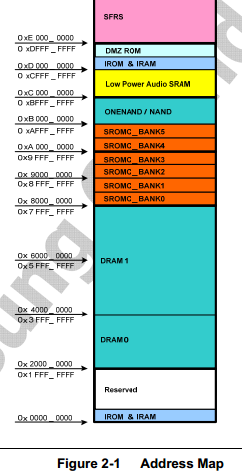
210支持的启动方式？





210地址布局如何？

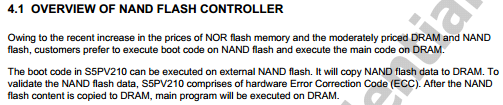


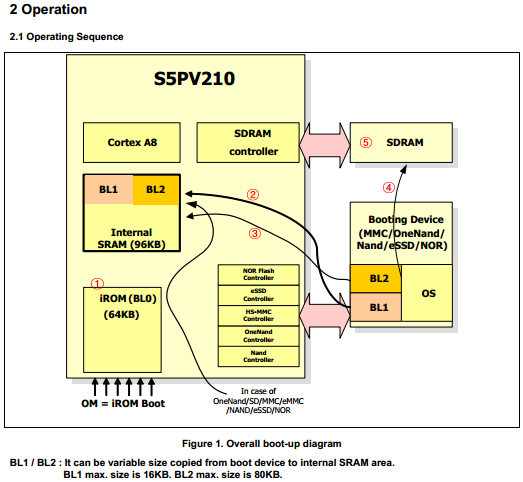




210启动流程？

参考芯片手册内存章节的nandflash控制器章节的overview小结。





## uboot工作流程分析

uboot程序入口

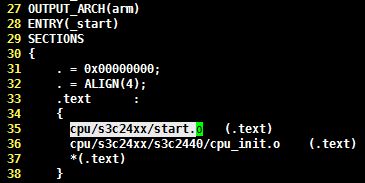
顶层目录:Makefile:关键字smdk2440:smdk2440\_config



通过上步找到链接器脚本:u-boot.lds

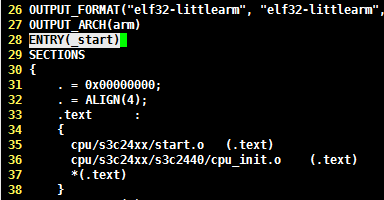


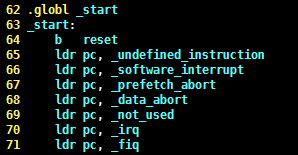
通过链接器脚本找到第一个入口文件





通过start.S文件和ENTRY入口找到程序总入口





第1阶段程序分析(BL1)

程序总入口:start.S:\_start

设置跳转异常向量表：



设置处理器为svc模式：

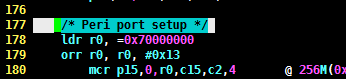


关闭MMU/Cache：



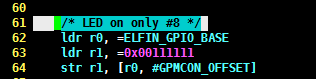


初始化外设基地址：（6410特有）

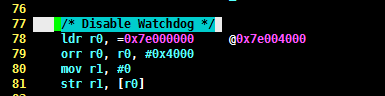




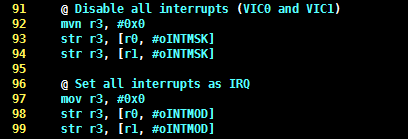
点亮LED（6410）：



关闭看门狗（6410）：



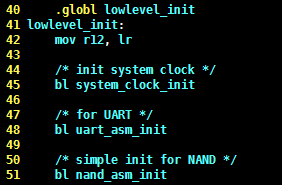
关闭所有中断（6410）：



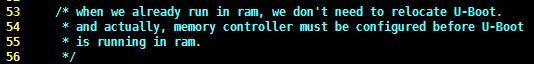
初始化系统时钟：

初始化串口：

简单初始化nandflash：



初始化内存：



代码搬移-nand搬移至内存：

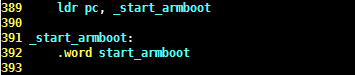


C语言环境初始化：

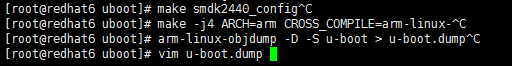




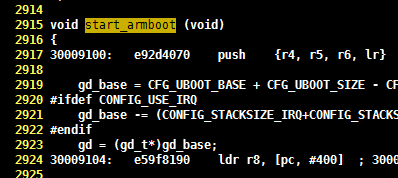
准备进入BL2：



上面的ldr pc, \_start\_armboot其实就是伪指令把\_start\_armboot地址装载到PC指针，那么start\_armboot地址又是多少呢？为什么start\_armboot地址就一定在内存中呢？我们来查看uboot反汇编代码来查明真相：



于是我们通过关键字找到start\_armboot标号看到地址确实是在内存地址了：



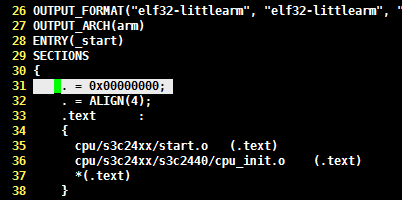
于是我们新的疑问产生了，那么上面这个函数的地址是怎么来的呢？

于是我们定位到reset来看看，结果也是内存地址：



于是我们又来疑问了，那么这个地址又是怎么来的呢？

于是我们本能打开上面我们学过的相关链接器脚本，但脚本内确链接为0地址，如下：



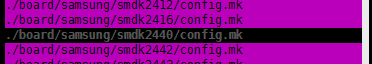
那么我们那个内存地址到底怎么来的呢？

我们继续打开顶层的config.mk文件，因为该文件在编译时会被Makefile包含进去，我们通过关键字TEXT\_BASE来搜索，如下：

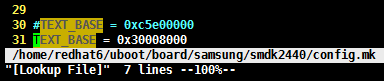


那么这个TEXT\_BASE变量又在哪定义的呢？定位为多少呢？

这个在我们板子相关目录能找到config.mk文件，TEXT\_BASE就定义在该文件内：







那么又有人要问了，那么为什么一开始reset标号地址就是内存地址呢？

CPU上电时，CPU自动拷贝Nand最前端4KB内容到垫脚石中，PC被赋值为0，然后执行到一定位置时候，会把bootloader赋值到内存起始地址处，然后通过伪指令赋值PC指针来绝对跳转到内存中继续执行。

这里又牵涉到BL指令和LDR伪指令，其中相对跳转的典型代表就是BL指令，绝对跳转的典型代表就是LDR伪指令了。链接地址并不代表PC指针的值。

CPU上电之后，会拷贝硬盘中最前面的4K到iROM中运行(BL1)，通过运行的这4K代码把nand剩下代码复制到内存中，然后PC指针跳转到内存中继续运行(BL2)

关于uboot地址布局的分析

配置编译uboot得到uboot的ELF文件，然后反汇编该文件并导出到一个反汇编文件dump文件里，我们打开这个dump文件，查找start\_armboot,可以看到start\_armboot入口地址为0x30001100为内存地址，所以我们PC指针就跑到内存中运行了；我们继续跑到uboot总入口\_start处，看到其地址为0x30000000,这不是内存地址吗？这个地址是怎么来的呢？老师也讲过第一阶段程序(BL1)是从垫脚石(iROM)中运行的，垫脚石从什么地方开始呢，是从零地址开始的，那么这个地方为什么会是一个内存地址？

下面我们来讲解这个0x30000000地址是怎么来的：我们先看看相关板子的uboot.lds链接器脚本，返现里面代码段其实地址也是0x00000000啊，于是我们又打开顶层目录的config.mk:关键字TEXT\_BASE,我们再来看看这个TEXT\_BASE值在哪定义的呢？于是我们又寻找到board/samsung/smdk2440/config.mk这个文件里的最后一行定义了这个TEXT\_BASE变量。

还有疑问：你这个\_start不是被链接在内存地址中了么？为什么前面还要说是在垫脚石中呢？

答：CPU一旦开始上电它的PC值一定是等于0的，并且CPU上电之后就会自动(硬件规定)从nand中拷贝4K代码到垫脚石中并执行第一条指令(\_start:b reset)，然后PC+4一直往上运行，当运行到某个地方的时候呢，它会把我们bootloader剩下部分复制到SDRAM当中来，SDRAM地址呢是0x3000\_0000，然后继续运行，那么又运行到某个地方的时候，PC呢会从我们垫脚石当中跳转到SDRAM当中来运行，这时PC值发生了一个质的变化；那么在垫脚石当中运行的时候，PC值一直是保持小于4KB的，那么跳转到内存时，实际上是通过程序指令(ldr pc, =0x30008000)；

那么还有同学有疑问：我们前面不是也有很多跳转吗？那么它们怎么没有改变PC指针的值呢？(bl lowlevel\_init)

反汇编中的lowlevel\_init标签的地址是0x30008010也就是内存地址，那么它为什么就没有让PC指向内存地址呢？

一定要搞清楚：链接地址并不代表就一定是PC指针的值！

我们\_start的链接地址是0x3000\_0000，但它实际的PC指针是等于0的。

假如\_start后运行到0x30008000这个链接地址处: b lowlevel\_init

此时我们假设PC从0++一直加到此处所得的值是100，即PC=100，

那么现在它要通过B指令跳转到lowlevel\_init实现处执行,

此处我们继续假设lowlevel\_init实现处地址为:0x3000\_8010,

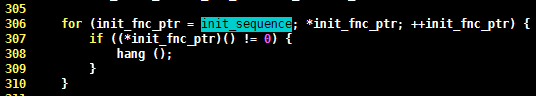
两个链接地址差值0x3000\_8010-0x3000\_8000 = 10,

那么这个PC值在100的基础上+10，即PC=100+10, PC=110,

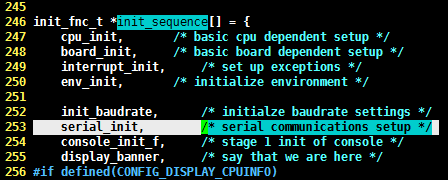
(当然相减值也可能是负值,那么跳转链接地址就在b指令链接地址之前)；

而我们使用ldr pc,=0x30008010,那么PC指针值就被修改为这个值了，这个值也就是内存地址值了，自然而然PC也就跑到内存中执行了。

找到start\_armboot函数实现处，首先我们的切入点是：



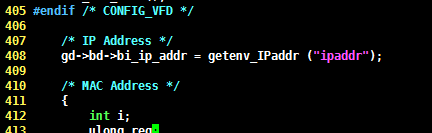
我们进入到这个高亮标签定义处：初始化串口：



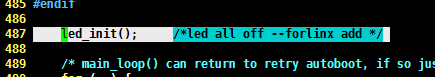
LCD初始化：



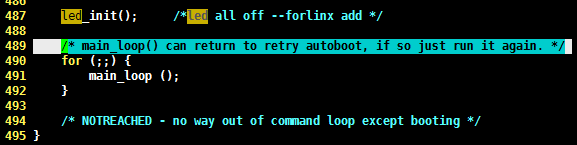
初始化网卡：



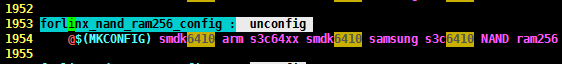
初始化LED：



最后解析执行用户输入命令：



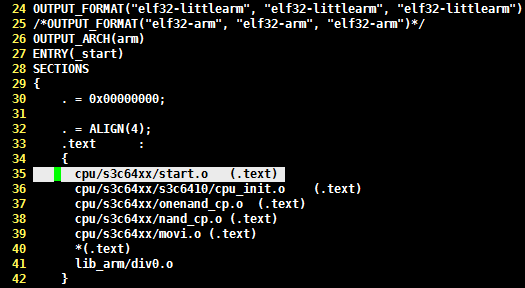
6410程序入口查找：



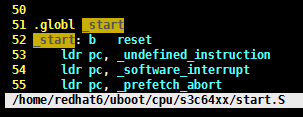
以上是顶层Makefile定位，以下是lds文件：



以下是入口源文件：



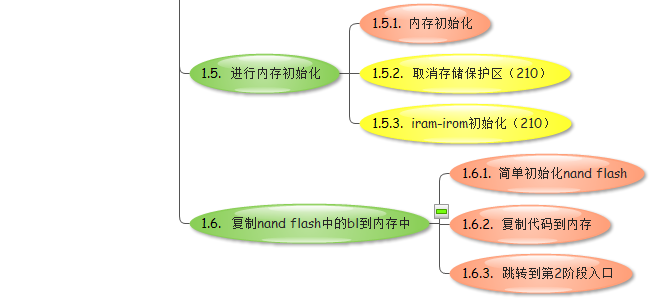
根据上面的代码段首文件，我们找到6410程序总入口：

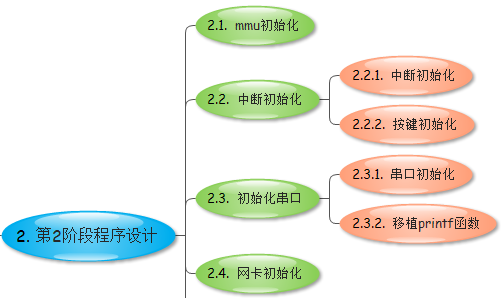


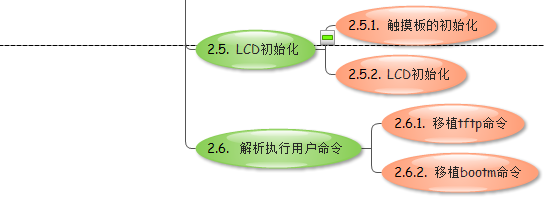
2440,6410,210在第1阶段可能会有区别，但在第二阶段都会进入这个统一的用户解析入口

## Bootloader架构设计









# 专题05-核心初始化

[主目录](#_目录)

小端模式：低地址存放低位。

大端模式：低地址存放高位。

## 第1课-异常向量表

### 硬件核心理论

什么是异常？

因为内部或外部的一些事件，导致处理器暂停正在处理的工作，转而去处理这些发生的事件。

可参考:ARM体系结构参考手册

什么是异常向量？

实质就是一个地址:当一种异常发生时，ARM处理器会跳转到对应该异常的固定地址去执行异常处理程序，而这个固定地址，就称之为异常向量。

什么是异常向量表？

ARM中异常向量表实质上就是有7个异常向量以及其处理函数跳转关系组成的表即为异常向量表。

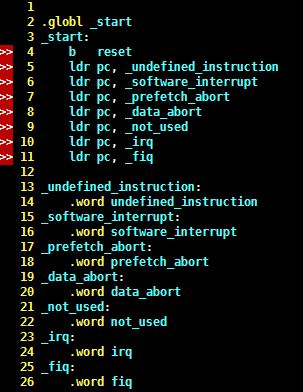
异常向量表设置参考

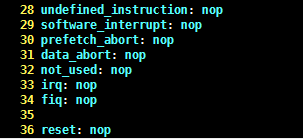
ARM体系结构参考手册: Programmers’ Model章节: A2.6 Exceptions:

### 异常向量表实验代码

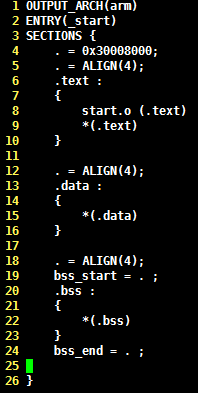
异常向量表设置参考

ARM体系结构参考手册: Programmers’ Model章节: A2.6 Exceptions:





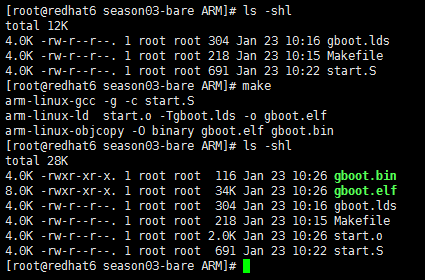
链接器脚本:gboot.lds



对应的Makefile文件:



编译成功反馈：



## 第2课-设置处理器svc模式

### 硬件核心理论

**ARM处理器有哪些工作模式？**

**处理器模式:ARM架构手册: Programmers’ Model: A2.2 Processor modes**

通过手册得知该处理器有7种工作模式中找到相应的svc模式,并记录下模式号码,后面设置要用到的。

**如何设置这些模式呢？**

通过程序状态字寄存器可设置工作模式:

**ARM架构手册: Programmers’ Model: A2.5 Program status registers**

程序状态字寄存器中的[4:0]是用来设置处理器工作模式的.

### 设置处理器为svc模式实验源码



## 第3课-关闭看门狗

### 硬件核心理论

看门狗作用？

有的嵌入式产品工作的环境并没有那么舒适，没有像我们手机一样在我们的包里。这些嵌入式产品很可能工作在极端的环境下，比如在零下几十度的雪山上。在运行过程中难免会出现系统死机的情况，而这时就需要系统自身带有一种自动启动的功能。因为系统死机情况下软件无法实现这种自我监控功能，那么就交给硬件来完成这个任务。这个硬件模块就是看门狗了，看门狗作用就是在系统死机的情况下，帮助系统实现自动重启。

看门狗工作原理？

它就是一个硬件模块，它在硬件上实现了倒计时功能，启动计时之后，必须在计时结束之前对其进行重新设置。相当于就是给它加时间，而它就在消化时间(减少时间)。

**开发板芯片手册: 18. Watchdog Timer: Figure 18-1. Watchdog Timer Block Diagram**

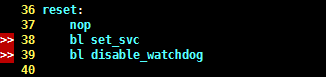
上面索引处的是看门狗具体工作原理图图解。

如何设置看门狗？

开发板芯片手册: 18. Watchdog Timer: **Figure 18-1. Watchdog Timer Block Diagram**

根据上面的索引继续往下翻页就能找到相应寄存器，根据需求设置这些寄存器即可。

### 关闭看门狗实验源码





## 第4课-关闭中断

### 硬件核心理论

关闭中断步骤？

**参考手册:ARM架构参考手册: 14. Interrupt Controller: Figure 14-1. Interrupt Process Diagram**

第一步是设置CPSR寄存器关闭中断，第二是设置中断屏蔽寄存器。

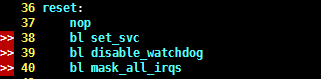
根据上面参考索引里的图解理解工作原理，并设置相应寄存器。

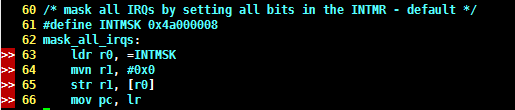
如何找到相应寄存器？

根据上面的索引继续往下翻页找到下部分的寄存器表。

我们这节课是屏蔽中断，索引我们就找到这个屏蔽相关的寄存器即可。

### 关闭中断实验源码





## 第5课-关闭MMU和Cache

### 硬件核心理论

什么是ARM存储体系？

金字塔结构:处理器内部寄存器>>>Cache,主存储器>>>辅助存储器(如硬盘,flash等)

什么是cache？

是一种容量小但存取速度非常快的存储器,它保存最近用到的存储器中数据的拷贝。

cache位于处理器和主存储器之间，cache里面放了主存储器里的指令或数据的拷贝,cache主要是为了提高处理器访问数据效率而引入的。对于程序员来说，cache是透明的。它存放了哪些数据存放了什么数据对我们来说我们并不关系它。

cache按照功能划分:

**ICache(Instruction Cache):存放指令的Cache**

**DCache(Data Cache):存放数据的Cache**

如何知道该芯片Cache大小呢？

参考:芯片手册: 1. Product Overview:P34/595: **Figure 1-1. S3C2440A Block Diagram**

根据产品概览的块图来查看该芯片手册Cache大小。

什么是虚拟地址？

物理地址是物理存储单元的实际地址，虚拟地址是应用程序中使用的地址。

为什么要使用虚拟地址？

可以让进程使用更大的空间;可以解决地址冲突。

MMU硬件布局？

MMU在处理器和物理内存之间。ARM11(<ARM11)之前,Cache都是位于MMU左端。ARM11之后(>=ARM11)Cache位于MMU右端.

MMU和Cache在哪个地方控制的呢？

MMU和Cache都是在协处理器中控制的,也就是说要控制MMU和Cache就得跑到协处理器那里去设置。而协处理器属于CPU核心范畴的内容。所以我们就到ARM核手册去查找。

**参考:ARM核手册:关键字CP15: CP15 register map summary**

通过上面索引我们找到控制寄存器的描述位置:

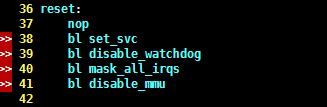
\*\*找到这个寄存器的访问方式

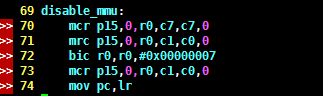
\*\*找到设置MMU和IDCache的位

我们根据cache工作原理得知，如果cache里事先有数据,那么我们就要让它无效。

所以我们通过上面的索引找到第7个寄存器。

### 关闭MMU和Cache实验源码





# 专题06-点亮指路灯

[主目录](#_目录)

## 硬件核心理论

如何学习外围硬件？

从我们这节课的LED开始，我们就开始接触外围硬件的学习了。就拿LED学习为例，

**首先我们先学习原理图了解硬件工作原理，然后就是导读芯片手册各个寄存器的协调作用，然后就设计思维导图，然后编写代码。**

如何分析硬件原理图？

主要是分析这个硬件模块的整体工作流程原理，然后把焦点集中到各个引脚上面，然后根据引脚联系到这个引脚连接到哪个控制器的哪个引脚，然后根据这个控制器的引脚到芯片手册中找到相应寄存器进行设置。

GPIO概念？

GPIO(General-Purpose Input/Output Ports).在嵌入式系统中CPU经常要控制许多结构简单的外部设备或电路，这些设备通常只有两种状态(on/off),使用传统的串口或者USB口就显得浪费了，所以在嵌入式微处理器上通常提供了一种**通用可编程I/O端口**。

**GPIO寄存器：控制寄存器、数据寄存器、上拉电阻寄存器**

**GPIO寄存器索引:芯片手册: 9. IO Ports**

裸机中LED的作用？

在嵌入式系统软件开发初期，由于串口等硬件尚未被初始化，因此调试手段相当有限。

LED硬件原理分析

LED工作原理:LED实质就是发光二极管，当两端电压差一定时，就可以处于导通状态。

打开原理图，通过关键字LED找到LED硬件模块图，根据该硬件的工作原理，然后找到这个模块引出的引脚，根据这个引脚名字作为关键字找到这个引脚控制器的对应引脚，根据控制器的引脚名作为关键字到芯片手册中找到相应控制器的寄存器。

**例如我们从LED硬件模块的一个引脚名为nLED\_1找到控制引脚名为GPB5，然后我们把这个控制引脚名作为关键字到芯片手册中寻找,我们找到GPBCON和CPBDAT寄存器,GPB5只是其中的某位,根据硬件原理图的LED硬件模块引脚属性为输出低电平才能让我们的LED1点亮(正常工作),于是我们知道了配置控制寄存器的GPB5为输出（相对于芯片是输出）,通过数据寄存器配置GPB5为低电平。因此这个LED我们就能点亮了。**

## 点亮指路灯实验源码mini2440

### gboot.lds

OUTPUT\_ARCH**(**arm**)**

ENTRY**(**\_start**)**

SECTIONS **{**

**.** **=** 0x30008000**;**

**.** **=** ALIGN**(**4**);**

**.**text **:**

**{**

start**.**o **(.**text**)**

**\*(.**text**)**

**}**

**.** **=** ALIGN**(**4**);**

**.**data **:**

**{**

**\*(.**data**)**

**}**

**.** **=** ALIGN**(**4**);**

bss\_start **=** **.;**

**.**bss **:**

**{**

**\*(.**bss**)**

**}**

bss\_end **=** **.;**

**}**

### Makefile

all**:** start.o

arm-linux-ld -Tgboot.lds -o gboot.elf $^

arm-linux-objcopy -O binary gboot.elf gboot.bin

%.o **:** %.S

arm-linux-gcc -g -c $^

%.o **:** %.c

arm-linux-gcc -g -c $^

.PHONY**:** clean

clean**:**

rm \*.o \*.elf \*.bin

### start.S

**.text**

.global \_start

\_start**:**

b reset

ldr pc**,** \_undifined\_instruction

ldr pc**,** \_software\_interrupt

ldr pc**,** \_prefetch\_abort

ldr pc**,** \_data\_abort

ldr pc**,** \_not\_used

ldr pc**,** \_irq

ldr pc**,** \_fiq

\_undifined\_instruction**:** .word undifined\_instruction

\_software\_interrupt**:** .word software\_interrupt

\_prefetch\_abort**:** .word prefetch\_abort

\_data\_abort**:** .word data\_abort

\_not\_used**:** .word not\_used

\_irq**:** .word irq

\_fiq**:** .word reset

undifined\_instruction**:**

**nop**

software\_interrupt**:**

**nop**

prefetch\_abort**:**

**nop**

data\_abort**:**

**nop**

not\_used**:**

**nop**

irq**:**

**nop**

fiq**:**

**nop**

reset**:**

**bl** set\_svc

**bl** disable\_watchdog

**bl** disable\_interrupt

**bl** disable\_mmu

**bl** light\_led

set\_svc**:**

mrs r0**,** cpsr

bic r0**,** r0**,**#0x1f

orr r0**,** r0**,**#0xd3

msr cpsr**,** r0

**mov** pc**,** lr

#define pWTCON 0x53000000

disable\_watchdog**:**

ldr r0**,** **=**pWTCON

**mov** r1**,** #0x0

**str** r1**,** **[**r0**]**

**mov** pc**,** lr

disable\_interrupt**:**

mvn r1**,** #0x0

ldr r0**,** **=**0x4a000008

**str** r1**,** **[**r0**]**

**mov** pc**,** lr

disable\_mmu**:**

mcr p15**,**0**,**r0**,**c7**,**c7**,**0

mrc p15**,**0**,**r0**,**c1**,**c0**,**0

bic r0**,** r0**,** #0x00000007

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

**mov** pc**,** lr

#define GPBCON 0x56000010

#define GPBDAT 0x56000014

light\_led**:**

ldr r0**,** **=**GPBCON

**mov** r1**,** #0x400

**str** r1**,** **[**r0**]**

ldr r0**,** **=**GPBDAT

**mov** r1**,** #0x0

**str** r1**,** **[**r0**]**

**mov** pc**,** lr

# 专题07-ARM跑快了---时钟初始化

[主目录](#_目录)

## 硬件核心理论

I2C（Inter－Integrated Circuit）总线是用于连接微[控制器](http://baike.so.com/doc/422704.html" \t "_blank)及其外围[设备](http://baike.so.com/doc/5342970.html" \t "_blank)。

PCI是Peripheral Component Interconnect([外设部件互连标准](http://baike.so.com/doc/782379-827795.html" \t "_blank))的缩写，它是目前个人电脑中使用最为广泛的接口，几乎所有的主板产品上都带有这种插槽

关于时钟的理解：

通俗易懂的例子，好吧，分析一下抬轿子，八抬大轿，轿夫要一起用力，用力的方向和时间需一致，移轿中每人的行进方向和速率也要一致，轿夫头负责喊号子，他的行动和速度是其它七个人的参照基点，这个轿夫头在某种意义上就是时钟，没有这个轿夫头，八个人各行其事，轿子是无法到达目的地的，甚至都不能被抬离地面。

搞清楚CPU内部主要是由什么器件组成的就知道为什么了

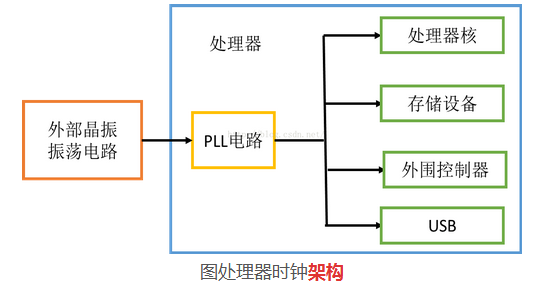
如果CPU内部没有一个时基（标准事件序列），CPU要么死亡，要么乱跳一气（部分封闭环自成“时钟”体系）。  
  
反观我们自己，我们的心脏，我们的生物钟，都是我们自己的时钟。紊乱了，后果可想而知。

那没时钟你怎么让它听话的工作呢？定时，I2C等等，怎么实现呢？

小时候玩的发条青蛙，记得不。那发条实际上是带动的内部一个齿轮。可以理解为齿轮上的每一个齿都是一个脉冲源。每一个齿的滑动，青蛙内部的其它机械结构就动一下，齿轮的连续转动，也就带动机械的连续性了。

齿轮有精密型和疏散型，精密型代表高频率，疏散型代表较低频率，只有这种有机组合才能使发条青蛙正常运动起来。

其中的晶振就代表发条，发条的动能带动各个时钟体系的工作。于是带动起青蛙运动。  
则我认为，CPU也是这样，一个时钟周期（齿轮上的一个齿）驱动内部的锁存器及状态机的值或状态的变化（因为是时序电路而不是组合电路），连续的时钟信号则使锁存器或状态机的值或状态产生连续的变化（当然是有条件的变化，不然就乱跑了）。



指令周期是执行一条指令所需要的时间，一般由若干个[机器周期](http://baike.so.com/doc/6004412-6217394.html" \t "_blank)组成，是从取指令、分析指令到执行完所需的全部时间。

计算机所以能自动地工作，是因为CPU能从存放程序的内存里取出一条指令并执行这条指令;紧接着又是取指令，执行指令，如此周而复始，构成了一个封闭的循环。除非遇到停机指令，否则这个循环将一直继续下去。

指令周期 :CPU从内存取出一条指令并执行这条指令的时间总和。

CPU周期 :又称机器周期，CPU访问一次内存所花的时间较长，因此用从内存读取一条指令字的最短时间来定义。 所以CPU周期或机器周期就是指读取一条指令所花的时间。

时钟周期: 通常称为节拍脉冲或T周期。一个CPU周期包含若干个[时钟周期](http://baike.so.com/doc/6190137-6403389.html" \t "_blank)。

比如说，在指令系统都相同的情况下，同样是12MHz的晶振，51单片机内部系统时钟为1MHz，而AVR单片机的内部系统时钟为10MHz。在执行指令的时候，AVR的速度是51的10倍。实际上，AVR大部分指令为单周期指令，而51有许多指令是多周期指令。

我对时钟频率的理解

CPU内部对指令的执行是严格按照时钟频率来执行的，未初始化的时钟频率就是晶振频率，非常低，但是能执行指令，只不过速度慢而已。换句话说，如果没有脉冲信号，CPU就无法执行指令。你也可以这样理解，CPU就像我们的大脑一样，而时钟频率就像我们的心脏一样。大脑想要正常工作心脏就得日夜不停地跳动。而你可以把其中的心脏跳动想象成脉冲信号，只要你开心就好。

时钟脉冲信号在嵌入式产品中到底起到什么作用？

**举个例子，大部分同学都读过大学，我们每天不同时刻都会对自己有一定的时间规划，比如早点9点到11点钟上课，到了11点以后吃饭，下午3点下午的课就开始了。那么问题来了，那么比如9点钟我是要去上课的，那么谁来告诉我9点钟到了呢？反正就是要有这个时间度量，用来提醒我这个时刻该做什么事。那么同样在我们处理器里面也是要有时间的概念。所以我们必须要有时间的提醒机制在这个处理器里面。那么什么用来做时间提醒机制呢？就是时钟脉冲信号。**

什么是时钟脉冲信号？

时钟脉冲信号:按一定电压幅度，一定时间间隔连续发出的脉冲信号。

什么是时钟脉冲频率？

单位时间内产生时钟脉冲个数就是时钟脉冲频率。通俗来讲就是主频，用来表示CPU的运算速度的，时钟频率和CPU运算速度成正比关系。

那么谁来提供这个时钟脉冲信号呢？

时钟源:第一种是晶振，第二种是锁相环(PLL)

什么是晶振？

晶振全称是晶体振荡器，是一种电子元件。好处是结构简单，噪声很低；缺点是当想得到高频的时候成本比较大，生产周期长。

什么是锁相环(PLL)？

PLL合成器是一种更为复杂的系统时钟源。通用PLL合成器需要一个外部晶体并包含一个能够对晶体的特定频率加倍或分频的集成锁相环(PLL)电路。有点高频产生成本低，可以配置不同频率。

不同处理器(不同核)时钟体系？

学习一款处理器的时钟体系方法，如何掌握这些固定信息，这些固定信息是:**首先我们要去了解它的晶振频率，其次我们要去了解这个时钟体系有多少PLL，这些PLL分别哪些时钟，产生的这些时钟分别用来做什么的？**

2440时钟体系？

晶振:12MHz

PLL？MPLL、UPLL

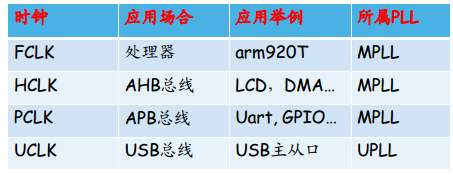
CLK？MPLL(FCLK, HCLK, PCLK), UPLL(UCLK)

FCLK:处理器:ARM920T

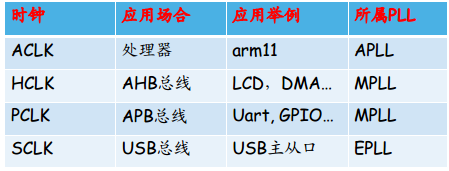
HCLK:AHB总线:LCD, DMA,...

PCLK:APB总线:uart,GPIO,...

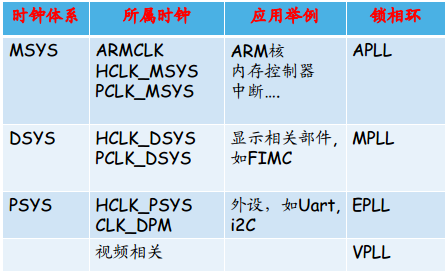
UCLK:USB总线:USB主从口



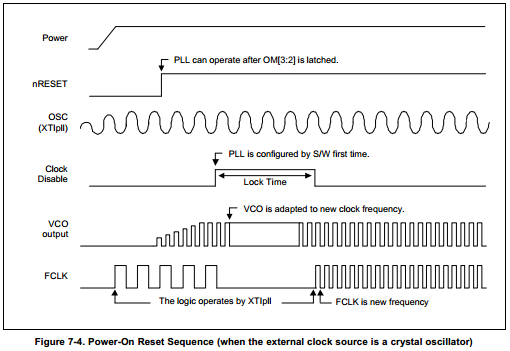
6410时钟体系？



210时钟体系？



时钟初始化流程？：芯片手册:时钟&电源管理章节



什么是AHB总线？APB总线？

AHB(Advanced High performance Bus)[系统总线](http://baike.so.com/doc/5682001.html" \t "_blank)和APB(Advanced Peripheral Bus)外围总线。

芯片手册导读

**参考:芯片手册: 7. Clock&Power Management: Figure 7-4. Power-On Reset Sequence (when the external clock source is a crystal oscillator)**

上面这个图反映的是CPU上电之后从低频率到高频率的这样一个演变的过程。

首先FCLK频率比较低的，一般就是晶振频率。PLL is configured by S/W first time.

lock time,之后就变为高频。

所以综合上述，我们总结时钟初始化流程:

\*\*配置lock time

\*\*配置FCLK

\*\*配置HCLK

\*\*配置PCLK

\*\*配置UCLK

总结上面过程得知:

\*\*配置lock time

\*\*配置分频系数（分频系数可以参考uboot代码）

\*\*配置FCLK,之后的HCLK,PCLK由分频系数自动搞定

**芯片手册:7. Clock&Power Management :NOTES:**

1. CLKDIVN should be set carefully not to exceed the limit of HCLK and PCLK.  
2. If HDIVN is not 0, the CPU bus mode has to be changed from the fast bus mode to the asynchronous  
bus mode using following instructions(S3C2440 does not support synchronous bus mode).  
**MMU\_SetAsyncBusMode**  
**mrc p15,0,r0,c1,c0,0**  
**orr r0,r0,#R1\_nF:OR:R1\_iA**  
**mcr p15,0,r0,c1,c0,0**

If HDIVN is not 0 and the CPU bus mode is the fast bus mode, the CPU will operate by the HCLK.  
This feature can be used to change the CPU frequency as a half or more without affecting the HCLK  
and PCLK.

根据上面的NOTE,我们进一步优化上面时钟初始化流程:

\*\*配置lock time

\*\*配置分频系数

\*\*设置处理器为异步工作模式

\*\*配置FCLK,之后的HCLK,PCLK由分频系数自动搞定

芯片手册导读

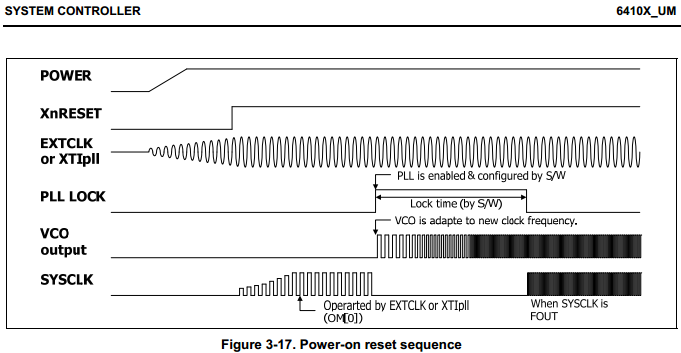
\*\*配置lock time:芯片手册关键字lock time:LOCKTIME:0x4c000000:我们使用默认值

\*\*配置分频系数:下翻页:可根据uboot来:CLKDIVN:0x4c000014:HDIVN:PDIVN

\*\*设置处理器为异步工作模式

\*\*配置FCLK,之后的HCLK,PCLK由分频系数自动搞定

6410时钟体系芯片手册导读：系统控制器章节



## 时钟初始化实验源码mini2440

### gboot.lds

OUTPUT\_ARCH**(**arm**)**

ENTRY**(**\_start**)**

SECTIONS **{**

**.** **=** 0x30008000**;**

**.** **=** ALIGN**(**4**);**

**.**text **:**

**{**

start**.**o **(.**text**)**

**\*(.**text**)**

**}**

**.** **=** ALIGN**(**4**);**

**.**data **:**

**{**

**\*(.**data**)**

**}**

**.** **=** ALIGN**(**4**);**

bss\_start **=** **.;**

**.**bss **:**

**{**

**\*(.**bss**)**

**}**

bss\_end **=** **.;**

**}**

### Makefile

all**:** start.o

arm-linux-ld -Tgboot.lds -o gboot.elf $^

arm-linux-objcopy -O binary gboot.elf gboot.bin

%.o **:** %.S

arm-linux-gcc -g -c $^

%.o **:** %.c

arm-linux-gcc -g -c $^

.PHONY**:** clean

clean**:**

rm \*.o \*.elf \*.bin

### start.S

**.text**

.global \_start

\_start**:**

b reset

ldr pc**,** \_undifined\_instruction

ldr pc**,** \_software\_interrupt

ldr pc**,** \_prefetch\_abort

ldr pc**,** \_data\_abort

ldr pc**,** \_not\_used

ldr pc**,** \_irq

ldr pc**,** \_fiq

\_undifined\_instruction**:** .word undifined\_instruction

\_software\_interrupt**:** .word software\_interrupt

\_prefetch\_abort**:** .word prefetch\_abort

\_data\_abort**:** .word data\_abort

\_not\_used**:** .word not\_used

\_irq**:** .word irq

\_fiq**:** .word reset

undifined\_instruction**:**

**nop**

software\_interrupt**:**

**nop**

prefetch\_abort**:**

**nop**

data\_abort**:**

**nop**

not\_used**:**

**nop**

irq**:**

**nop**

fiq**:**

**nop**

reset**:**

**bl** set\_svc

**bl** disable\_watchdog

**bl** disable\_interrupt

**bl** disable\_mmu

**bl** init\_clock

**bl** light\_led

set\_svc**:**

mrs r0**,** cpsr

bic r0**,** r0**,**#0x1f

orr r0**,** r0**,**#0xd3

msr cpsr**,** r0

**mov** pc**,** lr

#define pWTCON 0x53000000

disable\_watchdog**:**

ldr r0**,** **=**pWTCON

**mov** r1**,** #0x0

**str** r1**,** **[**r0**]**

**mov** pc**,** lr

disable\_interrupt**:**

mvn r1**,** #0x0

ldr r0**,** **=**0x4a000008

**str** r1**,** **[**r0**]**

**mov** pc**,** lr

disable\_mmu**:**

mcr p15**,**0**,**r0**,**c7**,**c7**,**0

mrc p15**,**0**,**r0**,**c1**,**c0**,**0

bic r0**,** r0**,** #0x00000007

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

**mov** pc**,** lr

#define CLKDIVN 0x4c000014

#define MPLLCON 0x4c000004

#define MPLL\_405MHZ **((**127**<<**12**)|(**2**<<**4**)|(**1**<<**0**))**

init\_clock**:**

ldr r0**,** **=**CLKDIVN

**mov** r1**,** #0x5

**str** r1**,** **[**r0**]**

mrc p15**,**0**,**r0**,**c1**,**c0**,**0

orr r0**,**r0**,**#0xc0000000

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

ldr r0**,** **=**MPLLCON

ldr r1**,** **=**MPLL\_405MHZ

**str** r1**,** **[**r0**]**

**mov** pc**,** lr

#define GPBCON 0x56000010

#define GPBDAT 0x56000014

light\_led**:**

ldr r0**,** **=**GPBCON

**mov** r1**,** #0x400

**str** r1**,** **[**r0**]**

ldr r0**,** **=**GPBDAT

**mov** r1**,** #0x0

**str** r1**,** **[**r0**]**

**mov** pc**,** lr

# 专题08-不用内存怎么行

[主目录](#_目录)

**NOTES：注意本课程2440、6410、210都要听，因为比较基础的东西放前面讲，比较深层次的知识放后面讲。**

## 硬件核心理论

内存学习清单

内存分类:SRAM, DRAM(SDRAM,DDR,DDR2,...)

内存内部结构:表结构,L-Bank,寻址信息(L-Bank选择信号, 行地址, 列地址)

什么是SRAM？

它是一种静态存取功能的内存，不需要定期刷新电路就能保持它内部存储的数据。其优点是存取速度快;但缺点是:功耗大,成本高。用在特殊场合,比如CPU内部的stepping stone.

什么是DRAM？

它基本的小元件是小电容，电容可以在两极上保留电荷，但是需要定期充电，定期刷新，否则数据会丢失。缺点:由于要定期刷新存储介质，存取速度较慢。

什么是SDRAM？

SDRAM(Synchronous Dynamic Random Access Memory):同步动态随机存储器。

同步:该内存工作需要有同步时钟,内部命令的发送和数据的接收都以该时钟为基准。

什么是表结构？

内存的内部如同表格,数据就存在每个单元格中。数据读写时，先指定行号(行地址RA),再指定列号(列地址CA),我们就能准确找到所需要的单元格。而这张表格成为Logical Bank(L-Bank)。由于技术、成本等原因，一块内存不可能把所有单元格都做到一个L-Bank中，现在的内存基本都会分割成4个L-Bank。

推导内存容量公式？

一块内存由4个L-Bank, L-Bank==单元格数目, 每个单元格容量。

例如: 4Banks x 4MB x 16Bit == 内存容量

**地址空间(地址线,片选信号,外设空间,存储控制器)**

地址线:通过2440芯片原理图得知它提供了27根地址线,也就是2的27次方也就是128MB，但是这个太小了。

地址总线宽度:它决定了CPU可以访问的物理地址空间。比如32bit可以访问2^32地址。

2440处理器对外提供了多少根地址线？

答:参考:原理图:S3C2440:ADDR:27根地址线，意味着我们S3C2440芯片对外提供了27根地址线，这意味着它能够访问2的27次方的外设空间。但128MB太小了，怎么扩大呢？

于是我们提出了片选信号的概念。

为什么我们2440内存起始地址是0x30000000？

因为我们内存安排在L-Bank6上，而L-Bank6的地址就是0x30000000。

处理器---地址总线---片选信号---外设

什么是片选信号？

提供28MB地址线的处理器，正好A芯片刚好128MB刚好能访问A芯片，B芯片也有128MB刚好也能访问B芯片，所以片选信号就是帮我们选中某个芯片，其他芯片就无法访问。2440提供了8个片选信号(nGCS0~nGCS7)。只能选择其中的128MB访问，不能同时访问这些片选信号。所以2440访问一下放大了:128MBx7 = 1GB。我们2440内存地址一般放在片选6和片选7上的。前面我们一直疑问的为什么内存地址在0x30000000呢？答案就在这里，因为开发板厂商把内存放在片选6上面了。

外设空间

存储控制器

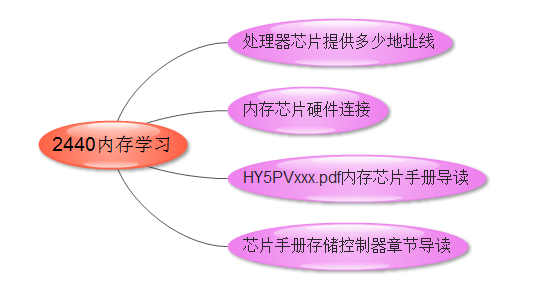
CPU通过存储控制器—访问(内存,norflash,网卡芯片等等),所以说我们要去控制内存，我们要去初始化内存，我们就是去控制处理器内部的存储控制器。

**2440内存芯片硬件连接**

参考手册:HY57V5....pdf

内存芯片硬件连接

内存芯片是如何和2440芯片连接的？



存储控制器寄存器

我们初始化内存就是初始化这个处理器内部的存储控制器来实现的。它主要是控制我们外设的访问。

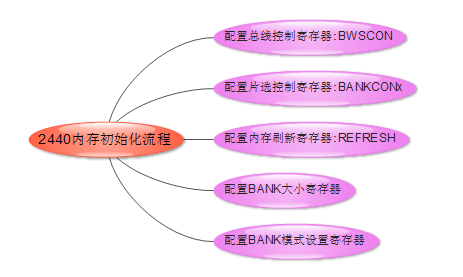
2440存储控制器芯片手册导读

芯片手册: 5. Memory Controller:BWSCON寄存器: Bus width & wait status control register

被分为8个组，即8个L-Bank.

这里我们配置L-Bank6~7即可，因为这两L-Banks是内存安排的地址。

根据芯片手册寄存器BWSCON:[31:0]:0010 0010 0000 ...



2440内存初始化分别要配置如下寄存器:

\*\*配置总线控制寄存器

\*\*配置片选控制寄存器

\*\*配置内存刷新寄存器

\*\*配置bank大小寄存器

\*\*配置bank模式设置寄存器

**BWSCON:0x48000000:[31:0]:0010 0010 0000 ...**

**BANKCON6:0x4800001c:[16:0]:11;[3:2]:0001:其中列的位数在SDRAM手册中:关键字column**

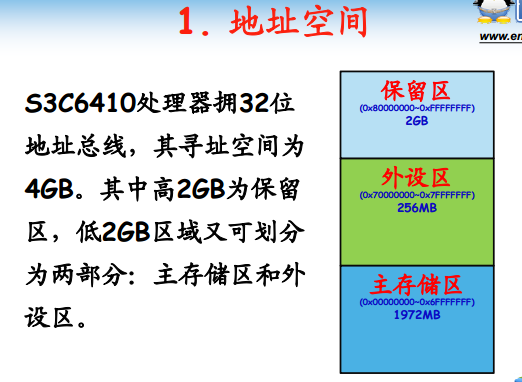
**BANKCON7:0x48000020:[16:0]:11;[3:2]:0001:其中列的位数在SDRAM手册中:关键字column**

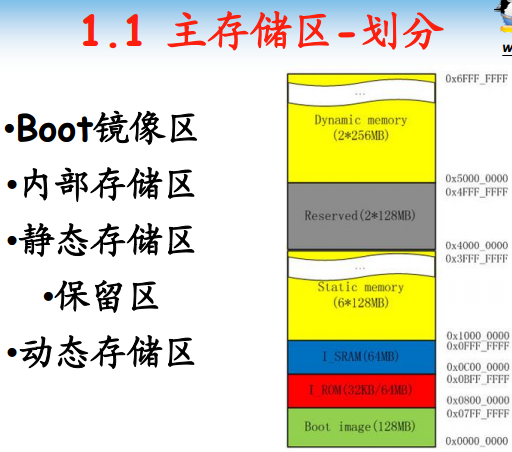
**REFRESH:0x48000024:[23:0]:1100 1100 0000 0000 0000 0000|(1269<<0)**

**BANKSIZE:0x48000028:[7:0]:1011 0001:[2:0]:我们内存芯片采用2片32MB**

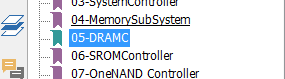
**MRSRB6:0x4800002C:[11:0]:0010 0011 0000**

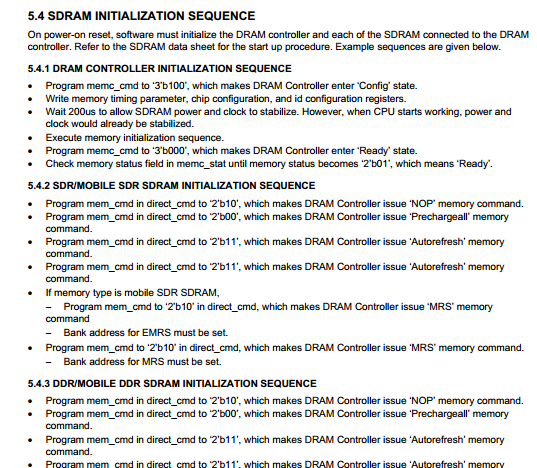
**MRSRB7:0x48000030:[11:0]:0010 0011 0000**





6410内存初始化芯片手册导读：





6410地址空间分布

S3C6410处理器拥有32位地址总线，所以其寻址空间为4GB。其中高2GB为保留区，低2GB又分为两区域:从低到高分别为主存储区和外设区。其中主存储区又分为5个区域，从低到高分别为boot镜像区,内部存储区,静态存储区,保留区,动态存储区。其中boot镜像区它不会关联某个硬件，但是当你选中某个硬件作为启动设备的时候，它会把硬件映射到这个区域来，boot镜像区主要是起到映射的作用。静态存储区，保留区为没有使用的区域，动态存储区对我们来说相当重要。

为什么6410内存起始地址是0x50000000？

因为我们把内存安排在动态存储区，而动态存储区的起始地址是在0x50000000。

**内存芯片硬件如何连接的？**

通过内存硬件模块原理图的地址线引脚数目和数据线引脚数目得出，和前面的2440内存连接相似，也是通过两块16bit地址宽度内存芯片并联成一个32bit地址宽度的内存地址宽度，从而充分使用了处理器的32bit地址总线宽度。

芯片手册导读

**芯片手册: 05-DRAMC: 5.4 SDRAM INITIALIZATION SEQUENCE:**

**:5.4.3 DDR/MOBILE DDR SDRAM INITIALIZATION SEQUENCE**

**:5.4.1 DRAM CONTROLLER INITIALIZATION SEQUENCE**

6410内存初始化是有一定步骤的，其中步骤中的操作数会与寄存器标明的操作数不一致，那么我们一般都以寄存器操作数为准。

210地址空间

210内存芯片硬件连接方式

这一部分很重要，因为后面的寄存器配置就需要这部分的理论知识。

210内存地址空间，首先210内存采用DDR2，我们210整个地址空间大小是4GB(32bit),我们的内存芯片是安排在哪个区域的？2440使用的是SDRAM，6410使用DDR，210使用DDR2，DDR2使用两个区域，下面区域DRAM0(DMC0)为512MB，上面区域DRAM1(DMC1)为1GB大小。所以210内存在地址空间的起始地址为0x20000000。6410的0地址是一个映射区，210的0地址处也是一个映射区，不连接任何硬件。2440和6410都是采用16bit数据宽度的芯片，为了把它构成32bit的数据宽度，我们在2440和6410都是采用两片芯片并联而成。而我们210的内存芯片是采用8bit数据宽度，那么把它构成32bit数据宽度，210就采用4片来并联。

210芯片手册导读

芯片手册:section 05\_memory:1 DRAM CONTROLLER:Initialization sequence for DDR2 memory type:我们这里就只配置DRAM0(512MB):DMC0

## 内存初始化实验源码mini2440

### start.S

**.text**

.global \_start

\_start**:**

b reset

ldr pc**,** \_undifined\_instruction

ldr pc**,** \_software\_interrupt

ldr pc**,** \_prefetch\_abort

ldr pc**,** \_data\_abort

ldr pc**,** \_not\_used

ldr pc**,** \_irq

ldr pc**,** \_fiq

\_undifined\_instruction**:** .word undifined\_instruction

\_software\_interrupt**:** .word software\_interrupt

\_prefetch\_abort**:** .word prefetch\_abort

\_data\_abort**:** .word data\_abort

\_not\_used**:** .word not\_used

\_irq**:** .word irq

\_fiq**:** .word reset

undifined\_instruction**:**

**nop**

software\_interrupt**:**

**nop**

prefetch\_abort**:**

**nop**

data\_abort**:**

**nop**

not\_used**:**

**nop**

irq**:**

**nop**

fiq**:**

**nop**

reset**:**

**bl** set\_svc

**bl** disable\_watchdog

**bl** disable\_interrupt

**bl** disable\_mmu

**bl** init\_clock

**bl** init\_sdram

**bl** light\_led

set\_svc**:**

mrs r0**,** cpsr

bic r0**,** r0**,**#0x1f

orr r0**,** r0**,**#0xd3

msr cpsr**,** r0

**mov** pc**,** lr

#define pWTCON 0x53000000

disable\_watchdog**:**

ldr r0**,** **=**pWTCON

**mov** r1**,** #0x0

**str** r1**,** **[**r0**]**

**mov** pc**,** lr

disable\_interrupt**:**

mvn r1**,** #0x0

ldr r0**,** **=**0x4a000008

**str** r1**,** **[**r0**]**

**mov** pc**,** lr

disable\_mmu**:**

mcr p15**,**0**,**r0**,**c7**,**c7**,**0

mrc p15**,**0**,**r0**,**c1**,**c0**,**0

bic r0**,** r0**,** #0x00000007

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

**mov** pc**,** lr

#define CLKDIVN 0x4c000014

#define MPLLCON 0x4c000008

#define MPLL\_405MHZ **((**127**<<**12**)|(**2**<<**4**)|(**1**<<**0**))**

init\_clock**:**

ldr r0**,** **=**CLKDIVN

**mov** r1**,** #0x5

**str** r1**,** **[**r0**]**

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

orr r0**,**r0**,**#0xc0000000

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

ldr r0**,** **=**MPLLCON

ldr r1**,** **=**MPLL\_405MHZ

**str** r1**,** **[**r0**]**

**mov** pc**,** lr

#define mem\_contrl 0x48000000

init\_sdram**:**

ldr r0**,** **=**mem\_contrl

**add** r3**,** r0**,** #4**\***13

adrl r1**,** mem\_data

0**:**

ldr r2**,** **[**r1**],** #4

**str** r2**,** **[**r0**],** #4

**cmp** r0**,** r3

bne 0b

**mov** pc**,** lr

mem\_data**:**

.long 0x22000000

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00018001

.long 0x00018001

.long 0x008c04f5

.long 0x000000b1

.long 0x00000030

.long 0x00000030

#define GPBCON 0x56000010

#define GPBDAT 0x56000014

light\_led**:**

ldr r0**,** **=**GPBCON

**mov** r1**,** #0x400

**str** r1**,** **[**r0**]**

ldr r0**,** **=**GPBDAT

**mov** r1**,** #0x0

**str** r1**,** **[**r0**]**

**mov** pc**,** lr

### gboot.lds

OUTPUT\_ARCH**(**arm**)**

ENTRY**(**\_start**)**

SECTIONS **{**

**.** **=** 0x30008000**;**

**.** **=** ALIGN**(**4**);**

**.**text **:**

**{**

start**.**o **(.**text**)**

**\*(.**text**)**

**}**

**.** **=** ALIGN**(**4**);**

**.**data **:**

**{**

**\*(.**data**)**

**}**

**.** **=** ALIGN**(**4**);**

bss\_start **=** **.;**

**.**bss **:**

**{**

**\*(.**bss**)**

**}**

bss\_end **=** **.;**

**}**

### Makefile

all**:** start.o

arm-linux-ld -Tgboot.lds -o gboot.elf $^

arm-linux-objcopy -O binary gboot.elf gboot.bin

%.o **:** %.S

arm-linux-gcc -g -c $^

%.o **:** %.c

arm-linux-gcc -g -c $^

.PHONY**:** clean

clean**:**

rm \*.o \*.elf \*.bin

# 专题09--代码搬移不可少

[主目录](#_目录)

gboot-代码搬移不可少-a++和++a操作区别

ptr[a++]测试成功, ptr[++a]测试成功

**ldr r0, [r1], #0x1 ;r0=r1, r1=r1+1**

**ldr r0, [r1, #0x1] ;r0=r1+1**

**ldr r0, [r1, #0x1]! ;r0=r1+1, r1=r1+1**

**str r1, [r0] ;[r0]=r1**

**常见的用法：**

**ldr r0, =WTCON @把WTCON地址装载进r0寄存器**

**...**

**str r1, [r0] @把r1数据写入r0指定地址(WTCON)**

**关于代码搬移的事儿，知道起点和终点就OK了。**

**关于起点和终点的事儿，知道芯片手册就OK了**

**关于芯片手册的是，知道内存管理或系统控制章节就OK了。**

## 硬件核心理论

2440-ARM启动流程回顾

我们这里主要选择从nand启动，nand启动呢就必须要依赖于一个部件，即片内SRAM又名垫脚石。nand不能统一编址。

比如我们烧进去的uboot.bin为100KB，当我们处理一旦上电之后，我们的2440会自动从nand拷贝出最前端4KB内容到垫脚石中，并从0地址也就是垫脚石起始地址处开始执行。而这4KB内容的任务之一就是把nand中这**整个100KB**内容拷贝到内存中，然后跳转到内存中执行**剩下的代码(SDRAM\_BASE+4KB)**。

6410启动流程情况

6410一旦上电后，首先运行SROM里面的BL0，而BL0会从nand中拷贝8KB内容到垫脚石中运行，我们利用这8KB代码把nand中100KB拷贝到SDRAM中。

210启动流程情况

210一旦上电后，首先运行SROM里面的BL0，而BL0会从nand中拷贝96KB内容到垫脚石中运行，如果还不够，这96KB同样把nand中100KB全部拷贝到SDRAM中执行。

代码搬移？

搬移的起点和终点？搬移的方式？

起点是nand, 终点是SDRAM/SRAM。我们目前编写的gboot.bin远远小于4KB，但我们仍然需要做搬移工作，因为这是为了让大家理解搬移原理。

搬移方式1：通过垫脚石从nand中搬移。

搬移方式2：直接通过nand传入内存。

起点:SRAM垫脚石:它的起始地址:2440:0x0

起点:SRAM垫脚石:它的起始地址:6410:芯片手册:memory map章节:stepping:0x0c000000

起点:SRAM垫脚石:它的起始地址:210:芯片手册:memory map:iRAM:0xd0020000

终点？

链接器脚本代码段链接起始地址？

终点就是内存起始地址。链接地址作用就是当PC指向reset时，PC会被赋值为这个链接起始地址，其次可以直接通过这个链接地址直接修改PC指针，比如跳转到SDRAM\_BASE，那么PC就被赋值: ldr pc, =SDRAM\_BASE 其中SDRAM\_BASE==0x30008000。执行后就直接跳转到内存执行bootloader剩下的代码了。

相对跳转：bl func (相对跳转就是通过PC当前值和标签处值的差值来跳转)

绝对跳转：ldr pc, xxx (绝对跳转就是直接修改PC指针)

## 代码搬移实验源码mini2440

本实验有出入，因为我们nandflash都还没初始化，所以我们的BootLoader实质上是在IROM中的，所以我们的代码搬移起始地址就再IROM或垫脚石中，终点就是在SDRAM中，而实际中的代码搬移的起点是在nandflash中，所以我们目前的代码搬移就是把垫脚石模拟成nandflash了。

### start.S

**.text**

.global \_start

\_start**:**

b reset

ldr pc**,** \_undifined\_instruction

ldr pc**,** \_software\_interrupt

ldr pc**,** \_prefetch\_abort

ldr pc**,** \_data\_abort

ldr pc**,** \_not\_used

ldr pc**,** \_irq

ldr pc**,** \_fiq

\_undifined\_instruction**:** .word undifined\_instruction

\_software\_interrupt**:** .word software\_interrupt

\_prefetch\_abort**:** .word prefetch\_abort

\_data\_abort**:** .word data\_abort

\_not\_used**:** .word not\_used

\_irq**:** .word irq

\_fiq**:** .word reset

undifined\_instruction**:**

**nop**

software\_interrupt**:**

**nop**

prefetch\_abort**:**

**nop**

data\_abort**:**

**nop**

not\_used**:**

**nop**

irq**:**

**nop**

fiq**:**

**nop**

reset**:**

**bl** set\_svc

**bl** disable\_watchdog

**bl** disable\_interrupt

**bl** disable\_mmu

**bl** init\_clock

**bl** init\_sdram

**bl** copy\_to\_ram

**bl** light\_led

set\_svc**:**

mrs r0**,** cpsr

bic r0**,** r0**,**#0x1f

orr r0**,** r0**,**#0xd3

msr cpsr**,** r0

**mov** pc**,** lr

#define pWTCON 0x53000000

disable\_watchdog**:**

ldr r0**,** **=**pWTCON

**mov** r1**,** #0x0

**str** r1**,** **[**r0**]**

**mov** pc**,** lr

disable\_interrupt**:**

mvn r1**,** #0x0

ldr r0**,** **=**0x4a000008

**str** r1**,** **[**r0**]**

**mov** pc**,** lr

disable\_mmu**:**

mcr p15**,**0**,**r0**,**c7**,**c7**,**0

mrc p15**,**0**,**r0**,**c1**,**c0**,**0

bic r0**,** r0**,** #0x00000007

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

**mov** pc**,** lr

#define CLKDIVN 0x4c000014

#define MPLLCON 0x4c000008

#define MPLL\_405MHZ **((**127**<<**12**)|(**2**<<**4**)|(**1**<<**0**))**

init\_clock**:**

ldr r0**,** **=**CLKDIVN

**mov** r1**,** #0x5

**str** r1**,** **[**r0**]**

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

orr r0**,**r0**,**#0xc0000000

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

ldr r0**,** **=**MPLLCON

ldr r1**,** **=**MPLL\_405MHZ

**str** r1**,** **[**r0**]**

**mov** pc**,** lr

#define mem\_contrl 0x48000000

init\_sdram**:**

ldr r0**,** **=**mem\_contrl

**add** r3**,** r0**,** #4**\***13

adrl r1**,** mem\_data

0**:**

ldr r2**,** **[**r1**],** #4

**str** r2**,** **[**r0**],** #4

**cmp** r0**,** r3

bne 0b

**mov** pc**,** lr

copy\_to\_ram**:**

ldr r0**,** **=**0x0

ldr r1**,** **=**0x30008000

**add** r3**,** r0**,** #1024**\***4

copy\_loop**:**

ldr r2**,** **[**r0**],** #4

**str** r2**,** **[**r1**],** #4

**cmp** r0**,** r3

bne copy\_loop

**mov** pc**,** lr

mem\_data**:**

.long 0x22000000

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00018001

.long 0x00018001

.long 0x008c04f5

.long 0x000000b1

.long 0x00000030

.long 0x00000030

#define GPBCON 0x56000010

#define GPBDAT 0x56000014

light\_led**:**

ldr r0**,** **=**GPBCON

**mov** r1**,** #0x400

**str** r1**,** **[**r0**]**

ldr r0**,** **=**GPBDAT

**mov** r1**,** #0x0

**str** r1**,** **[**r0**]**

**mov** pc**,** lr

### 参考链接器脚本：

OUTPUT\_ARCH**(**arm**)**

ENTRY**(**\_start**)**

SECTIONS **{**

**.** **=** 0x30008000**;**

**.** **=** ALIGN**(**4**);**

**.**text **:**

**{**

start**.**o **(.**text**)**

**\*(.**text**)**

**}**

**.** **=** ALIGN**(**4**);**

**.**data **:**

**{**

**\*(.**data**)**

**}**

**.** **=** ALIGN**(**4**);**

bss\_start **=** **.;**

**.**bss **:**

**{**

**\*(.**bss**)**

**}**

bss\_end **=** **.;**

**}**

### 相应Makefile：

all**:**start.o

arm-linux-ld -Tgboot.lds -o gboot.elf $^

arm-linux-objcopy -O binary gboot.elf gboot.bin

%.o**:**%.S

arm-linux-gcc -g -c $^

%.o**:**%.c

arm-linux-gcc -g -c $^

.PHONY**:**clean

clean**:**

rm -rf \*.o \*.elf \*.bin

# 专题10-C语言环境初始化

[主目录](#_目录)

## 堆栈核心理论

为什么需要C环境初始化？

因为BL2主要是用C语言编写的，而C语言执行需要堆栈环境，所以我们本课内容就是初始化堆栈。

概念解析:栈

这样理解就OK了:你把栈理解为一摞书本,最上面的为栈顶，最底部为栈底。我们放书都是从最上面放然后越摞越高，而我们取书的时候我们都是从书的最顶端开始取书。如果我们要取这摞书栈底那本，我们则需要从栈顶

概念解析:空/满栈

满栈:当栈指针SP总是指向最后压入堆栈的数据。

空栈:当栈指针SP总是指向下一个将要放入数据的空位置。

ARM采用满栈！

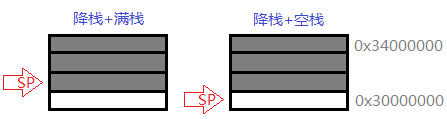
概念解析:升/降栈

升栈和降栈区分依据就是SP指针移动方向。

升栈：随着数据的入栈，SP指针从低地址到高地址移动。

降栈：随着数据的入栈，SP指针从高地址到低地址移动。

ARM采用降栈！



概念解析:栈帧

简单来讲，栈帧就是一个函数所使用的那部分栈，所有函数的栈帧串起来就组成了一个完整的栈。栈帧的两个边界分别由FP(r11)和SP(r13)来限定的。

栈作用:保存局部变量

int main(void)

{

int a;

a++;

return a;

}

# arm-linux-gcc -g test.c -o test.elf

# arm-linux-objdump -D -S test.elf >test.dump

查看反汇编文件：

**首先为main函数建立栈帧:这里我们先假设sp=100, fp=100**

int main(void)

{

**push {fp} ;str fp,[sp, #-4]! @(满栈)str fp,[sp-4]; sp=sp-4 ==96**

**add fp, sp, #0 ;0x0 @fp = sp ==96**

**sub sp, sp, #12 ;0xc @sp = sp -12 == 96-12==84**

int a;

**a++;**

**ldr r3, [fp, #-8] @**

**add r3, r3, #1**

**str r3, [fp, #-8]**

return a;

ldr r3, [fp, #-8]

}

mov r0, r3

add sp, fp, #0

pop {fp}

bx lr

栈作用:参数传递

当参数大于4个时候就采用栈来存储了，如果小于等于4个就直接用寄存器来存储。

void func1(int a, int b, int c, int d, int e, int f)

{

int k;

k = e+f;

}

int main(void)

{

func1 (1,2,3,4,5,6);

return 0;

}

# arm-linux-gcc -g test.c -o test.elf

# arm-linux-objdump -D -S test.elf >test.dump

查看反汇编文件：

栈作用:保存寄存器值

当函数出现递归调用的时候栈的作用就是保存寄存器的值。

void func2(int a, int b)

{

int k;

k = e+f;

}

void func1(int a, int b)

{

int c;

func2(3,4);

c = a+b;

return ;

}

int main(void)

{

func1 (1,2);

return 0;

}

# arm-linux-gcc -g test.c -o test.elf

# arm-linux-objdump -D -S test.elf >test.dump

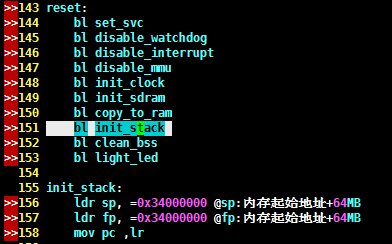
查看反汇编文件：

2440栈初始化原理

根据上面的满栈降栈原理，所以我们的栈初始化

## 栈初始化实验源码2440

范例代码：start.S



## BSS段核心理论

&global\_init\_a = 0x80498dc //结合进程对应maps:数据段

&global\_uninit\_a = 0x80498f8 //结合进程对应maps:数据段(bss)

&static\_global\_init\_a = 0x80498e0 //结合进程对应maps:数据段

&static\_global\_uninit\_a = 0x80498f0 //结合进程对应maps:数据段 (bss)

&const\_global\_a = 0x8048614 //结合进程对应maps:代码段

&local\_init\_a = 0xbfd31b98 //结合进程对应maps:栈

&local\_uninit\_a = 0xbfd31b94 //结合进程对应maps:栈

&static\_local\_init\_a = 0x80498e4 //结合进程对应maps:数据段

&static\_local\_uninit\_a = 0x80498f4 //结合进程对应maps:数据段 (bss)

&const\_local\_a = 0xbfd31b90 //结合进程对应maps:栈

malloc\_point\_a = 0x823d008 //结合进程对应maps:堆

CURRENT PROCESS: 3883 //通过getpid()函数得到该进程号

代码段：代码，全局常量(const)，字符串常量

数据段：全局变量(init/uninit)，静态变量(全局/局部，init/uninit)

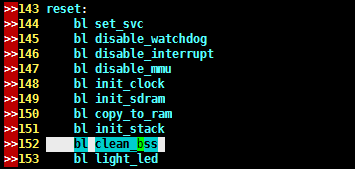
BSS段：除了栈以外的所有未初始化变量

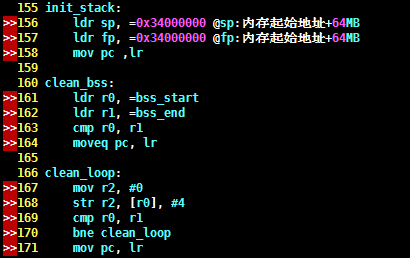
堆：动态分配的区域

栈：局部变量(init/uninit，但不包含静态变量)，局部只读变量,局部常量

## BSS段初始化实验源码2440

范例代码：start.S





## 一跃进入C大门核心理论

**该课程主要是从BL1跳转到BL2的关键步骤**

**其中210的BL0把BL1连头都复制到内存中了，为了使gboot\_main能点亮LED，我们必须跳过这个头，而这个头大小是16字节，所以我们在拷贝到内存这个函数标签处，找到搬移起点+16B的偏移来避过这个头，这样我们拷过去的内容就是纯bootloader文件。**

## 一跃进入C大门实验源码2440

main.c:

#define GPBCON (volatile unsigned long\*)0x56000010

#define GPBDAT (volatile unsigned long\*)0x56000014

int **gboot\_main**()

{

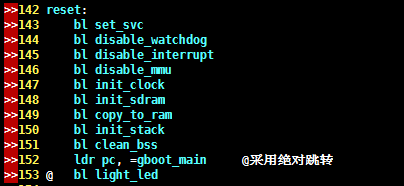
\*(GPBCON) = 0x400;

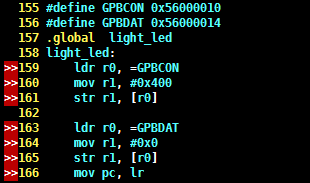
\*(GPBDAT) = 0x0;

return 0;

}

范例代码：start.S(局部核心)





# 专题11-瞻前顾后

[主目录](#_目录)

# 第三季-裸奔吧ARM(bootloader)-下学期

# 专题01-世界一下变大了-MMU

[主目录](#_目录)

## MMU核心理论

MMU作用？

将虚拟地址转化为物理地址；管理访问权限。在OS或者比较复杂点的bootloader上都会把这个MMU使用起来，就是为了避免地址使用冲突。通过MMU访问某个物理地址，这个地址的数据是只读的，如果我们写的话就会出错，这就是MMU在访问权限上的管理。

MMU针对不同进程有不同转化过程，有不同映射关系的。

深入剖析地址转化

转化流程总体分析**:**ARM核手册**:**memory managerment unit章节

**:** Figure 3**-**2 Translating page tables

mmu如何工作的？

TTB**:**translation table base**:**这个TTB里面存放这张表的基地址**:**而TTB被记录在CP15\_c2里面**:**然后我们MMU启动时会自动去cp15\_c2里取出TTB**:**然后读出TTB内容也就是这张表的基地址**:**于是根据这个基地址就找到这张表了**:**MMU就可以工作了。

所以说我们系统工程师的任务就是建立这张表并把TTB写入cp15\_c2寄存器中。

TTB配置

指定TTB基地址

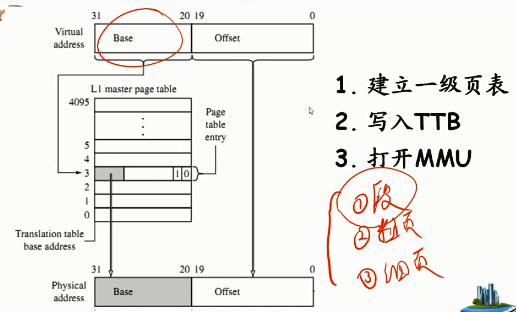
把这个基地址写入cp15\_c2寄存器

MMU初始化流程

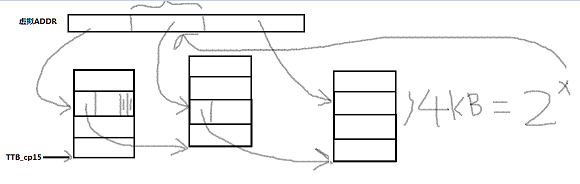
**\*\***建立一级页表

**\*\***写入TTB

**\*\***打开MMU



细页转化分析图解，如下图：



## mmu实验代码mini2440

## start.S

**.text**

.global \_start

\_start**:**

b reset

ldr pc**,** \_undifined\_instruction

ldr pc**,** \_software\_interrupt

ldr pc**,** \_prefetch\_abort

ldr pc**,** \_data\_abort

ldr pc**,** \_not\_used

ldr pc**,** \_irq

ldr pc**,** \_fiq

\_undifined\_instruction**:** .word undifined\_instruction

\_software\_interrupt**:** .word software\_interrupt

\_prefetch\_abort**:** .word prefetch\_abort

\_data\_abort**:** .word data\_abort

\_not\_used**:** .word not\_used

\_irq**:** .word irq

\_fiq**:** .word reset

undifined\_instruction**:**

**nop**

software\_interrupt**:**

**nop**

prefetch\_abort**:**

**nop**

data\_abort**:**

**nop**

not\_used**:**

**nop**

irq**:**

**nop**

fiq**:**

**nop**

reset**:**

**bl** set\_svc

**bl** disable\_watchdog

**bl** disable\_interrupt

**bl** disable\_mmu

**bl** init\_clock

**bl** init\_sdram

**bl** copy\_to\_ram

**bl** init\_stack

**bl** clean\_bss

ldr pc**,=**gboot\_main

@ **bl** light\_led

set\_svc**:**

mrs r0**,** cpsr

bic r0**,** r0**,**#0x1f

orr r0**,** r0**,**#0xd3

msr cpsr**,** r0

**mov** pc**,** lr

#define pWTCON 0x53000000

disable\_watchdog**:**

ldr r0**,=**pWTCON

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_interrupt**:**

mvn r1**,** #0x0

ldr r0**,=**0x4a000008

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_mmu**:**

mcr p15**,**0**,**r0**,**c7**,**c7**,**0

mrc p15**,**0**,**r0**,**c1**,**c0**,**0

bic r0**,** r0**,** #0x00000007

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

**mov** pc**,** lr

#define CLKDIVN 0x4c000014

#define MPLLCON 0x4c000008

#define MPLL\_405MHZ **((**127**<<**12**)|(**2**<<**4**)|(**1**<<**0**))**

init\_clock**:**

ldr r0**,=**CLKDIVN

**mov** r1**,** #0x5

**str** r1**,[**r0**]**

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

orr r0**,**r0**,**#0xc0000000

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

ldr r0**,=**MPLLCON

ldr r1**,=**MPLL\_405MHZ

**str** r1**,[**r0**]**

**mov** pc**,** lr

#define mem\_contrl 0x48000000

init\_sdram**:**

ldr r0**,=**mem\_contrl

**add** r3**,** r0**,** #4**\***13

adrl r1**,** mem\_data

0**:**

ldr r2**,[**r1**],** #4

**str** r2**,[**r0**],** #4

**cmp** r0**,** r3

bne 0b

**mov** pc**,** lr

copy\_to\_ram**:**

ldr r0**,=**0x0

ldr r1**,=**0x30008000

**add** r3**,** r0**,** #1024**\***4

copy\_loop**:**

ldr r2**,[**r0**],** #4

**str** r2**,[**r1**],** #4

**cmp** r0**,** r3

bne copy\_loop

**mov** pc**,** lr

init\_stack**:**

ldr **sp,=**0x34000000

**mov** pc **,**lr

clean\_bss**:**

ldr r0**,=**bss\_start

ldr r1**,=**bss\_end

**cmp** r0**,** r1

moveq pc**,** lr

clean\_loop**:**

**mov** r2**,** #0

**str** r2**,[**r0**],** #4

**cmp** r0**,** r1

bne clean\_loop

**mov** pc**,** lr

mem\_data**:**

.long 0x22000000

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00018001

.long 0x00018001

.long 0x008c04f5

.long 0x000000b1

.long 0x00000030

.long 0x00000030

#define GPBCON 0x56000010

#define GPBDAT 0x56000014

light\_led**:**

ldr r0**,=**GPBCON

**mov** r1**,** #0x400

**str** r1**,[**r0**]**

ldr r0**,=**GPBDAT

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

**mov** pc**,** lr

## Makefile

all**:** start.o main.o

arm-linux-ld -Tgboot.lds -o gboot.elf $^

arm-linux-objcopy -O binary gboot.elf gboot.bin

%.o**:** %.S

arm-linux-gcc -g -c $^

%.o**:** %.c

arm-linux-gcc -g -c $^

.PHONY**:** clean

clean**:**

rm \*.o \*.elf \*.bin

## gboot.lds

OUTPUT\_ARCH**(**arm**)**

ENTRY**(**\_start**)**

SECTIONS **{**

**.=**0x30008000**;**

**.=** ALIGN**(**4**);**

**.**text **:**

**{**

start.o **(.**text**)**

**\*(.**text**)**

**}**

**.=** ALIGN**(**4**);**

**.**data **:**

**{**

**\*(.**data**)**

**}**

**.=** ALIGN**(**4**);**

bss\_start **=.;**

**.**bss **:**

**{**

**\*(.**bss**)**

**}**

bss\_end **=.;**

**}**

## main.c

/\*

\* Filename: main.c

\* Platform: windows (code)

\* Platform: linux (compile)

\*/

#define C\_GPBCON (volatile unsigned long \*)0x56000010 //LED control register

#define C\_GPBDAT (volatile unsigned long \*)0x56000014 //LED data register

#define ASM\_GPBCON 0x56000010

#define ASM\_GPBDAT 0x56000014

#define V\_GPBCON (volatile unsigned long\*)0xA0000010

#define V\_GPBDAT (volatile unsigned long\*)0xA0000014

#define GPBCON (volatile unsigned long\*)0xA0000010

#define GPBDAT (volatile unsigned long\*)0xA0000014

/\*

\* 用于段描述符的一些宏定义

\*/

#define MMU\_FULL\_ACCESS (3 << 10) /\* 访问权限 \*/

#define MMU\_DOMAIN (0 << 5) /\* 属于哪个域 \*/

#define MMU\_SPECIAL (1 << 4) /\* 必须是1 \*/

#define MMU\_CACHEABLE (1 << 3) /\* cacheable \*/

#define MMU\_BUFFERABLE (1 << 2) /\* bufferable \*/

#define MMU\_SECTION (2 << 0) /\* 表示这是段描述符 \*/

#define MMU\_SECDESC (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_SECTION)

#define MMU\_SECDESC\_WB (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_CACHEABLE | MMU\_BUFFERABLE | MMU\_SECTION)

staticvoid asm\_in\_c\_light\_led **(**void**);**

staticvoid c\_light\_led**(**void**);**

staticvoid mmu\_init**(**void**);**

staticvoid create\_page\_table**(**void**);**

/\* 函数:建立页表 \*/

void create\_page\_table**(**void**)**

**{**

unsignedlong**\***ttb **=(**unsignedlong**\*)**0x30000000**;**

unsignedlong vaddr**,** paddr**;**

vaddr **=**0xA0000000**;** /\* 这里把LED\_GPIO地址转化 \*/

paddr **=**0x56000000**;**

/\* \*(TTB+虚拟高12位) = (物理高12位)|MMU配置位\*/

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC**;**

vaddr **=**0x30000000**;**

paddr **=**0x30000000**;**

**while(**vaddr **<**0x34000000**)** /\* 64MB \*/

**{**

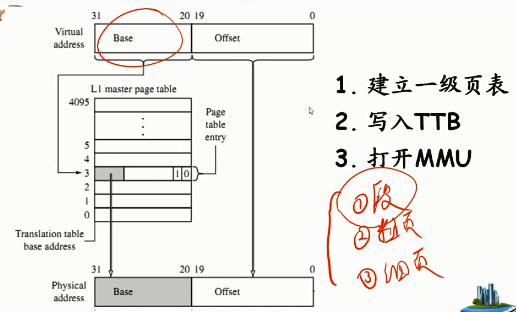
**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC\_WB**;**

vaddr **+=**0x100000**;**

paddr **+=**0x100000**;**

**}**

**}**



void mmu\_init**()**

**{**

\_\_asm\_\_**(**

/\*设置TTB\*/

"ldr r0, =0x30000000\n"

"mcr p15, 0, r0, c2, c0, 0\n"

/\*不进行权限检查\*/

"mvn r0, #0\n"

"mcr p15, 0, r0, c3, c0, 0\n"

/\*使能MMU\*/

"mrc p15, 0, r0, c1, c0, 0\n"

"orr r0, r0, #0x0001\n"

"mcr p15, 0, r0, c1, c0, 0\n"

**:**

**:**

**);**

**}**

staticvoid asm\_in\_c\_light\_led **(**void**)**

**{**

\_\_asm\_\_ volatile**(**

"ldr r1,=0x15400\n"

"str r1,[%0]\n"

"ldr r1,=0x1f\n"

"str r1,[%1]\n"

**:**

**:**"r"**(**ASM\_GPBCON**),**"r"**(**ASM\_GPBDAT**)**

**:**"r1"

**);**

**return;**

**}**

staticvoid c\_light\_led**(**void**)**

**{**

**\***C\_GPBCON **=**0x15400**;**

**\***C\_GPBDAT **=**0x1f**;** //点亮4颗LED

**return;**

**}**

int gboot\_main **(**void**)**

**{**

create\_page\_table**();**

mmu\_init**();**

//点亮4颗LED

**\*(**V\_GPBCON**)=**0x15400**;**

**\*(**V\_GPBDAT**)=**0x1f**;**

**return**0**;**

**}**

# 专题02-通过按键玩中断

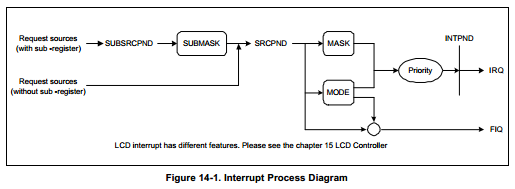
[主目录](#_目录)

## 按键中断核心理论

中断处理流程深度剖析

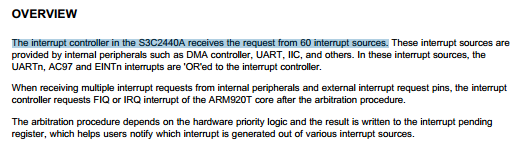
中断生命周期

轮询方式会极大影响CPU的使用率，于是现在硬件都采用中断方式来提高CPU工作效率。中断源产生中断信号，然后把这个中断信号送给中断控制器，中断控制器负责过滤这个中断信号，最后过滤之后就交给CPU来对这个中断信号做相应处理。中断处理流程可以参考芯片手册的中断控制章节如图：



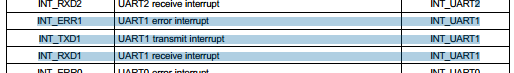
中断源

参考芯片手册的中断控制器章节，2440有60个中断源，如图：

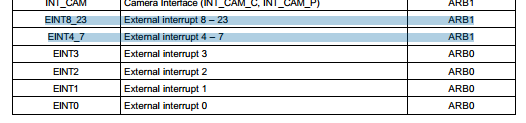


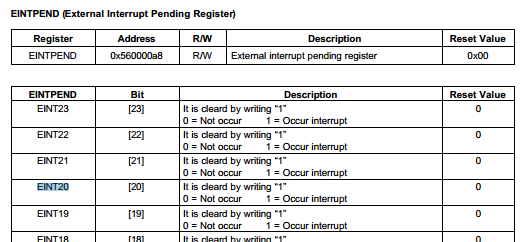
并且我们要注意到还有子中断源，比如串口1的中断又分为串口1的出错中断、发送中断和接收中断，如图：



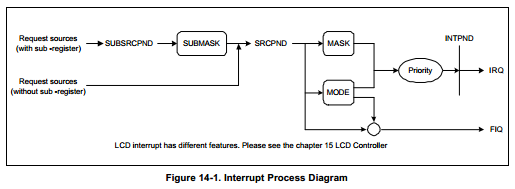


外部中断的EINT4-7和EINT8-23的中断我们完全可以通过关键字如EINT20找到相应寄存器或位来找到相应位置，例如通过该关键字我们找到了控制这两组中断组的使能开关寄存器，如图：





中断过滤



我们跟着上图的流程走，比如串口0产生了TX0子中断，那么SUBSRCPND寄存器的相应位就会被设置，然后紧接着就交给SUBMASK寄存器来过滤，然后紧接着设置SRCPND，然后紧接着走向MASK和MODE，最后根据FIQ或IRQ走优先级设置寄存器，最后交给CPU去处理中断。

中断处理

非向量中断处理总流程概述（硬件流程）

首先某硬件如按键产生中断源，然后中断控制器根据中断源类型来过滤中断源，然后最后交给CPU去执行这个中断程序。

非向量中断处理总流程概述（软件流程）

首先按下按键，然后CPU立马跳转到中断总入口，并做了相应的环境保存之后，跳转到中断处理总函数中，并读取并判断中断源，再根据中断源调用相应的中断处理函数。然后该函数处理完后清除中断并返回，最后CPU又回到总入口处并接着恢复环境。

向量中断处理总流程概述（硬件流程）

首先某硬件如按键产生中断源，然后中断控制器根据中断源类型来过滤中断源，然后最后交给CPU去执行这个中断程序。

向量中断处理总流程概述（软件流程）

首先按下按键，然后CPU直接跳转到用户设置好的中断处理程序处，为什么向量中断没有读取并判断中断源流程呢？CPU是怎么知道中断处理程序处呢？这里是由于硬件已经帮我们完成了，但是首先在中断控制器初始化时候就必须告诉中断控制器相应中断处理程序地址，比如串口0发送中断处理程序的地址就告诉中断控制器的特定寄存器或位，这样CPU执行中断程序效率就更高了，因为寻找判断中断源都是由硬件帮我们完成了，所以这就是所谓的向量中断。然后保存环境，然后直接就进行相应设备的中断处理程序了，最后恢复环境。

硬件如何产生中断或软件如何处理中断？

硬件是如何产生中断呢？比如首先我们串口要能产生中断，我们的串口得进行初始化吧，比如说我们串口接收到数据时候产生中断，那么我们得让串口工作起来对吧，那么这就涉及到串口初始化的问题，如果我们抽象一下，其实就是中断源初始化的问题；其次还有就是要对中断控制器进行初始化，设计相应的中断处理程序。

## 按键中断实验源码mini2440

非向量中断

》》》跳转到中断总入口》》》

》》》保存环境》》》

》》》调用中断处理函数》》》

》》》恢复环境》》》

### start.S

**.text**

.global \_start

\_start**:**

b reset

ldr pc**,** \_undifined\_instruction

ldr pc**,** \_software\_interrupt

ldr pc**,** \_prefetch\_abort

ldr pc**,** \_data\_abort

ldr pc**,** \_not\_used

ldr pc**,** \_irq

ldr pc**,** \_fiq

\_undifined\_instruction**:** .word undifined\_instruction

\_software\_interrupt**:** .word software\_interrupt

\_prefetch\_abort**:** .word prefetch\_abort

\_data\_abort**:** .word data\_abort

\_not\_used**:** .word not\_used

\_irq**:** .word irq

\_fiq**:** .word reset

undifined\_instruction**:**

**nop**

software\_interrupt**:**

**nop**

prefetch\_abort**:**

**nop**

data\_abort**:**

**nop**

not\_used**:**

**nop**

irq**:**

**sub** lr**,** lr **,**#4

stmfd **sp**! **,** {r0**-**r12**,**lr} **/\***保存环境**\*/**

**bl** handle\_int

ldmfd **sp**!**,** {r0**-**r12**,** pc}**^/\***恢复环境**\*/**

fiq**:**

**nop**

reset**:**

**bl** set\_svc

**bl** disable\_watchdog

**bl** disable\_interrupt

**bl** disable\_mmu

**bl** init\_clock

**bl** init\_sdram

**bl** copy\_to\_ram

**bl** init\_stack

**bl** clean\_bss

ldr pc**,=**gboot\_main

@ **bl** light\_led

set\_svc**:**

mrs r0**,** cpsr

bic r0**,** r0**,**#0x1f

orr r0**,** r0**,**#0xd3

msr cpsr**,** r0

**mov** pc**,** lr

#define pWTCON 0x53000000

disable\_watchdog**:**

ldr r0**,=**pWTCON

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_interrupt**:**

mvn r1**,** #0x0

ldr r0**,=**0x4a000008

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_mmu**:**

mcr p15**,**0**,**r0**,**c7**,**c7**,**0

mrc p15**,**0**,**r0**,**c1**,**c0**,**0

bic r0**,** r0**,** #0x00000007

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

**mov** pc**,** lr

#define CLKDIVN 0x4c000014

#define MPLLCON 0x4c000008

#define MPLL\_405MHZ **((**127**<<**12**)|(**2**<<**4**)|(**1**<<**0**))**

init\_clock**:**

ldr r0**,=**CLKDIVN

**mov** r1**,** #0x5

**str** r1**,[**r0**]**

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

orr r0**,**r0**,**#0xc0000000

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

ldr r0**,=**MPLLCON

ldr r1**,=**MPLL\_405MHZ

**str** r1**,[**r0**]**

**mov** pc**,** lr

#define mem\_contrl 0x48000000

init\_sdram**:**

ldr r0**,=**mem\_contrl

**add** r3**,** r0**,** #4**\***13

adrl r1**,** mem\_data

0**:**

ldr r2**,[**r1**],** #4

**str** r2**,[**r0**],** #4

**cmp** r0**,** r3

bne 0b

**mov** pc**,** lr

copy\_to\_ram**:**

ldr r0**,=**0x0

ldr r1**,=**0x30008000

**add** r3**,** r0**,** #1024**\***4

copy\_loop**:**

ldr r2**,[**r0**],** #4

**str** r2**,[**r1**],** #4

**cmp** r0**,** r3

bne copy\_loop

**mov** pc**,** lr

init\_stack**:**

msr cpsr\_c**,** #0xd2

ldr **sp,=**0x33000000**//**初始化r13\_irq

msr cpsr\_c**,** #0xd3

ldr **sp,=**0x34000000**//**初始化R13\_svc

**mov** pc **,**lr

clean\_bss**:**

ldr r0**,=**bss\_start

ldr r1**,=**bss\_end

**cmp** r0**,** r1

moveq pc**,** lr

clean\_loop**:**

**mov** r2**,** #0

**str** r2**,[**r0**],** #4

**cmp** r0**,** r1

bne clean\_loop

**mov** pc**,** lr

mem\_data**:**

.long 0x22000000

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00018001

.long 0x00018001

.long 0x008c04f5

.long 0x000000b1

.long 0x00000030

.long 0x00000030

#define GPBCON 0x56000010

#define GPBDAT 0x56000014

light\_led**:**

ldr r0**,=**GPBCON

**mov** r1**,** #0x400

**str** r1**,[**r0**]**

ldr r0**,=**GPBDAT

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

**mov** pc**,** lr

### mmu.c

/\*

\* 用于段描述符的一些宏定义

\*/

#define MMU\_FULL\_ACCESS (3 << 10) /\* 访问权限 \*/

#define MMU\_DOMAIN (0 << 5) /\* 属于哪个域 \*/

#define MMU\_SPECIAL (1 << 4) /\* 必须是1 \*/

#define MMU\_CACHEABLE (1 << 3) /\* cacheable \*/

#define MMU\_BUFFERABLE (1 << 2) /\* bufferable \*/

#define MMU\_SECTION (2 << 0) /\* 段描述符 \*/

#define MMU\_SECDESC (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_SECTION)

#define MMU\_SECDESC\_WB (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_CACHEABLE | MMU\_BUFFERABLE | MMU\_SECTION)

void create\_page\_table**(**void**)**

**{**

unsignedlong**\***ttb **=(**unsignedlong**\*)**0x31000000**;**

unsignedlong vaddr**,** paddr**;**

vaddr **=**0x00000000**;**

paddr **=**0x30000000**;**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xfff00000**)|** MMU\_SECDESC\_WB**;**

vaddr **=**0x56000000**;**

paddr **=**0x56000000**;**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC**;**

vaddr **=**0x4A000000**;**

paddr **=**0x4A000000**;**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xfff00000**)|** MMU\_SECDESC**;**

vaddr **=**0x30000000**;**

paddr **=**0x30000000**;**

**while(**vaddr **<**0x34000000**)**

**{**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC\_WB**;**

vaddr **+=**0x100000**;**

paddr **+=**0x100000**;**

}

}

void mmu\_enable()

{

\_\_asm\_\_(

/\*设置TTB\*/

"ldr r0, =0x31000000\n"

"mcr p15, 0, r0, c2, c0, 0\n"

/\*不进行权限检查\*/

"mvn r0, #0\n"

"mcr p15, 0, r0, c3, c0, 0\n"

/\*使能MMU\*/

"mrc p15, 0, r0, c1, c0, 0\n"

"orr r0, r0, #0x0001\n"

"mcr p15, 0, r0, c1, c0, 0\n"

:

:

);

}

void mmu\_init()

{

create\_page\_table();

mmu\_enable();

}

### Makefile

all**:** start.o main.o mmu.o led.o button.o interrupt.o

arm-linux-ld -Tgboot.lds -o gboot.elf $^

arm-linux-objcopy -O binary gboot.elf gboot.bin

%.o**:** %.S

arm-linux-gcc -g -c $^

%.o**:** %.c

arm-linux-gcc -g -c $^

.PHONY**:** clean

clean**:**

rm \*.o \*.elf \*.bin

### main.c

int gboot\_main**()**

**{**

#ifdef MMU\_ON

mmu\_init**();**

#endif

led\_init**();**

button\_init**();**

init\_irq**();**

**while(**1**);**

**return**0**;**

**}**

### led.c

#define GPBCON (volatile unsigned long\*)0x56000010

#define GPBDAT (volatile unsigned long\*)0x56000014

void led\_init**()**

**{**

**\*(**GPBCON**)=**0x400**;**

**}**

void led\_off**()**

**{**

**\*(**GPBDAT**)=**0x7ff**;**

**}**

void led\_on**()**

**{**

**\*(**GPBDAT**)=**0x0**;**

**}**

### interrupt.c

/\*interrupt registes\*/

#define SRCPND (volatile unsigned long \*)0x4A000000

#define INTMOD (volatile unsigned long \*)0x4A000004

#define INTMSK (volatile unsigned long \*)0x4A000008

#define PRIORITY (volatile unsigned long \*)0x4A00000c

#define INTPND (volatile unsigned long \*)0x4A000010

#define INTOFFSET (volatile unsigned long \*)0x4A000014

#define SUBSRCPND (volatile unsigned long \*)0x4A000018

#define INTSUBMSK (volatile unsigned long \*)0x4A00001c

#define EINTMASK (volatile unsigned long \*)0x560000a4

#define EINTPEND (volatile unsigned long \*)0x560000a8

void init\_irq**()**

**{**

// 在EINTMASK寄存器中使能它们

**\*(**EINTMASK**)&=(~(**1**<<**8**))&(~(**1**<<**11**))&(~(**1**<<**13**))&(~(**1**<<**14**));**

// EINT8\_23使能

**\*(**INTMSK**)&=~(**1**<<**5**);**

\_\_asm\_\_**(**

/\*开中断\*/

"mrs r0,cpsr\n"

"bic r0, r0, #0x80\n"

"msr cpsr\_c, r0\n"

**:**

**:**

**);**

**}**

void handle\_int**()**

**{**

/\*读取产生中断的源\*/

unsignedlong value **=\*(**EINTPEND**)&((**1**<<**8**)|(**1**<<**11**)|(**1**<<**13**)|(**1**<<**14**));**

**switch(**value**)**

**{**

**case(**1**<<**8**):**//K1

led\_on**();**

**break;**

**case(**1**<<**11**):**//K2

led\_off**();**

**break;**

**case(**1**<<**13**):**//K3

led\_on**();**

**break;**

**case(**1**<<**14**):**//K4

led\_off**();**

**break;**

**default:**

**break;**

**}**

/\* 中断清除 \*/

**\*(**EINTPEND**)=** value**;**

**\*(**SRCPND**)=**1**<<**5**;**

**\*(**INTPND**)=**1**<<**5**;**

**}**

### gboot.lds

OUTPUT\_ARCH(arm)

ENTRY(\_start)

SECTIONS {

.**=** 0x30008000;

.**=** ALIGN(4);

.text :

{

**start**.o (.text)

**\***(.text)

}

.**=** ALIGN(4);

.data :

{

**\***(.data)

}

.**=** ALIGN(4);

bss\_start**=** .;

.bss :

{

**\***(.bss)

}

bss\_end**=** .;

}

### button.c

#define GPGCON (volatile unsigned long \*)0x56000060

/\*

\* K1,K2,K3,K4对应GPG0、GPG3、GPG5、GPG6

\*/

#define GPG0\_int (0x2<<(0\*2))

#define GPG3\_int (0x2<<(3\*2))

#define GPG5\_int (0x2<<(5\*2))

#define GPG6\_int (0x2<<(6\*2))

#define GPG0\_msk (3<<(0\*2))

#define GPG3\_msk (3<<(3\*2))

#define GPG5\_msk (3<<(5\*2))

#define GPG6\_msk (3<<(6\*2))

void button\_init**()**

**{**

**\*(**GPGCON**)&=~(**GPG0\_msk **|** GPG3\_msk **|** GPG5\_msk **|** GPG6\_msk**);**

**\*(**GPGCON**)|=** GPG0\_int **|** GPG3\_int **|** GPG5\_int **|** GPG6\_int**;**

**}**

## 按键中断实验源码OK6410

向量中断

当中断产生时CPU直接跳转到用户设置好的中断处理程序处》》》

保存环境》》》

设备的中断处理》》》

恢复环境》

### start.S

**.text**

.global \_start

\_start**:**

b reset

ldr pc**,** \_undifined\_instruction

ldr pc**,** \_software\_interrupt

ldr pc**,** \_prefetch\_abort

ldr pc**,** \_data\_abort

ldr pc**,** \_not\_used

ldr pc**,** \_irq

ldr pc**,** \_fiq

\_undifined\_instruction**:** .word undifined\_instruction

\_software\_interrupt**:** .word software\_interrupt

\_prefetch\_abort**:** .word prefetch\_abort

\_data\_abort**:** .word data\_abort

\_not\_used**:** .word not\_used

\_irq**:** .word irq

\_fiq**:** .word reset

undifined\_instruction**:**

**nop**

software\_interrupt**:**

**nop**

prefetch\_abort**:**

**nop**

data\_abort**:**

**nop**

not\_used**:**

**nop**

irq**:**

**nop**

fiq**:**

**nop**

reset**:**

**bl** set\_svc

**bl** set\_peri\_port

**bl** disable\_watchdog

**bl** disable\_interrupt

**bl** disable\_mmu

**bl** init\_clock

**bl** mem\_init

**bl** copy\_to\_ram

**bl** init\_stack

**bl** clean\_bss

ldr pc**,=**gboot\_main

@ **bl** light\_led

set\_peri\_port**:**

ldr r0**,=**0x70000000

orr r0**,** r0**,** #0x13

mcr p15**,**0**,**r0**,**c15**,**c2**,**4

**mov** pc**,** lr

set\_svc**:**

mrs r0**,** cpsr

bic r0**,** r0**,**#0x1f

orr r0**,** r0**,**#0xd3

msr cpsr**,** r0

**mov** pc**,** lr

#define pWTCON 0x7e004000

disable\_watchdog**:**

ldr r0**,=**pWTCON

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_interrupt**:**

mvn r1**,**#0x0

ldr r0**,=**0x71200014

**str** r1**,[**r0**]**

ldr r0**,=**0x71300014

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_mmu**:**

mcr p15**,**0**,**r0**,**c7**,**c7**,**0

mrc p15**,**0**,**r0**,**c1**,**c0**,**0

bic r0**,** r0**,** #0x00000007

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

**mov** pc**,** lr

#define CLK\_DIV0 0x7e00f020

#define OTHERS 0x7e00f900

#define MPLL\_CON 0x7e00f010

#define APLL\_CON 0x7e00f00c

#define CLK\_SRC 0x7e00f01c

#define DIV\_VAL **((**0x0**<<**0**)|(**0x1**<<**9**)|(**0x1**<<**8**)|(**0x3**<<**12**))**

#define PLL\_VAL **((**1**<<**31**)|(**266**<<**16**)|(**3**<<**8**)|(**1**<<**0**))**

init\_clock**:**

ldr r0**,=**CLK\_DIV0

ldr r1**,=**DIV\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**OTHERS

ldr r1**,[**r0**]**

bic r1**,**r1**,**#0xc0

**str** r1**,[**r0**]**

ldr r0**,=**APLL\_CON

ldr r1**,=**PLL\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**MPLL\_CON

ldr r1**,=**PLL\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**CLK\_SRC

**mov** r1**,** #0x3

**str** r1**,[**r0**]**

**mov** pc**,** lr

copy\_to\_ram**:**

ldr r0**,=**0x0c000000

ldr r1**,=**0x50008000

**add** r3**,** r0**,** #1024**\***4

copy\_loop**:**

ldr r2**,[**r0**],** #4

**str** r2**,[**r1**],** #4

**cmp** r0**,** r3

bne copy\_loop

**mov** pc**,** lr

init\_stack**:**

msr cpsr\_c**,** #0xd2

ldr **sp,=**0x53000000**//**初始化r13\_irq

msr cpsr\_c**,** #0xd3

ldr **sp,=**0x54000000**//**初始化R13\_svc

**mov** pc **,**lr

clean\_bss**:**

ldr r0**,=**bss\_start

ldr r1**,=**bss\_end

**cmp** r0**,** r1

moveq pc**,** lr

clean\_loop**:**

**mov** r2**,** #0

**str** r2**,[**r0**],** #4

**cmp** r0**,** r1

bne clean\_loop

**mov** pc**,** lr

light\_led**:**

ldr r1**,=**0x7F008820

ldr r0**,=**0x1111

**str** r0**,[**r1**]**

ldr r1**,=**0x7F008824

**mov** r0**,**#0xe

**str** r0**,[**r1**]**

**mov** pc**,** lr

### mmu.c

/\*

\* 用于段描述符的一些宏定义

\*/

#define MMU\_FULL\_ACCESS (3 << 10) /\* 访问权限 \*/

#define MMU\_DOMAIN (0 << 5) /\* 属于哪个域 \*/

#define MMU\_SPECIAL (1 << 4) /\* 必须是1 \*/

#define MMU\_CACHEABLE (1 << 3) /\* cacheable \*/

#define MMU\_BUFFERABLE (1 << 2) /\* bufferable \*/

#define MMU\_SECTION (2) /\* 表示这是段描述符 \*/

#define MMU\_SECDESC (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_SECTION)

#define MMU\_SECDESC\_WB (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_CACHEABLE | MMU\_BUFFERABLE | MMU\_SECTION)

void create\_page\_table**(**void**)**

**{**

unsignedlong**\***ttb **=(**unsignedlong**\*)**0x50000000**;**

unsignedlong vaddr**,** paddr**;**

vaddr **=**0xA0000000**;**

paddr **=**0x7f000000**;**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC**;**

vaddr **=**0x50000000**;**

paddr **=**0x50000000**;**

**while(**vaddr **<**0x54000000**)**

**{**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC\_WB**;**

vaddr **+=**0x100000**;**

paddr **+=**0x100000**;**

**}**

**}**

void mmu\_enable**()**

**{**

\_\_asm\_\_**(**

/\*设置TTB\*/

"ldr r0, =0x50000000\n"

"mcr p15, 0, r0, c2, c0, 0\n"

/\*不进行权限检查\*/

"mvn r0, #0\n"

"mcr p15, 0, r0, c3, c0, 0\n"

/\*使能MMU\*/

"mrc p15, 0, r0, c1, c0, 0\n"

"orr r0, r0, #0x0001\n"

"mcr p15, 0, r0, c1, c0, 0\n"

:

:

);

}

void mmu\_init()

{

create\_page\_table();

mmu\_enable();

}

### mem.S

**.text**

.global mem\_init

mem\_init**:**

ldr r0**,=**0x7e00f120

**mov** r1**,** #0x8

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001004 @内存控制命令寄存器

**mov** r1**,** #0x4 @根据手册得知需要先进入配置模式

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001010 @刷新寄存器地址

ldr r1**,=((**7800**/(**1000000000**/**133000000**)+**1**))** @设置刷新时间

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001014 @CAS latency寄存器

**mov** r1**,** #**(**3**<<**1**)**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001018 @t\_DQSS寄存器

**mov** r1**,** #0x1

**str** r1**,[**r0**]**

ldr r0**,=**0x7e00101c @T\_MRD寄存器

**mov** r1**,** #0x2

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001020 @t\_RAS寄存器

ldr r1**,=((**45**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001024 @t\_RC寄存器

ldr r1**,=((**68**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001028 @t\_RCD寄存器

ldr r1**,=((**23**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e00102c @t\_RFC寄存器

ldr r1**,=((**80**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001030 @t\_RP寄存器

ldr r1**,=((**23**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001034 @t\_rrd寄存器

ldr r1**,=((**15**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001038 @t\_wr寄存器

ldr r1**,=((**15**/(**1000000000**/**133000000**)+**1**))**

@ ldr r2**,[**r0**]**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e00103c @t\_wtr寄存器

**mov** r1**,** #0x07

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001040 @t\_xp寄存器

**mov** r1**,** #0x02

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001044 @t\_xsr寄存器

ldr r1**,=((**120**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001048 @t\_esr寄存器

ldr r1**,=((**120**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e00100c @内存控制配置寄存器

ldr r1**,=**0x00010012 @配置控制器

**str** r1**,[**r0**]**

ldr r0**,=**0x7e00104c @32位DRAM配置控制寄存器

ldr r1**,=**0x0b45

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001200 @片选寄存器

ldr r1**,=**0x150f8

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001304 @用户配置寄存器

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001008

ldr r1**,=**0x000c0000

**str** r1**,[**r0**]**

ldr r1**,=**0x00000000

**str** r1**,[**r0**]**

ldr r1**,=**0x00040000

**str** r1**,[**r0**]**

ldr r1**,=**0x000a0000

**str** r1**,[**r0**]**

ldr r1**,=**0x00080032

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001004

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

check\_dmc1\_ready**:**

ldr r0**,=**0x7e001000

ldr r1**,[**r0**]**

**mov** r2**,** #0x3

**and** r1**,** r1**,** r2

**cmp** r1**,** #0x1

bne check\_dmc1\_ready

**nop**

**mov** pc**,** lr

### led.c

#define GPKCON (volatile unsigned long\*)0x7f008800

#define GPKDAT (volatile unsigned long\*)0x7f008808

void led\_init**()**

**{**

**\*(**GPKCON**)=**0x1111**;**

**}**

void led\_on**()**

**{**

**\*(**GPKDAT**)=**0x0e**;**

**}**

void led\_off**()**

**{**

**\*(**GPKDAT**)=**0x0f**;**

**}**

### interrupt.c

/\*interrupt registes\*/

#define EINT0CON0 \*((volatile unsigned int \*)0x7f008900)

#define EINT0MASK \*((volatile unsigned int \*)0x7f008920)

#define EINT0PEND \*((volatile unsigned int \*)0x7f008924)

#define VIC0INTENABLE \*((volatile unsigned int \*)0x71200010)

#define EINT0\_VECTADDR \*((volatile unsigned int \*)0x71200100)

#define EINT5\_VECTADDR \*((volatile unsigned int \*)0x71200104)

#define VIC0ADDRESS \*((volatile unsigned int \*)0x71200f00)

#define VIC1ADDRESS \*((volatile unsigned int \*)0x71300f00)

void key1\_handle**()**

**{**

\_\_asm\_\_**(**

"sub lr, lr, #4\n"

"stmfd sp!, {r0-r12, lr}\n"

**:**

**:**

**);**

led\_on**();**

/\* 清除中断 \*/

EINT0PEND **=~**0x0**;**

VIC0ADDRESS **=**0**;**

VIC1ADDRESS **=**0**;**

\_\_asm\_\_**(**

"ldmfd sp!, {r0-r12, pc}^ \n"

**:**

**:**

**);**

**}**

void key6\_handle**()**

**{**

\_\_asm\_\_**(**

"sub lr, lr, #4\n"

"stmfd sp!, {r0-r12, lr}\n"

**:**

**:**

**);**

led\_off**();**

/\* 清除中断 \*/

EINT0PEND **=~**0x0**;**

VIC0ADDRESS **=**0**;**

VIC1ADDRESS **=**0**;**

\_\_asm\_\_**(**

"ldmfd sp!, {r0-r12, pc}^ \n"

**:**

**:**

**);**

**}**

void init\_irq**()**

**{**

EINT0CON0 **=(**0b010**)|(**0b010**<<**8**);**/\* 配置为下降沿触发 \*/

EINT0MASK **=**0**;**/\* 取消屏蔽外部中断 \*/

VIC0INTENABLE **|=(**0b1**)|(**0b10**);**/\* 使能外部中断\*/

EINT0\_VECTADDR **=(**int**)**key1\_handle**;**/\* 用户按下key时，CPU就会自动的将VIC0VECTADDR0的值赋给VIC0ADDRESS并跳转到这个地址去执 \*/

EINT5\_VECTADDR **=(**int**)**key6\_handle**;**

\_\_asm\_\_**(**

"mrc p15,0,r0,c1,c0,0\n"

"orr r0,r0,#(1<<24)\n"

"mcr p15,0,r0,c1,c0,0\n"

"mrs r0,cpsr\n"

"bic r0, r0, #0x80\n"

"msr cpsr\_c, r0\n"

**:**

**:**

**);**

**}**

### gboot.lds

OUTPUT\_ARCH**(**arm**)**

ENTRY**(**\_start**)**

SECTIONS **{**

**.=**0x50008000**;**

**.=** ALIGN**(**4**);**

**.**text **:**

**{**

start.o **(.**text**)**

**\*(.**text**)**

**}**

**.=** ALIGN**(**4**);**

**.**data **:**

**{**

**\*(.**data**)**

**}**

**.=** ALIGN**(**4**);**

bss\_start **=.;**

**.**bss **:**

**{**

**\*(.**bss**)**

**}**

bss\_end **=.;**

**}**

### button.c

#define GPNCON (volatile unsigned long\*)0x7f008830

void button\_init**()**

**{**

**\*(**GPNCON**)=**0b10**|(**0b10**<<**10**);**//设置按键1与按键6

**}**

### main.c

int gboot\_main**()**

**{**

#ifdef MMU\_ON

mmu\_init**();**

#endif

led\_init**();**

button\_init**();**

init\_irq**();**

led\_off**();**

**while(**1**);**

**return**0**;**

**}**

### Makefile

all**:** start.o main.o mmu.o led.o button.o interrupt.o mem.o

arm-linux-ld -Tgboot.lds -o gboot.elf $^

arm-linux-objcopy -O binary gboot.elf gboot.bin

%.o**:** %.S

arm-linux-gcc -g -c $^

%.o**:** %.c

arm-linux-gcc -g -c $^

.PHONY**:** clean

clean**:**

rm \*.o \*.elf \*.bin

## 按键中断实验源码TQ210

### start.S

@**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

@File**:**start.S

@

@成都国嵌嵌入式培训中心

@**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**.text**

.global \_start

\_start**:**

b reset

ldr pc**,** \_undefined\_instruction

ldr pc**,** \_software\_interrupt

ldr pc**,** \_prefetch\_abort

ldr pc**,** \_data\_abort

ldr pc**,** \_not\_used

ldr pc**,** \_irq

ldr pc**,** \_fiq

\_undefined\_instruction**:** .word undefined\_instruction

\_software\_interrupt**:** .word software\_interrupt

\_prefetch\_abort**:** .word prefetch\_abort

\_data\_abort**:** .word data\_abort

\_not\_used**:** .word not\_used

\_irq**:** .word irq

\_fiq**:** .word fiq

undefined\_instruction**:**

**nop**

software\_interrupt**:**

**nop**

prefetch\_abort**:**

**nop**

data\_abort**:**

**nop**

not\_used**:**

**nop**

irq**:**

**nop**

fiq**:**

**nop**

reset**:**

**bl** set\_svc

**bl** disable\_watchdog

**bl** disable\_interrupt

**bl** disable\_mmu

**bl** init\_clock

**bl** mem\_init

**bl** copy\_to\_ram

**bl** init\_stack

**bl** clean\_bss

ldr pc**,=**gboot\_main

@ **bl** light\_led

set\_svc**:**

mrs r0**,** cpsr

bic r0**,** r0**,** #0x1f

orr r0**,** r0**,** #0xd3

msr cpsr**,** r0

**mov** pc**,** lr

#define pWTCON 0xE2700000

disable\_watchdog**:**

ldr r0**,=**pWTCON

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_interrupt**:**

mvn r1**,**#0x0

ldr r0**,=**0xf2000014

**str** r1**,[**r0**]**

ldr r0**,=**0xf2100014

**str** r1**,[**r0**]**

ldr r0**,=**0xf2200014

**str** r1**,[**r0**]**

ldr r0**,=**0xf2300014

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_mmu**:**

mcr p15**,**0**,**r0**,**c7**,**c5**,**0

mcr p15**,**0**,**r0**,**c7**,**c6**,**1

mrc p15**,**0**,**r0**,**c1**,**c0**,**0

bic r0**,** r0**,** #0x00000007

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

**mov** pc**,** lr

#define CLK\_DIV0 0xe0100300

#define MPLL\_CON 0xe0100108

#define APLL\_CON 0xe0100100

#define CLK\_SRC 0xe0100200

#define DIV\_VAL **((**0x0**<<**0**)|(**4**<<**4**)|(**4**<<**8**)|(**1**<<**12**)|(**3**<<**16**)|(**1**<<**20**)|(**4**<<**24**)|(**1**<<**28**))**

#define APLL\_VAL **((**1**<<**31**)|(**3**<<**8**)|(**125**<<**16**)|(**1**<<**0**))**

#define MPLL\_VAL **((**1**<<**31**)|(**12**<<**8**)|(**667**<<**16**)|(**1**<<**0**))**

init\_clock**:**

ldr r0**,=**CLK\_DIV0

ldr r1**,=**DIV\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**APLL\_CON

ldr r1**,=**APLL\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**MPLL\_CON

ldr r1**,=**MPLL\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**CLK\_SRC

ldr r1**,=**0x1111

**str** r1**,[**r0**]**

**mov** pc**,** lr

copy\_to\_ram**:**

ldr r0**,=**0xd0020010

ldr r1**,=**0x20008000

**add** r3**,** r0**,** #1024**\***4

copy\_loop**:**

ldr r2**,[**r0**],** #4

**str** r2**,[**r1**],** #4

**cmp** r0**,** r3

bne copy\_loop

**mov** pc**,** lr

init\_stack**:**

msr cpsr\_c**,** #0xd2

ldr **sp,=**0x23000000 @此处实际设置的是r13\_irq

msr cpsr\_c**,** #0xd3

ldr **sp,=**0x24000000 @此处实际设置的是r13\_svc

**mov** pc**,** lr

init\_stacktmp**:**

ldr **sp,=**0xd0030000

**mov** pc **,**lr

clean\_bss**:**

ldr r0**,=**bss\_start

ldr r1**,=**bss\_end

**cmp** r0**,** r1

moveq pc**,** lr

clean\_loop**:**

**mov** r2**,** #0

**str** r2**,[**r0**],** #4

**cmp** r0**,** r1

bne clean\_loop

**mov** pc**,** lr

light\_led**:**

ldr R0**,=**0xE0200060

ldr R1**,=**0x00011000

**str** R1**,[**R0**]**

ldr R0**,=**0xE0200064

ldr R1**,=**0x00000018

**str** R1**,[**R0**]**

**mov** pc**,** lr

### mem.S

#define DMC\_PHYCONTROL0 0xf0000018

#define DMC\_PHYCONTROL1 0xf000001c

#define DMC\_CONCONTROL 0xf0000000

#define DMC\_MEMCONTROL 0xf0000004

#define DMC\_MEMCONFIG0 0xf0000008

#define DMC\_MEMCONFIG1 0xf000000c

#define DMC\_PRECHCONFIG 0xf0000014

#define DMC\_TIMINGAREF 0xf0000030

#define DMC\_TIMINGROW 0xf0000034

#define DMC\_TIMINGDATA 0xf0000038

#define DMC\_TIMINGPOWER 0xf000003c

#define DMC\_PHYSTATUS 0xf0000040

#define DMC\_DIRECTCMD 0xf0000010

#define DMC\_PWRDNCONFIG 0xf0000028

#define DMC0\_MEMCONTROL 0x00202400

#define DMC0\_MEMCONFIG\_0 0x20F00313

#define DMC0\_MEMCONFIG\_1 0x00F00313

#define DMC0\_TIMINGA\_REF 0x00000618

#define DMC0\_TIMING\_ROW 0x2B34438A

#define DMC0\_TIMING\_DATA 0x24240000

#define DMC0\_TIMING\_PWR 0x0BDC0343

.globl mem\_init

mem\_init**:**

ldr r0**,=**DMC\_PHYCONTROL0

ldr r1**,=**0x00101000

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_PHYCONTROL1

ldr r1**,=**0x00000086

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_PHYCONTROL0

ldr r1**,=**0x00101002

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_PHYCONTROL0

ldr r1**,=**0x00101003

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_CONCONTROL

ldr r1**,=**0x0FFF2010

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_MEMCONTROL

ldr r1**,=**DMC0\_MEMCONTROL

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_MEMCONFIG0

ldr r1**,=**DMC0\_MEMCONFIG\_0

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_MEMCONFIG1

ldr r1**,=**DMC0\_MEMCONFIG\_1

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_PRECHCONFIG

ldr r1**,=**0xFF000000

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_TIMINGAREF

ldr r1**,=**DMC0\_TIMINGA\_REF

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_TIMINGROW

ldr r1**,=**DMC0\_TIMING\_ROW

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_TIMINGDATA

ldr r1**,=**DMC0\_TIMING\_DATA

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_TIMINGPOWER

ldr r1**,=**DMC0\_TIMING\_PWR

**str** r1**,[**r0**]**

wait\_lock**:**

**//Loop** until DLL is locked

ldr r0**,=**DMC\_PHYSTATUS

ldr r1**,[**r0**]**

**and** r2**,** r1**,** #0x4

**cmp** r2**,** #0x4

bne wait\_lock

ldr r0**,=**DMC\_DIRECTCMD

ldr r1**,=**0x07000000

**str** r1**,[**r0**]**

ldr r1**,=**0x01000000

**str** r1**,[**r0**]**

ldr r1**,=**0x00020000

**str** r1**,[**r0**]**

ldr r1**,=**0x00030000

**str** r1**,[**r0**]**

ldr r1**,=**0x00010400

**str** r1**,[**r0**]**

ldr r1**,=**0x00000542

**str** r1**,[**r0**]**

ldr r1**,=**0x01000000

**str** r1**,[**r0**]**

ldr r1**,=**0x05000000

**str** r1**,[**r0**]**

ldr r1**,=**0x05000000

**str** r1**,[**r0**]**

ldr r1**,=**0x00000442

**str** r1**,[**r0**]**

ldr r1**,=**0x00010780

**str** r1**,[**r0**]**

ldr r1**,=**0x00010400

**str** r1**,[**r0**]**

ldr r1**,=**0x07100000

**str** r1**,[**r0**]**

ldr r1**,=**0x01100000

**str** r1**,[**r0**]**

ldr r1**,=**0x00120000

**str** r1**,[**r0**]**

ldr r1**,=**0x00130000

**str** r1**,[**r0**]**

ldr r1**,=**0x00110400

**str** r1**,[**r0**]**

ldr r1**,=**0x00100542

**str** r1**,[**r0**]**

ldr r1**,=**0x01100000

**str** r1**,[**r0**]**

ldr r1**,=**0x05100000

**str** r1**,[**r0**]**

ldr r1**,=**0x05100000

**str** r1**,[**r0**]**

ldr r1**,=**0x00100442

**str** r1**,[**r0**]**

ldr r1**,=**0x00110780

**str** r1**,[**r0**]**

ldr r1**,=**0x00110400

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_CONCONTROL

ldr r1**,=**0x0FF02030

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_PWRDNCONFIG

ldr r1**,=**0xFFFF00FF

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_CONCONTROL

ldr r1**,=**0x00202400

**str** r1**,[**r0**]**

**mov** pc**,** lr

### mmu.c

/\*

\* 用于段描述符的一些宏定义

\*/

#define MMU\_FULL\_ACCESS (3 << 10) /\* 访问权限 \*/

#define MMU\_DOMAIN (0 << 5) /\* 属于哪个域 \*/

#define MMU\_SPECIAL (1 << 4) /\* 必须是1 \*/

#define MMU\_CACHEABLE (1 << 3) /\* cacheable \*/

#define MMU\_BUFFERABLE (1 << 2) /\* bufferable \*/

#define MMU\_SECTION (2) /\* 表示这是段描述符 \*/

#define MMU\_SECDESC (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_SECTION)

#define MMU\_SECDESC\_WB (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_CACHEABLE | MMU\_BUFFERABLE | MMU\_SECTION)

void create\_page\_table**(**void**)**

**{**

unsignedlong**\***ttb **=(**unsignedlong**\*)**0x20000000**;**

unsignedlong vaddr**,** paddr**;**

vaddr **=**0xA0000000**;**

paddr **=**0xE0200000**;**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC**;**

vaddr **=**0x20000000**;**

paddr **=**0x20000000**;**

**while(**vaddr **<**0x24000000**)**

**{**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC\_WB**;**

vaddr **+=**0x100000**;**

paddr **+=**0x100000**;**

**}**

**}**

void mmu\_enable**()**

**{**

\_\_asm\_\_**(**

/\*设置TTB\*/

"ldr r0, =0x20000000\n"

"mcr p15, 0, r0, c2, c0, 0\n"

/\*不进行权限检查\*/

"mvn r0, #0\n"

"mcr p15, 0, r0, c3, c0, 0\n"

/\*使能MMU\*/

"mrc p15, 0, r0, c1, c0, 0\n"

"orr r0, r0, #0x0001\n"

"mcr p15, 0, r0, c1, c0, 0\n"

:

:

);

}

void mmu\_init()

{

create\_page\_table();

mmu\_enable();

}

### interrupt.c

/\*interrupt registes\*/

#define EXT\_INT\_0\_CON \*((volatile unsigned int \*)0xE0200E00)

#define EXT\_INT\_0\_MASK \*((volatile unsigned int \*)0xE0200F00)

#define VIC0INTSELECT \*((volatile unsigned int \*)0xF200000C)

#define VIC0INTENABLE \*((volatile unsigned int \*)0xF2000010)

#define key1\_VICADDR \*((volatile unsigned int \*)0xF2000100)

#define key2\_VICADDR \*((volatile unsigned int \*)0xF2000104)

#define VIC0ADDRESS \*((volatile unsigned int \*)0xF2000F00)

#define EXT\_INT\_0\_PEND \*((volatile unsigned int \*)0xE0200F40)

void key1\_isr**()**

**{**

\_\_asm\_\_**(**

"sub lr, lr, #4\n"

"stmfd sp!, {r0-r12, lr}\n"

**:**

**:**

**);**

led\_on**();**

/\* 清除中断 \*/

EXT\_INT\_0\_PEND **=~**0x0**;**

VIC0ADDRESS **=**0**;**

\_\_asm\_\_**(**

"ldmfd sp!, {r0-r12, pc}^ \n"

**:**

**:**

**);**

**}**

void key2\_isr**()**

**{**

\_\_asm\_\_**(**

"sub lr, lr, #4\n"

"stmfd sp!, {r0-r12, lr}\n"

**:**

**:**

**);**

led\_off**();**

/\* 清除中断 \*/

EXT\_INT\_0\_PEND **=~**0x0**;**

VIC0ADDRESS **=**0**;**

\_\_asm\_\_**(**

"ldmfd sp!, {r0-r12, pc}^ \n"

**:**

**:**

**);**

**}**

void init\_irq**()**

**{**

EXT\_INT\_0\_CON **&=~(**0xFF**<<**0**);**

EXT\_INT\_0\_CON **|=**2**|(**2**<<**4**);**/\* 配置EXT\_INT[0]和EXT\_INT[1]为下降沿触发 \*/

EXT\_INT\_0\_MASK **&=~**3**;**/\* 取消屏蔽外部中断EXT\_INT[0]和EXT\_INT[1] \*/

VIC0INTENABLE **|=**3**;**/\* 使能外部中断EXT\_INT[0]和EXT\_INT[1] \*/

key1\_VICADDR **=(**int**)**key1\_isr**;**/\* 当EXT\_INT[0]触发中断，即用户按下key时，

CPU就会自动的将VIC0VECTADDR0的值赋给VIC0ADDRESS并跳转到这个地址去执 \*/

key2\_VICADDR **=(**int**)**key2\_isr**;**

\_\_asm\_\_**(**

/\*开中断\*/

"mrs r0,cpsr\n"

"bic r0, r0, #0x80\n"

"msr cpsr\_c, r0\n"

**:**

**:**

**);**

**}**

### led.c

#define GPC0CON \*((volatile unsigned long\*)0xE0200060)

#define GPC0DAT \*((volatile unsigned long\*)0xE0200064)

void led\_init**()**

**{**

GPC0CON **=**0x00011000**;**

**}**

void led\_off**()**

**{**

GPC0DAT **=**0x0**;**

**}**

void led\_on**()**

**{**

GPC0DAT **=**0x00000018**;**

**}**

### button.c

#define GPH0CON \*((volatile unsigned long \*)0xe0200c00)

/\*

\* K1,K2,K3,K4对应GPH0\_1、GPH0\_2、GPH0\_3、GPH0\_4

\*/

#define GPH0\_int (0xf<<(0\*4))

#define GPH1\_int (0xf<<(1\*4))

#define GPH2\_int (0xf<<(2\*4))

#define GPH4\_int (0xf<<(4\*4))

#define GPH0\_msk (0xf<<(0\*4))

#define GPH1\_msk (0xf<<(1\*4))

#define GPH2\_msk (0xf<<(2\*4))

#define GPH4\_msk (0xf<<(4\*4))

void button\_init**()**

**{**

GPH0CON **&=~(**GPH0\_msk **|** GPH1\_msk **|** GPH2\_msk **|** GPH4\_msk**);**

GPH0CON **|=** GPH0\_int **|** GPH1\_int **|** GPH2\_int **|** GPH4\_int**;**

**}**

### main.c

int gboot\_main**()**

**{**

#ifdef MMU\_ON

mmu\_init**();**

#endif

led\_init**();**

button\_init**();**

init\_irq**();**

**while(**1**);**

**return**0**;**

**}**

### gboot.lds

OUTPUT\_ARCH**(**arm**)**

ENTRY**(**\_start**)**

SECTIONS **{**

**.=**0x20008000**;**

**.=** ALIGN**(**4**);**

**.**text **:**

**{**

start.o **(.**text**)**

**\*(.**text**)**

**}**

**.=** ALIGN**(**4**);**

**.**data **:**

**{**

**\*(.**data**)**

**}**

**.=** ALIGN**(**4**);**

bss\_start **=.;**

**.**bss **:**

**{**

**\*(.**bss**)**

**}**

bss\_end **=.;**

**}**

### Makefile

all**:** start.o mem.o main.o button.o mmu.o interrupt.o led.o

arm-linux-ld -Tgboot.lds -o gboot.elf $^

arm-linux-objcopy -O binary gboot.elf gboot.bin

%.o**:** %.S

arm-linux-gcc -g -c $^

%.o**:** %.c

arm-linux-gcc -g -c $^

.PHONY**:** clean

clean**:**

rm \*.o \*.elf \*.bin

## 按键中断实验210故障排除

上节课中断初始化流程回顾：

首先我们初始化了按键，然后初始化了中断，最后呢我们也实现了中断处理函数，那么最后我们要达到的目的是按下按键，通过中断方式点亮LED。那么问题了，在210平台时，当我们按下按键之后，LED没有亮！

那么我们如何来分析这个问题呢？在这个问题当中，我们的两个流程被关注，第一是按键初始化，第二是中断初始化，但是我们两个一起查也不好查，所以我们可以采用简化法，比如我们只涉及到按键，就先暂时不使用中断，那么我们如何让按键实现工作呢？答案就是轮询，所以我们就可以把中断部分注释掉，如果说按键能以轮询方式工作，那么我们这个按键初始化是没问题的。

LED轮询的改造代码

#define GPH2DAT \*((volatile unsigned long \*)0xe0200c44)

int gboot\_main**()**

**{**

#ifdef MMU\_ON

mmu\_init**();**

#endif

led\_init**();**

button\_init**();**

//init\_irq();

unsignedint data**;**

**while(**1**)**

**{**

data **=** GPH2DAT**;**

data **&=(**1**<<**0**)**

**if(**data **==**1**)**

**{**

led\_off**();** //led\_on();也不亮，于是这个地方卡住了，于是我们能想到的办法就是反汇编来分析代码

**}**

**else**

**{**

led\_on**();**

**}**

**}**

**return**0**;**

**}**

上述代码不管if任何真条件LED都不亮，那么我们故意设置一个真值在if里，结果LED还是不亮，那么这个就和我们的LED就没有关系了，问题是我们的栈出现问题了，于是我们又打开start.S定位到栈初始化位置，但是检查栈也没问题，于是我们又开始怀疑内存初始化有问题了，于是我们又定位到内存初始化位置，由于内存初始化代码量比较大，我们于是就到uboot相应位置找到，并补上我们没有做的内存初始化代码，然后编译烧写测试发现LED立刻被点亮，于是我们把LED改回中断方式点亮，然后编译烧写测试发现当我们按下按键时LED能够被点亮了！

# 专题03-NandFlash变硬盘

[主目录](#_目录)

## 硬件核心理论

[主目录](#_目录)

nandflash裸机驱动博客：<http://blog.csdn.net/feihuxiaozi/article/details/6943124>

nandflash分类

SLC（single level cell）:单层次存储

MLC（Multi level cell）:多层次存储

SLC在每个存储格上只存放一位数据，MLC在每个存储格上两位数据。

价格：同容量MLC价格上远低于SLC。

访问速度：SLC要比MLC快3倍以上。

使用寿命：SLC能进行10万次擦写，MLC只能1万次。

功耗：MLC功耗比SLC高15%左右。

nandflash访问方式

编址方式、地址结构、信号引脚。

\*（0x30008000）= 0xabc;

nandflash属于独立编址设备，那CPU如何控制nandflash呢？

因为CPU内部有个nandflash控制器，所以CPU通过这个内部控制器来间接控制nandflash。

\*\*传地址(CPU要访问的nandflash地址)----------》nandflash

\*\*传命令(是读、写、还是擦除nandflash)---------》nandflash

\*\*传数据()----------------------------------》nandflash

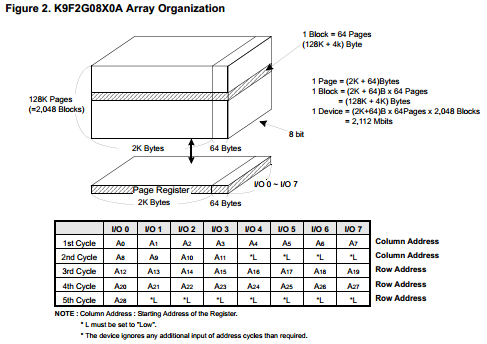
怎么来定位nandflash地址呢？也就是说nandflash地址构成？

参考:

开发板光盘:nand芯片手册: k9f2g08x0a\_rev13.pdf:

**Figure 2. K9F2G08X0A Array Organization**

比如:



行地址==页编号

列地址==页偏移

nandflash信号引脚

CLE(command latch enable):命令锁存允许

ALE(address latch enable):地址锁存允许

CE:芯片选择

RE:读允许

WE:写允许

WP:在写或擦除期间，提供写保护

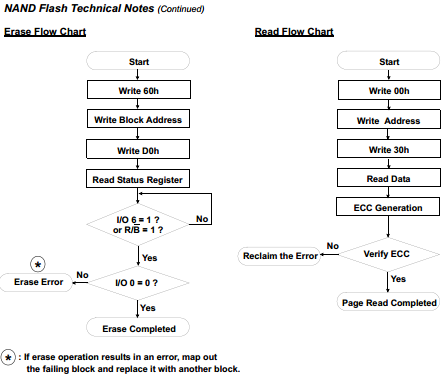
R/B:读/忙

nandflash驱动设计-读功能实现

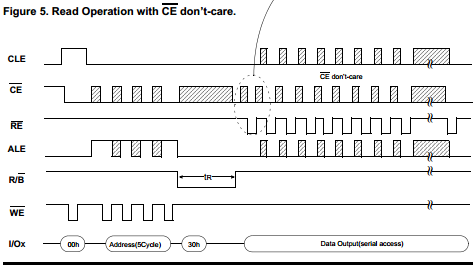
根据上面理论我们知道nandflash由很多页组成，而页又分为主数据区和空闲数据区，主数据区存放数据，空闲数据区存放校验码等信息，相当于页头部。

第一种读nand方式：页读(需要列地址,数据存放的buf)

参考手册: 开发板光盘:nand芯片手册: k9f2g08x0a\_rev13.pdf:



我们找到读的时序图，如下图：



**Figure 5. Read Operation with CE don’t-care.找到这个时序图**

**我们关注I/Ox这个pin脚上的信息:page\_read( )**

**\*\*选中nandflash芯片:使能nand片选信号:芯片手册**

**\*\*清除R/B信号**

**\*\*发送命令0x00**

**\*\*发送5个周期的地址:发送列2个列地址，发送3个行地址**

**\*\*发送0x30命令**

**\*\*等待R/B信号**

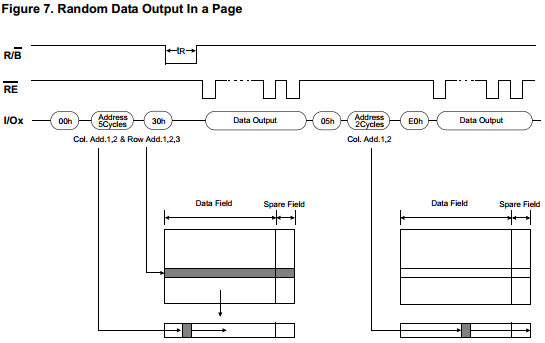
**\*\*读取数据(串口通道)**

**\*\*取消选中nandflash芯片**

第二种读nand方式：随机读（需要我们提供页地址(行地址)+列地址,存放数据的buf）

芯片手册导读

开发板光盘:nand芯片手册: k9f2g08x0a\_rev13.pdf:



## nandflash变硬盘-读操作mini2440

### start.S

**.text**

.global \_start

\_start**:**

b reset

ldr pc**,** \_undifined\_instruction

ldr pc**,** \_software\_interrupt

ldr pc**,** \_prefetch\_abort

ldr pc**,** \_data\_abort

ldr pc**,** \_not\_used

ldr pc**,** \_irq

ldr pc**,** \_fiq

\_undifined\_instruction**:** .word undifined\_instruction

\_software\_interrupt**:** .word software\_interrupt

\_prefetch\_abort**:** .word prefetch\_abort

\_data\_abort**:** .word data\_abort

\_not\_used**:** .word not\_used

\_irq**:** .word irq

\_fiq**:** .word reset

undifined\_instruction**:**

**nop**

software\_interrupt**:**

**nop**

prefetch\_abort**:**

**nop**

data\_abort**:**

**nop**

not\_used**:**

**nop**

irq**:**

**sub** lr**,** lr **,**#4

stmfd **sp**! **,** {r0**-**r12**,**lr} **/\***保存环境**\*/**

**bl** handle\_int

ldmfd **sp**!**,** {r0**-**r12**,** pc}**^/\***恢复环境**\*/**

fiq**:**

**nop**

reset**:**

**bl** set\_svc

**bl** disable\_watchdog

**bl** disable\_interrupt

**bl** disable\_mmu

**bl** init\_clock

**bl** init\_sdram

**bl** init\_stack

**bl** nandflash\_init

**bl** copy\_to\_ram

**bl** clean\_bss

ldr pc**,=**gboot\_main

@ **bl** light\_led

set\_svc**:**

mrs r0**,** cpsr

bic r0**,** r0**,**#0x1f

orr r0**,** r0**,**#0xd3

msr cpsr**,** r0

**mov** pc**,** lr

#define pWTCON 0x53000000

disable\_watchdog**:**

ldr r0**,=**pWTCON

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_interrupt**:**

mvn r1**,** #0x0

ldr r0**,=**0x4a000008

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_mmu**:**

mcr p15**,**0**,**r0**,**c7**,**c7**,**0

mrc p15**,**0**,**r0**,**c1**,**c0**,**0

bic r0**,** r0**,** #0x00000007

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

**mov** pc**,** lr

#define CLKDIVN 0x4c000014

#define MPLLCON 0x4c000004

#define MPLL\_405MHZ **((**127**<<**12**)|(**2**<<**4**)|(**1**<<**0**))**

init\_clock**:**

ldr r0**,=**CLKDIVN

**mov** r1**,** #0x5

**str** r1**,[**r0**]**

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

orr r0**,**r0**,**#0xc0000000

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

ldr r0**,=**MPLLCON

ldr r1**,=**MPLL\_405MHZ

**str** r1**,[**r0**]**

**mov** pc**,** lr

#define mem\_contrl 0x48000000

init\_sdram**:**

ldr r0**,=**mem\_contrl

**add** r3**,** r0**,** #4**\***13

adrl r1**,** mem\_data

0**:**

ldr r2**,[**r1**],** #4

**str** r2**,[**r0**],** #4

**cmp** r0**,** r3

bne 0b

**mov** pc**,** lr

copy\_to\_ram**:**

**mov** r0**,**#0x00

ldr r1**,=**\_start

ldr r2**,=**bss\_end

**sub** r2**,**r2**,**r1

**mov** ip**,**lr

**bl** nand\_to\_ram

**mov** lr**,**ip

**mov** pc**,**lr

init\_stack**:**

msr cpsr\_c**,** #0xd2

ldr **sp,=**0x33000000**//**初始化r13\_irq

msr cpsr\_c**,** #0xd3

ldr **sp,=**0x34000000**//**初始化R13\_svc

**mov** pc **,**lr

clean\_bss**:**

ldr r0**,=**bss\_start

ldr r1**,=**bss\_end

**cmp** r0**,** r1

moveq pc**,** lr

clean\_loop**:**

**mov** r2**,** #0

**str** r2**,[**r0**],** #4

**cmp** r0**,** r1

bne clean\_loop

**mov** pc**,** lr

mem\_data**:**

.long 0x22000000

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00018001

.long 0x00018001

.long 0x008c04f5

.long 0x000000b1

.long 0x00000030

.long 0x00000030

#define GPBCON 0x56000010

#define GPBDAT 0x56000014

light\_led**:**

ldr r0**,=**GPBCON

**mov** r1**,** #0x400

**str** r1**,[**r0**]**

ldr r0**,=**GPBDAT

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

**mov** pc**,** lr

### uart.c

#define GPHCON (\*(volatile unsigned long\*)0x56000070)

#define ULCON0 (\*(volatile unsigned long\*)0x50000000)

#define UCON0 (\*(volatile unsigned long\*)0x50000004)

#define UBRDIV0 (\*(volatile unsigned long\*)0x50000028)

#define UTRSTAT0 (\*(volatile unsigned long\*)0x50000010)

#define UTXH0 (\*(volatile unsigned long\*)0x50000020)

#define URXH0 (\*(volatile unsigned long\*)0x50000024)

#define PCLK 50000000

#define BAUD 115200

void uart\_init**()**

**{**

//1.配置引脚功能

GPHCON **&=~(**0xf**<<**4**);**

GPHCON **|=(**0xa**<<**4**);**

//2.1 设置数据格式

ULCON0 **=**0b11**;**

//2.2 设置工作模式

UCON0 **=**0b0101**;**

//3. 设置波特率

UBRDIV0 **=(**int**)(**PCLK**/(**BAUD**\***16**)-**1**);**

**}**

void putc**(**unsignedchar ch**)**

**{**

**while(!(**UTRSTAT0 **&(**1**<<**1**)));**

UTXH0 **=** ch**;**

**}**

unsignedchar getc**(**void**)**

**{**

unsignedchar ret**;**

**while(!(**UTRSTAT0 **&(**1**<<**0**)));**

// 取数据

ret **=** URXH0**;**

**if((**ret **==**0x0d**)||(**ret **==**0x0a**))**

**{**

putc**(**0x0d**);**

putc**(**0x0a**);**

**}**

**else**

putc**(**ret**);**

**return** ret**;**

**}**

### flash.c

#define NFCONF (\*(volatile unsigned long\*)0x4E000000)

#define NFCONT (\*(volatile unsigned long\*)0x4E000004)

#define NFCMD (\*(volatile unsigned char\*)0x4E000008)

#define NFADDR (\*(volatile unsigned char\*)0x4E00000C)

#define NFDATA (\*(volatile unsigned char\*)0x4E000010)

#define NFSTAT (\*(volatile unsigned char\*)0x4E000020)

#define TACLS 1

#define TWRPH0 2

#define TWRPH1 1

void select\_chip**()**

**{**

NFCONT **&=~(**1**<<**1**);**

**}**

void deselect\_chip**()**

**{**

NFCONT **|=(**1**<<**1**);**

**}**

void clear\_RnB**()**

**{**

NFSTAT **|=(**1**<<**2**);**

**}**

void send\_cmd**(**unsigned cmd**)**

**{**

NFCMD **=** cmd**;**

**}**

void send\_addr**(**unsigned addr**)**

**{**

NFADDR **=** addr**;**

**}**

void wait\_RnB**()**

**{**

**while(!(**NFSTAT**&(**1**<<**2**)))**

**{**

**;**

**}**

**}**

void nand\_reset**()**

**{**

//选中flash

select\_chip**();**

//清除RnB

clear\_RnB**();**

//发送0xff命令

send\_cmd**(**0xff**);**

//等待RnB

wait\_RnB**();**

//取消选中flash

deselect\_chip**();**

**}**

void nandflash\_init**()**

**{**

//初始化NFCONF

NFCONF **=(**TACLS**<<**12**)|(**TWRPH0**<<**8**)|(**TWRPH1**<<**4**);**

//初始化NFCONT

NFCONT **=(**1**<<**0**)|(**1**<<**1**);**

//复位

nand\_reset**();**

**}**

void NF\_PageRead**(**unsignedlong addr**,**unsignedchar**\*** buff**)**

**{**

int i**;**

//选中nandflash芯片

select\_chip**();**

//清除RnB

clear\_RnB**();**

//发送命令0x00

send\_cmd**(**0x00**);**

//发送列地址

send\_addr**(**0x00**);**

send\_addr**(**0x00**);**

//发送行地址

send\_addr**(**addr**&**0xff**);**

send\_addr**((**addr**>>**8**)&**0xff**);**

send\_addr**((**addr**>>**16**)&**0xff**);**

//发送命令0x30

send\_cmd**(**0x30**);**

//等待RnB

wait\_RnB**();**

//读取数据

**for(**i**=**0**;**i**<**2048**;**i**++)**

**{**

buff**[**i**]=** NFDATA**;**

**}**

//取消选中nandflash芯片

deselect\_chip**();**

**}**

void nand\_to\_ram**(**unsignedlong start\_addr**,**unsignedchar**\*** sdram\_addr**,**int size**)**

**{**

int i**;**

**for(** i**=(**start\_addr **>>**11**);** size**>**0**;)**

**{**

NF\_PageRead**(**i**,**sdram\_addr**);**

size **-=**2048**;**

sdram\_addr **+=**2048**;**

i**++;**

**}**

**}**

int NF\_Erase**(**unsignedlong addr**)**

**{**

int ret**;**

//选中flash芯片

select\_chip**();**

//清除RnB

clear\_RnB**();**

//发送命令0x60

send\_cmd**(**0x60**);**

//发送行地址

send\_addr**(**addr**&**0xff**);**

send\_addr**((**addr**>>**8**)&**0xff**);**

send\_addr**((**addr**>>**16**)&**0xff**);**

//发送命令D0

send\_cmd**(**0xD0**);**

//等待RnB

wait\_RnB**();**

//发送命令0x70

send\_cmd**(**0x70**);**

//读取擦除结果

ret **=** NFDATA**;**

//取消选中flash芯片

deselect\_chip**();**

**return** ret**;**

**}**

int NF\_WritePage**(**unsignedlong addr**,**unsignedchar**\***buff**)**

**{**

unsignedint i**,**ret **=**0**;**

//选中nandflash

select\_chip**();**

//清除RnB

clear\_RnB**();**

//发送0x80命令

send\_cmd**(**0x80**);**

//发送2个列地址

send\_addr**(**0x00**);**

send\_addr**(**0x00**);**

//发送3个行地址

send\_addr**(**addr**&**0xff**);**

send\_addr**((**addr**>>**8**)&**0xff**);**

send\_addr**((**addr**>>**16**)&**0xff**);**

//发送数据

**for(**i**=**0**;**i**<**2048**;**i**++)**

**{**

NFDATA **=** buff**[**i**];**

**}**

//发送0x10命令

send\_cmd**(**0x10**);**

//等待RnB

wait\_RnB**();**

//发送0x70命令

send\_cmd**(**0x70**);**

//读取写入结果

ret **=** NFDATA**;**

//关闭nandflash

deselect\_chip**();**

**return** ret**;**

**}**

### mmu.c

/\*

\* 用于段描述符的一些宏定义

\*/

#define MMU\_FULL\_ACCESS (3 << 10) /\* 访问权限 \*/

#define MMU\_DOMAIN (0 << 5) /\* 属于哪个域 \*/

#define MMU\_SPECIAL (1 << 4) /\* 必须是1 \*/

#define MMU\_CACHEABLE (1 << 3) /\* cacheable \*/

#define MMU\_BUFFERABLE (1 << 2) /\* bufferable \*/

#define MMU\_SECTION (2 << 0) /\* 段描述符 \*/

#define MMU\_SECDESC (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_SECTION)

#define MMU\_SECDESC\_WB (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_CACHEABLE | MMU\_BUFFERABLE | MMU\_SECTION)

void create\_page\_table**(**void**)**

**{**

unsignedlong**\***ttb **=(**unsignedlong**\*)**0x31000000**;**

unsignedlong vaddr**,** paddr**;**

vaddr **=**0x00000000**;**

paddr **=**0x30000000**;**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xfff00000**)|** MMU\_SECDESC\_WB**;**

vaddr **=**0x56000000**;**

paddr **=**0x56000000**;**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC**;**

vaddr **=**0x4A000000**;**

paddr **=**0x4A000000**;**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xfff00000**)|** MMU\_SECDESC**;**

vaddr **=**0x30000000**;**

paddr **=**0x30000000**;**

**while(**vaddr **<**0x34000000**)**

**{**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC\_WB**;**

vaddr **+=**0x100000**;**

paddr **+=**0x100000**;**

**}**

**}**

void mmu\_enable**()**

**{**

\_\_asm\_\_**(**

/\*设置TTB\*/

"ldr r0, =0x31000000\n"

"mcr p15, 0, r0, c2, c0, 0\n"

/\*不进行权限检查\*/

"mvn r0, #0\n"

"mcr p15, 0, r0, c3, c0, 0\n"

/\*使能MMU\*/

"mrc p15, 0, r0, c1, c0, 0\n"

"orr r0, r0, #0x0001\n"

"mcr p15, 0, r0, c1, c0, 0\n"

**:**

**:**

**);**

**}**

void mmu\_init**()**

**{**

create\_page\_table**();**

mmu\_enable**();**

**}**

### led.c

#define GPBCON (volatile unsigned long\*)0x56000010

#define GPBDAT (volatile unsigned long\*)0x56000014

void led\_init**()**

**{**

**\*(**GPBCON**)=**0x400**;**

**}**

void led\_off**()**

**{**

**\*(**GPBDAT**)=**0x7ff**;**

**}**

void led\_on**()**

**{**

**\*(**GPBDAT**)=**0x0**;**

**}**

### interrupt.c

/\*interrupt registes\*/

#define SRCPND (volatile unsigned long \*)0x4A000000

#define INTMOD (volatile unsigned long \*)0x4A000004

#define INTMSK (volatile unsigned long \*)0x4A000008

#define PRIORITY (volatile unsigned long \*)0x4A00000c

#define INTPND (volatile unsigned long \*)0x4A000010

#define INTOFFSET (volatile unsigned long \*)0x4A000014

#define SUBSRCPND (volatile unsigned long \*)0x4A000018

#define INTSUBMSK (volatile unsigned long \*)0x4A00001c

#define EINTMASK (volatile unsigned long \*)0x560000a4

#define EINTPEND (volatile unsigned long \*)0x560000a8

void init\_irq**()**

**{**

// 在EINTMASK寄存器中使能它们

**\*(**EINTMASK**)&=(~(**1**<<**8**))&(~(**1**<<**11**))&(~(**1**<<**13**))&(~(**1**<<**14**));**

// EINT8\_23使能

**\*(**INTMSK**)&=~(**1**<<**5**);**

\_\_asm\_\_**(**

/\*开中断\*/

"mrs r0,cpsr\n"

"bic r0, r0, #0x80\n"

"msr cpsr\_c, r0\n"

**:**

**:**

**);**

**}**

void handle\_int**()**

**{**

/\*读取产生中断的源\*/

unsignedlong value **=\*(**EINTPEND**)&((**1**<<**8**)|(**1**<<**11**)|(**1**<<**13**)|(**1**<<**14**));**

**switch(**value**)**

**{**

**case(**1**<<**8**):**//K1

led\_on**();**

**break;**

**case(**1**<<**11**):**//K2

led\_off**();**

**break;**

**case(**1**<<**13**):**//K3

led\_on**();**

**break;**

**case(**1**<<**14**):**//K4

led\_off**();**

**break;**

**default:**

**break;**

**}**

/\* 中断清除 \*/

**\*(**EINTPEND**)=** value**;**

**\*(**SRCPND**)=**1**<<**5**;**

**\*(**INTPND**)=**1**<<**5**;**

**}**

### button.c

#define GPGCON (volatile unsigned long \*)0x56000060

/\*

\* K1,K2,K3,K4对应GPG0、GPG3、GPG5、GPG6

\*/

#define GPG0\_int (0x2<<(0\*2))

#define GPG3\_int (0x2<<(3\*2))

#define GPG5\_int (0x2<<(5\*2))

#define GPG6\_int (0x2<<(6\*2))

#define GPG0\_msk (3<<(0\*2))

#define GPG3\_msk (3<<(3\*2))

#define GPG5\_msk (3<<(5\*2))

#define GPG6\_msk (3<<(6\*2))

void button\_init**()**

**{**

**\*(**GPGCON**)&=~(**GPG0\_msk **|** GPG3\_msk **|** GPG5\_msk **|** GPG6\_msk**);**

**\*(**GPGCON**)|=** GPG0\_int **|** GPG3\_int **|** GPG5\_int **|** GPG6\_int**;**

**}**

### gboot.lds

OUTPUT\_ARCH**(**arm**)**

ENTRY**(**\_start**)**

SECTIONS **{**

**.=**0x30008000**;**

**.=** ALIGN**(**4**);**

**.**text **:**

**{**

start.o **(.**text**)**

**\*(.**text**)**

**}**

**.=** ALIGN**(**4**);**

**.**data **:**

**{**

**\*(.**data**)**

**}**

**.=** ALIGN**(**4**);**

bss\_start **=.;**

**.**bss **:**

**{**

**\*(.**bss**)**

**}**

bss\_end **=.;**

**}**

### Makefile

all**:** start.o main.o mmu.o led.o interrupt.o button.o flash.o

arm-linux-ld -Tgboot.lds -o gboot.elf $^

arm-linux-objcopy -O binary gboot.elf gboot.bin

%.o**:** %.S

arm-linux-gcc -g -c $^

%.o**:** %.c

arm-linux-gcc -g -c $^

.PHONY**:** clean

clean**:**

rm \*.o \*.elf \*.bin

### main.c

int gboot\_main**()**

**{**

#ifdef MMU\_ON

mmu\_init**();**

#endif

led\_init**();**

button\_init**();**

init\_irq**();**

**while(**1**);**

**return**0**;**

**}**

## nandflash变硬盘-读操作OK6410

### start.S

**.text**

.global \_start

\_start**:**

b reset

ldr pc**,** \_undifined\_instruction

ldr pc**,** \_software\_interrupt

ldr pc**,** \_prefetch\_abort

ldr pc**,** \_data\_abort

ldr pc**,** \_not\_used

ldr pc**,** \_irq

ldr pc**,** \_fiq

\_undifined\_instruction**:** .word undifined\_instruction

\_software\_interrupt**:** .word software\_interrupt

\_prefetch\_abort**:** .word prefetch\_abort

\_data\_abort**:** .word data\_abort

\_not\_used**:** .word not\_used

\_irq**:** .word irq

\_fiq**:** .word reset

undifined\_instruction**:**

**nop**

software\_interrupt**:**

**nop**

prefetch\_abort**:**

**nop**

data\_abort**:**

**nop**

not\_used**:**

**nop**

irq**:**

**nop**

fiq**:**

**nop**

reset**:**

**bl** set\_svc

**bl** set\_peri\_port

**bl** disable\_watchdog

**bl** disable\_interrupt

**bl** disable\_mmu

**bl** init\_clock

**bl** mem\_init

**bl** init\_stack

**bl** nand\_init

**bl** copy\_to\_ram

**bl** clean\_bss

ldr pc**,=**gboot\_main

@ **bl** light\_led

set\_peri\_port**:**

ldr r0**,=**0x70000000

orr r0**,** r0**,** #0x13

mcr p15**,**0**,**r0**,**c15**,**c2**,**4

**mov** pc**,** lr

set\_svc**:**

mrs r0**,** cpsr

bic r0**,** r0**,**#0x1f

orr r0**,** r0**,**#0xd3

msr cpsr**,** r0

**mov** pc**,** lr

#define pWTCON 0x7e004000

disable\_watchdog**:**

ldr r0**,=**pWTCON

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_interrupt**:**

mvn r1**,**#0x0

ldr r0**,=**0x71200014

**str** r1**,[**r0**]**

ldr r0**,=**0x71300014

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_mmu**:**

mcr p15**,**0**,**r0**,**c7**,**c7**,**0

mrc p15**,**0**,**r0**,**c1**,**c0**,**0

bic r0**,** r0**,** #0x00000007

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

**mov** pc**,** lr

#define CLK\_DIV0 0x7e00f020

#define OTHERS 0x7e00f900

#define MPLL\_CON 0x7e00f010

#define APLL\_CON 0x7e00f00c

#define CLK\_SRC 0x7e00f01c

#define DIV\_VAL **((**0x0**<<**0**)|(**0x1**<<**9**)|(**0x1**<<**8**)|(**0x3**<<**12**))**

#define PLL\_VAL **((**1**<<**31**)|(**266**<<**16**)|(**3**<<**8**)|(**1**<<**0**))**

init\_clock**:**

ldr r0**,=**CLK\_DIV0

ldr r1**,=**DIV\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**OTHERS

ldr r1**,[**r0**]**

bic r1**,**r1**,**#0xc0

**str** r1**,[**r0**]**

ldr r0**,=**APLL\_CON

ldr r1**,=**PLL\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**MPLL\_CON

ldr r1**,=**PLL\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**CLK\_SRC

**mov** r1**,** #0x3

**str** r1**,[**r0**]**

**mov** pc**,** lr

copy\_to\_ram**:**

**mov** r0**,**#0

ldr r1**,=**\_start

ldr r2**,=**bss\_end

**sub** r2**,**r2**,**r1

**mov** ip**,**lr

**bl** nand\_to\_ram

**mov** lr**,**ip

**mov** pc**,**lr

init\_stack**:**

msr cpsr\_c**,** #0xd2

ldr **sp,=**0x53000000**//**初始化r13\_irq

msr cpsr\_c**,** #0xd3

ldr **sp,=**0x54000000**//**初始化R13\_svc

**mov** pc **,**lr

clean\_bss**:**

ldr r0**,=**bss\_start

ldr r1**,=**bss\_end

**cmp** r0**,** r1

moveq pc**,** lr

clean\_loop**:**

**mov** r2**,** #0

**str** r2**,[**r0**],** #4

**cmp** r0**,** r1

bne clean\_loop

**mov** pc**,** lr

light\_led**:**

ldr r1**,=**0x7F008820

ldr r0**,=**0x1111

**str** r0**,[**r1**]**

ldr r1**,=**0x7F008824

**mov** r0**,**#0xe

**str** r0**,[**r1**]**

**mov** pc**,** lr

### uart.c

#define ULCON0 (\*((volatile unsigned long \*)0x7F005000))

#define UCON0 (\*((volatile unsigned long \*)0x7F005004))

#define UFCON0 (\*((volatile unsigned long \*)0x7F005008))

#define UMCON0 (\*((volatile unsigned long \*)0x7F00500C))

#define UTRSTAT0 (\*((volatile unsigned long \*)0x7F005010))

#define UFSTAT0 (\*((volatile unsigned long \*)0x7F005018))

#define UTXH0 (\*((volatile unsigned char \*)0x7F005020))

#define URXH0 (\*((volatile unsigned char \*)0x7F005024))

#define UBRDIV0 (\*((volatile unsigned short \*)0x7F005028))

#define UDIVSLOT0 (\*((volatile unsigned short \*)0x7F00502C))

#define GPACON (\*((volatile unsigned long \*)0x7F008000))

#define PCLK 66500000

#define BOUD 115200

void uart\_init**(**void**)**

**{**

GPACON **&=~**0xff**;**

GPACON **|=**0x22**;**

/\* ULCON0 \*/

ULCON0 **=**0x3**;**/\* 数据位:8, 无较验, 停止位: 1, 8n1 \*/

UCON0 **=**0x5**;**/\* 使能UART发送、接收 \*/

UFCON0 **=**0x01**;**/\* FIFO ENABLE \*/

UMCON0 **=**0**;**

/\* 波特率 \*/

/\* DIV\_VAL = (PCLK / (bps x 16 ) ) - 1

\* bps = 115200

\* DIV\_VAL = (66000000 / (115200 x 16 ) ) - 1

\* = 35.08

\*/

UBRDIV0 **=(**int**)(**PCLK**/(**BOUD**\***16**)-**1**);**

/\* x/16 = 0.08

\* x = 1

\*/

UDIVSLOT0 **=**0x1**;**

**}**

void putc**(**unsignedchar c**)**

**{**

**while(**UFSTAT0 **&(**1**<<**14**));**/\* 如果TX FIFO满，等待 \*/

UTXH0 **=** c**;**/\* 写数据 \*/

**}**

unsignedchar getc**(**void**)**

**{**

unsignedchar ret**;**

**while((**UFSTAT0 **&**0x7f**)==**0**);**/\* 如果RX FIFO空，等待 \*/

ret **=** URXH0**;**/\* 取数据 \*/

**if((**ret **==**0x0d**)||(**ret **==**0x0a**))**

**{**

putc**(**0x0d**);**

putc**(**0x0a**);**

**}**

**else**

**{**

putc**(**ret**);**

**}**

**return** ret**;**

**}**

### flash.c

#define NFCONF (\*((volatile unsigned long\*)0x70200000))

#define NFCONT (\*((volatile unsigned long\*)0x70200004))

#define NFCMMD (\*((volatile unsigned char\*)0x70200008))

#define NFSTAT (\*((volatile unsigned char\*)0x70200028))

#define NFADDR (\*((volatile unsigned char\*)0x7020000c))

#define NFDATA (\*((volatile unsigned char\*)0x70200010))

void select\_ship**(**void**)**

**{**

NFCONT **&=~(**1**<<**1**);**

**}**

void delselect\_ship**(**void**)**

**{**

NFCONT **|=(**1**<<**1**);**

**}**

void clean\_RnB**()**

**{**

NFSTAT **|=(**1**<<**4**);**

**}**

void nand\_cmd**(**unsignedchar cmd**)**

**{**

NFCMMD **=** cmd**;**

**}**

void wait\_RnB**(**void**)**

**{**

**while(!(**NFSTAT **&**0x1**));**

**}**

void nand\_addr**(**unsignedchar addr**)**

**{**

NFADDR **=** addr**;**

**}**

void nand\_reset**(**void**)**

**{**

/\* 选中 \*/

select\_ship**();**

/\* 清除RnB \*/

clean\_RnB**();**

/\* 发出复位信号 \*/

nand\_cmd**(**0xff**);**

/\* 等待就绪 \*/

wait\_RnB**();**

/\* 取消选中 \*/

delselect\_ship**();**

**}**

void nand\_init**(**void**)**

**{**

/\*

HCLK的频率为100MHZ，周期就为10ns

TACLS > 0 ns

TWRPH0 > 15ns

TWRPH1 > 5ns

TACLS的值 = HCLK x TACLS > 0ns

TWRPH0的值 = HCLK x (TWRPH0 + 1) > 15ns

TWRPH1的值 = HCLK x (TWRPH1 +1) > 5ns

\*/

/\* 设置时间参数 \*/

#define TACLS 1

#define TWRPH0 2

#define TWRPH1 1

NFCONF **&=~((**7**<<**12**)|(**7**<<**8**)|(**7**<<**4**));**

NFCONF **|=(**TACLS**<<**12**)|(**TWRPH0**<<**8**)|(**TWRPH1**<<**4**);**

/\* 使能 nandflash controller\*/

NFCONT **=**1**|(**1**<<**1**);**

/\* 复位 \*/

nand\_reset**();**

**}**

void NF\_PageRead**(**unsignedlong addr**,**unsignedchar**\*** buff**)**

**{**

int i**;**

/\* 选中芯片 \*/

select\_ship**();**

/\* 清除RnB \*/

clean\_RnB**();**

/\* 发出命令0x00 \*/

nand\_cmd**(**0x00**);**

/\* 发出列地址 \*/

nand\_addr**(**0x00**);**

nand\_addr**(**0x00**);**

/\* 发出行地址 \*/

nand\_addr**(**addr**&**0xff**);**

nand\_addr**((**addr **>>**8**)&(**0xff**));**

nand\_addr**((**addr **>>**16**)&(**0xff**));**

/\* 发出命令0x30 \*/

nand\_cmd**(**0x30**);**

/\* 等待就绪 \*/

wait\_RnB**();**

/\* 读数据 \*/

**for(**i **=**0**;** i**<**1024**\***4**;** i**++)**

**{**

buff**[**i**]=** NFDATA**;**

**}**

/\* 取消片选 \*/

delselect\_ship**();**

**}**

void nand\_to\_ram**(**unsignedlong start\_addr**,**unsignedchar**\*** sdram\_addr**,**int size**)**

**{**

/\* i为页号、sdram\_addr为内存中的位置、size拷贝数据的大小 \*/

int i**;**

/\*

S3C6410启动时拷贝的8K代码不是存储在Nand flash的第一页上，

而是存储在Nand flash的前4页上，每页2K，总共8K，

\*/

**for(**i **=**0**;** i **<**4**;** i**++,** sdram\_addr**+=**2048**)**

**{**

NF\_PageRead**(**i**,**sdram\_addr**);**

**}**

size **-=**1024**\***8**;**

**for(** i**=**4**;** size**>**0**;)**

**{**

NF\_PageRead**(**i**,**sdram\_addr**);**

size **-=**4096**;**

sdram\_addr **+=**4096**;**

i**++;**

**}**

**}**

### mmu.c

/\*

\* 用于段描述符的一些宏定义

\*/

#define MMU\_FULL\_ACCESS (3 << 10) /\* 访问权限 \*/

#define MMU\_DOMAIN (0 << 5) /\* 属于哪个域 \*/

#define MMU\_SPECIAL (1 << 4) /\* 必须是1 \*/

#define MMU\_CACHEABLE (1 << 3) /\* cacheable \*/

#define MMU\_BUFFERABLE (1 << 2) /\* bufferable \*/

#define MMU\_SECTION (2) /\* 表示这是段描述符 \*/

#define MMU\_SECDESC (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_SECTION)

#define MMU\_SECDESC\_WB (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_CACHEABLE | MMU\_BUFFERABLE | MMU\_SECTION)

void create\_page\_table**(**void**)**

**{**

unsignedlong**\***ttb **=(**unsignedlong**\*)**0x50000000**;**

unsignedlong vaddr**,** paddr**;**

vaddr **=**0xA0000000**;**

paddr **=**0x7f000000**;**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC**;**

vaddr **=**0x50000000**;**

paddr **=**0x50000000**;**

**while(**vaddr **<**0x54000000**)**

**{**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC\_WB**;**

vaddr **+=**0x100000**;**

paddr **+=**0x100000**;**

**}**

**}**

void mmu\_enable**()**

**{**

\_\_asm\_\_**(**

/\*设置TTB\*/

"ldr r0, =0x50000000\n"

"mcr p15, 0, r0, c2, c0, 0\n"

/\*不进行权限检查\*/

"mvn r0, #0\n"

"mcr p15, 0, r0, c3, c0, 0\n"

/\*使能MMU\*/

"mrc p15, 0, r0, c1, c0, 0\n"

"orr r0, r0, #0x0001\n"

"mcr p15, 0, r0, c1, c0, 0\n"

**:**

**:**

**);**

**}**

void mmu\_init**()**

**{**

create\_page\_table**();**

mmu\_enable**();**

**}**

### mem.S

**.text**

.global mem\_init

mem\_init**:**

ldr r0**,=**0x7e00f120

**mov** r1**,** #0x8

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001004 @内存控制命令寄存器

**mov** r1**,** #0x4 @根据手册得知需要先进入配置模式

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001010 @刷新寄存器地址

ldr r1**,=((**7800**/(**1000000000**/**133000000**)+**1**))** @设置刷新时间

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001014 @CAS latency寄存器

**mov** r1**,** #**(**3**<<**1**)**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001018 @t\_DQSS寄存器

**mov** r1**,** #0x1

**str** r1**,[**r0**]**

ldr r0**,=**0x7e00101c @T\_MRD寄存器

**mov** r1**,** #0x2

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001020 @t\_RAS寄存器

ldr r1**,=((**45**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001024 @t\_RC寄存器

ldr r1**,=((**68**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001028 @t\_RCD寄存器

ldr r1**,=((**23**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e00102c @t\_RFC寄存器

ldr r1**,=((**80**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001030 @t\_RP寄存器

ldr r1**,=((**23**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001034 @t\_rrd寄存器

ldr r1**,=((**15**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001038 @t\_wr寄存器

ldr r1**,=((**15**/(**1000000000**/**133000000**)+**1**))**

@ ldr r2**,[**r0**]**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e00103c @t\_wtr寄存器

**mov** r1**,** #0x07

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001040 @t\_xp寄存器

**mov** r1**,** #0x02

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001044 @t\_xsr寄存器

ldr r1**,=((**120**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001048 @t\_esr寄存器

ldr r1**,=((**120**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e00100c @内存控制配置寄存器

ldr r1**,=**0x00010012 @配置控制器

**str** r1**,[**r0**]**

ldr r0**,=**0x7e00104c @32位DRAM配置控制寄存器

ldr r1**,=**0x0b45

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001200 @片选寄存器

ldr r1**,=**0x150f8

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001304 @用户配置寄存器

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001008

ldr r1**,=**0x000c0000

**str** r1**,[**r0**]**

ldr r1**,=**0x00000000

**str** r1**,[**r0**]**

ldr r1**,=**0x00040000

**str** r1**,[**r0**]**

ldr r1**,=**0x000a0000

**str** r1**,[**r0**]**

ldr r1**,=**0x00080032

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001004

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

check\_dmc1\_ready**:**

ldr r0**,=**0x7e001000

ldr r1**,[**r0**]**

**mov** r2**,** #0x3

**and** r1**,** r1**,** r2

**cmp** r1**,** #0x1

bne check\_dmc1\_ready

**nop**

**mov** pc**,** lr

### led.c

#define GPMCON (volatile unsigned long\*)0x7f008820

#define GPMDAT (volatile unsigned long\*)0x7f008824

void led\_init**()**

**{**

**\*(**GPMCON**)=**0x1111**;**

**}**

void led\_on**()**

**{**

**\*(**GPMDAT**)=**0x0e**;**

**}**

void led\_off**()**

**{**

**\*(**GPMDAT**)=**0x0f**;**

**}**

### interrupt.c

/\*interrupt registes\*/

#define EXT\_INT\_0\_CON \*((volatile unsigned int \*)0x7f008900)

#define EXT\_INT\_0\_MASK \*((volatile unsigned int \*)0x7f008920)

#define EXT\_INT\_0\_PEND \*((volatile unsigned int \*)0x7f008924)

#define VIC0INTENABLE \*((volatile unsigned int \*)0x71200010)

#define EINT0\_VECTADDR \*((volatile unsigned int \*)0x71200100)

#define EINT5\_VECTADDR \*((volatile unsigned int \*)0x71200104)

#define VIC0ADDRESS \*((volatile unsigned int \*)0x71200f00)

#define VIC1ADDRESS \*((volatile unsigned int \*)0x71300f00)

void key1\_handle**()**

**{**

\_\_asm\_\_**(**

"sub lr, lr, #4\n"

"stmfd sp!, {r0-r12, lr}\n"

**:**

**:**

**);**

led\_on**();**

/\* 清除中断 \*/

EXT\_INT\_0\_PEND **=~**0x0**;**

VIC0ADDRESS **=**0**;**

VIC1ADDRESS **=**0**;**

\_\_asm\_\_**(**

"ldmfd sp!, {r0-r12, pc}^ \n"

**:**

**:**

**);**

**}**

void key6\_handle**()**

**{**

\_\_asm\_\_**(**

"sub lr, lr, #4\n"

"stmfd sp!, {r0-r12, lr}\n"

**:**

**:**

**);**

led\_off**();**

/\* 清除中断 \*/

EXT\_INT\_0\_PEND **=~**0x0**;**

VIC0ADDRESS **=**0**;**

VIC1ADDRESS **=**0**;**

\_\_asm\_\_**(**

"ldmfd sp!, {r0-r12, pc}^ \n"

**:**

**:**

**);**

**}**

void init\_irq**()**

**{**

EXT\_INT\_0\_CON **=(**0b010**)|(**0b010**<<**8**);**/\* 配置为下降沿触发 \*/

EXT\_INT\_0\_MASK **=**0**;**/\* 取消屏蔽外部中断 \*/

VIC0INTENABLE **|=(**0b1**)|(**0b10**);**/\* 使能外部中断\*/

EINT0\_VECTADDR **=(**int**)**key1\_handle**;**/\* 用户按下key时，CPU就会自动的将VIC0VECTADDR0的值赋给VIC0ADDRESS并跳转到这个地址去执 \*/

EINT5\_VECTADDR **=(**int**)**key6\_handle**;**

\_\_asm\_\_**(**

"mrc p15,0,r0,c1,c0,0\n"

"orr r0,r0,#(1<<24)\n"

"mcr p15,0,r0,c1,c0,0\n"

"mrs r0,cpsr\n"

"bic r0, r0, #0x80\n"

"msr cpsr\_c, r0\n"

**:**

**:**

**);**

**}**

### gboot.lds

OUTPUT\_ARCH**(**arm**)**

ENTRY**(**\_start**)**

SECTIONS **{**

**.=**0x50008000**;**

**.=** ALIGN**(**4**);**

**.**text **:**

**{**

start.o **(.**text**)**

**\*(.**text**)**

**}**

**.=** ALIGN**(**4**);**

**.**data **:**

**{**

**\*(.**data**)**

**}**

**.=** ALIGN**(**4**);**

bss\_start **=.;**

**.**bss **:**

**{**

**\*(.**bss**)**

**}**

bss\_end **=.;**

**}**

### button.c

#define GPNCON (volatile unsigned long\*)0x7f008830

void button\_init**()**

**{**

**\*(**GPNCON**)=**0b10**|(**0b10**<<**10**);**//设置按键1与按键6

**}**

### Makefile

all**:** start.o main.o mmu.o led.o button.o interrupt.o mem.o flash.o uart.o

arm-linux-ld -Tgboot.lds -o gboot.elf $^

arm-linux-objcopy -O binary gboot.elf gboot.bin

%.o**:** %.S

arm-linux-gcc -g -c $^

%.o**:** %.c

arm-linux-gcc -g -fno-builtin -c $^

.PHONY**:** clean

clean**:**

rm \*.o \*.elf \*.bin

### main.c

int gboot\_main**()**

**{**

#ifdef MMU\_ON

mmu\_init**();**

#endif

uart\_init**();**

led\_init**();**

button\_init**();**

init\_irq**();**

led\_on**();**

**while(**1**);**

**return**0**;**

**}**

## nandflash变硬盘-读操作Smart210

### start.S

@**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

@File**:**start.S

@

@成都国嵌嵌入式培训中心

@**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**.text**

.global \_start

\_start**:**

b reset

ldr pc**,** \_undefined\_instruction

ldr pc**,** \_software\_interrupt

ldr pc**,** \_prefetch\_abort

ldr pc**,** \_data\_abort

ldr pc**,** \_not\_used

ldr pc**,** \_irq

ldr pc**,** \_fiq

\_undefined\_instruction**:** .word undefined\_instruction

\_software\_interrupt**:** .word software\_interrupt

\_prefetch\_abort**:** .word prefetch\_abort

\_data\_abort**:** .word data\_abort

\_not\_used**:** .word not\_used

\_irq**:** .word irq

\_fiq**:** .word fiq

undefined\_instruction**:**

**nop**

software\_interrupt**:**

**nop**

prefetch\_abort**:**

**nop**

data\_abort**:**

**nop**

not\_used**:**

**nop**

irq**:**

**nop**

fiq**:**

**nop**

reset**:**

**bl** set\_svc

**bl** disable\_watchdog

**bl** disable\_interrupt

**bl** disable\_mmu

**bl** init\_clock

**bl** mem\_init

**bl** init\_stack

**bl** nand\_init

**bl** copy\_to\_ram

**bl** clean\_bss

ldr pc**,=**gboot\_main

@ **bl** light\_led

set\_svc**:**

mrs r0**,** cpsr

bic r0**,** r0**,** #0x1f

orr r0**,** r0**,** #0xd3

msr cpsr**,** r0

**mov** pc**,** lr

#define pWTCON 0xE2700000

disable\_watchdog**:**

ldr r0**,=**pWTCON

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_interrupt**:**

mvn r1**,**#0x0

ldr r0**,=**0xf2000014

**str** r1**,[**r0**]**

ldr r0**,=**0xf2100014

**str** r1**,[**r0**]**

ldr r0**,=**0xf2200014

**str** r1**,[**r0**]**

ldr r0**,=**0xf2300014

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_mmu**:**

mcr p15**,**0**,**r0**,**c7**,**c5**,**0

mcr p15**,**0**,**r0**,**c7**,**c6**,**1

mrc p15**,**0**,**r0**,**c1**,**c0**,**0

bic r0**,** r0**,** #0x00000007

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

**mov** pc**,** lr

#define CLK\_DIV0 0xe0100300

#define MPLL\_CON 0xe0100108

#define APLL\_CON 0xe0100100

#define CLK\_SRC 0xe0100200

#define DIV\_VAL **((**0x0**<<**0**)|(**4**<<**4**)|(**4**<<**8**)|(**1**<<**12**)|(**3**<<**16**)|(**1**<<**20**)|(**4**<<**24**)|(**1**<<**28**))**

#define APLL\_VAL **((**1**<<**31**)|(**3**<<**8**)|(**125**<<**16**)|(**1**<<**0**))**

#define MPLL\_VAL **((**1**<<**31**)|(**12**<<**8**)|(**667**<<**16**)|(**1**<<**0**))**

init\_clock**:**

ldr r0**,=**CLK\_DIV0

ldr r1**,=**DIV\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**APLL\_CON

ldr r1**,=**APLL\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**MPLL\_CON

ldr r1**,=**MPLL\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**CLK\_SRC

ldr r1**,=**0x1111

**str** r1**,[**r0**]**

**mov** pc**,** lr

copy\_to\_ram**:**

**mov** r0**,**#0

ldr r1**,=**\_start

ldr r2**,=**bss\_end

**sub** r2**,**r2**,**r1

**sub** r1**,**r1**,**#0x10

**mov** ip**,**lr

**bl** nand\_to\_ram

**mov** lr**,**ip

**mov** pc**,**lr

copy\_loop**:**

ldr r2**,[**r0**],** #4

**str** r2**,[**r1**],** #4

**cmp** r0**,** r3

bne copy\_loop

**mov** pc**,** lr

init\_stack**:**

msr cpsr\_c**,** #0xd2

ldr **sp,=**0x23000000 @此处实际设置的是r13\_irq

msr cpsr\_c**,** #0xd3

ldr **sp,=**0x24000000 @此处实际设置的是r13\_svc

**mov** pc**,** lr

clean\_bss**:**

ldr r0**,=**bss\_start

ldr r1**,=**bss\_end

**cmp** r0**,** r1

moveq pc**,** lr

clean\_loop**:**

**mov** r2**,** #0

**str** r2**,[**r0**],** #4

**cmp** r0**,** r1

bne clean\_loop

**mov** pc**,** lr

light\_led**:**

ldr R0**,=**0xE0200280

**mov** R1**,**#0x1

**str** R1**,[**R0**]**

ldr R0**,=**0xE0200284

**mov** R1**,**#0x0

**str** R1**,[**R0**]**

**mov** pc**,** lr

### nand.c

#define NFCONF (\*((volatile unsigned long\*)0xB0E00000))

#define NFCONT (\*((volatile unsigned long\*)0xB0E00004))

#define NFCMMD (\*((volatile unsigned char\*)0xB0E00008))

#define NFSTAT (\*((volatile unsigned char\*)0xB0E00028))

#define NFADDR (\*((volatile unsigned char\*)0xB0E0000C))

#define NFDATA (\*((volatile unsigned char\*)0xB0E00010))

#define MP0\_1CON (\*((volatile unsigned long \*)0xE02002E0))

#define MP0\_3CON (\*((volatile unsigned long \*)0xE0200320))

#define MP0\_1PUD (\*((volatile unsigned long \*)0xE02002E8))

#define MP0\_3PUD (\*((volatile unsigned long \*)0xE0200328))

#define page\_size 2048

/\* 设置时间参数 \*/

#define TACLS 7

#define TWRPH0 7

#define TWRPH1 7

//发片选

void select\_chip**(**void**)**

**{**

NFCONT **&=~(**1**<<**1**);**

**}**

//取消片选

void deselect\_chip**(**void**)**

**{**

NFCONT **|=(**1**<<**1**);**

**}**

//清除RnB

void clean\_RnB**()**

**{**

NFSTAT **|=(**1**<<**4**);**

**}**

//发命令

void send\_cmd**(**unsignedchar cmd**)**

**{**

NFCMMD **=** cmd**;**

**}**

//等待RnB就绪

void wait\_RnB**(**void**)**

**{**

**while(!(**NFSTAT **&(**1**<<**4**)));**

**}**

//发地址

void send\_addr**(**unsignedchar addr**)**

**{**

NFADDR **=**0**;**

NFADDR **=**0**;**

NFADDR **=** addr **&**0xff**;**

NFADDR **=(**addr **>>**8**)&**0xff**;**

NFADDR **=(**addr **>>**16**)&**0xff**;**

**}**

//复位

void nand\_reset**(**void**)**

**{**

/\* 选中 \*/

select\_chip**();**

/\* 清除RnB \*/

clean\_RnB**();**

/\* 发出复位信号 \*/

send\_cmd**(**0xff**);**

/\* 等待就绪 \*/

wait\_RnB**();**

/\* 取消选中 \*/

deselect\_chip**();**

**}**

//nandflash初始化

void nand\_init**(**void**)**

**{**

// 配置引脚

MP0\_1CON **&=~(**0xf**<<**8**);**

MP0\_1CON **|=(**0x3**<<**8**);**

MP0\_1PUD **&=~(**0x3**<<**4**);**

MP0\_3CON **&=~**0xFFFFFF**;**

MP0\_3CON **|=**0x22222222**;**

MP0\_3PUD **&=~(**0x3fff**);**

NFCONF **&=~(**0x777F**);**

NFCONF **|=(**TACLS**<<**12**)|(**TWRPH0**<<**8**)|(**TWRPH1**<<**4**)|(**0**<<**3**)|(**0**<<**2**)|(**1**<<**1**)|(**0**<<**0**);**

NFCONT **&=~(**0x707C7**);**

NFCONT **|=(**0x1**<<**23**)|(**0x1**<<**22**)|(**0**<<**18**)|(**0**<<**17**)|(**0**<<**16**)|(**0**<<**10**)|(**0**<<**9**)|(**0**<<**8**)|(**0**<<**7**)|(**0**<<**6**)|(**0x2**<<**1**)|(**1**<<**0**);**

NFCONF **|=**0x70**|**0x7700**;**

NFCONT **|=**0x03**;**

/\* 复位 \*/

nand\_reset**();**

**}**

void nand\_read\_page**(**unsignedlong addr **,**unsignedchar**\*** buff**)**

**{**

int i**;**

/\* 选中芯片 \*/

select\_chip**();**

/\* 清除RnB \*/

clean\_RnB**();**

/\* 发出命令0x00 \*/

send\_cmd**(**0x00**);**

/\* 发出地址 \*/

send\_addr**(**addr**);**

/\* 发出命令0x30 \*/

send\_cmd**(**0x30**);**

/\* 等待就绪 \*/

wait\_RnB**();**

/\* 读数据 \*/

**for(**i **=**0**;** i**<**page\_size**;)**

**{**

**\***buff**++=** NFDATA**;**

i**++;**

**}**

/\* 取消片选 \*/

deselect\_chip**();**

**}**

void nand\_to\_ram**(**unsignedlong start\_addr**,**unsignedchar**\*** sdram\_addr**,**int size**)**

**{**

int i**;**

unsignedint page\_shift **=**11**;**

**for(**i **=(**start\_addr**>>**page\_shift**);** size **>**0**;)**

**{**

nand\_read\_page**(**i **,** sdram\_addr**);**

size **-=** page\_size**;**

sdram\_addr **+=** page\_size**;**

i**++;**

**}**

**}**

void NF\_Erase**(**unsignedlong addr**)**

**{**

int ret**;**

//选中flash芯片

select\_chip**();**

//清除RnB

clean\_RnB**();**

//发送命令0x60

send\_cmd**(**0x60**);**

//发送行地址

send\_addr**(**addr**&**0xff**);**

send\_addr**((**addr**>>**8**)&**0xff**);**

send\_addr**((**addr**>>**16**)&**0xff**);**

//发送命令D0

send\_cmd**(**0xD0**);**

//等待RnB

wait\_RnB**();**

//发送命令0x70

send\_cmd**(**0x70**);**

//读取擦除结果

ret **=** NFDATA**;**

//取消选中flash芯片

deselect\_chip**();**

**return** ret**;**

**}**

int NF\_WritePage**(**unsignedlong addr**,**unsignedchar**\*** buff**)**

**{**

int ret**;**

int i**;**

//选中flash芯片

select\_chip**();**

//清除RnB

clean\_RnB**();**

//发送命令0x80

send\_cmd**(**0x80**);**

//发送列地址

send\_addr**(**0x00**);**

send\_addr**(**0x00**);**

//发送行地址

send\_addr**(**addr**&**0xff**);**

send\_addr**((**addr**>>**8**)&**0xff**);**

send\_addr**((**addr**>>**16**)&**0xff**);**

//写入数据

**for(**i**=**0**;**i**<**2048**;**i**++)**

**{**

NFDATA **=** buff**[**i**];**

**}**

//发送命令0x10

send\_cmd**(**0x10**);**

//等待RnB

wait\_RnB**();**

//发送命令0x70

send\_cmd**(**0x70**);**

//读取写入结果

ret **=** NFDATA**;**

//取消选中flash芯片

deselect\_chip**();**

**return** ret**;**

**}**

### mmu.c

/\*

\* 用于段描述符的一些宏定义

\*/

#define MMU\_FULL\_ACCESS (3 << 10) /\* 访问权限 \*/

#define MMU\_DOMAIN (0 << 5) /\* 属于哪个域 \*/

#define MMU\_SPECIAL (1 << 4) /\* 必须是1 \*/

#define MMU\_CACHEABLE (1 << 3) /\* cacheable \*/

#define MMU\_BUFFERABLE (1 << 2) /\* bufferable \*/

#define MMU\_SECTION (2) /\* 表示这是段描述符 \*/

#define MMU\_SECDESC (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_SECTION)

#define MMU\_SECDESC\_WB (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_CACHEABLE | MMU\_BUFFERABLE | MMU\_SECTION)

void create\_page\_table**(**void**)**

**{**

unsignedlong**\***ttb **=(**unsignedlong**\*)**0x20000000**;**

unsignedlong vaddr**,** paddr**;**

vaddr **=**0xA0000000**;**

paddr **=**0xE0200000**;**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC**;**

vaddr **=**0x20000000**;**

paddr **=**0x20000000**;**

**while(**vaddr **<**0x24000000**)**

**{**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC\_WB**;**

vaddr **+=**0x100000**;**

paddr **+=**0x100000**;**

**}**

**}**

void mmu\_enable**()**

**{**

\_\_asm\_\_**(**

/\*设置TTB\*/

"ldr r0, =0x20000000\n"

"mcr p15, 0, r0, c2, c0, 0\n"

/\*不进行权限检查\*/

"mvn r0, #0\n"

"mcr p15, 0, r0, c3, c0, 0\n"

/\*使能MMU\*/

"mrc p15, 0, r0, c1, c0, 0\n"

"orr r0, r0, #0x0001\n"

"mcr p15, 0, r0, c1, c0, 0\n"

**:**

**:**

**);**

**}**

void mmu\_init**()**

**{**

create\_page\_table**();**

mmu\_enable**();**

**}**

### mem.S

#define DMC\_PHYCONTROL0 0xf0000018

#define DMC\_PHYCONTROL1 0xf000001c

#define DMC\_CONCONTROL 0xf0000000

#define DMC\_MEMCONTROL 0xf0000004

#define DMC\_MEMCONFIG0 0xf0000008

#define DMC\_MEMCONFIG1 0xf000000c

#define DMC\_PRECHCONFIG 0xf0000014

#define DMC\_TIMINGAREF 0xf0000030

#define DMC\_TIMINGROW 0xf0000034

#define DMC\_TIMINGDATA 0xf0000038

#define DMC\_TIMINGPOWER 0xf000003c

#define DMC\_PHYSTATUS 0xf0000040

#define DMC\_DIRECTCMD 0xf0000010

#define DMC\_PWRDNCONFIG 0xf0000028

#define DMC0\_MEMCONTROL 0x00202400

#define DMC0\_MEMCONFIG\_0 0x20F00313

#define DMC0\_MEMCONFIG\_1 0x00F00313

#define DMC0\_TIMINGA\_REF 0x00000618

#define DMC0\_TIMING\_ROW 0x2B34438A

#define DMC0\_TIMING\_DATA 0x24240000

#define DMC0\_TIMING\_PWR 0x0BDC0343

.globl mem\_init

mem\_init**:**

@ step 2.1

ldr r0**,=**DMC\_PHYCONTROL0

ldr r1**,=**0x00101000

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_PHYCONTROL1

ldr r1**,=**0x00000086

**str** r1**,[**r0**]**

@ step 2.2

ldr r0**,=**DMC\_PHYCONTROL0

ldr r1**,=**0x00101002

**str** r1**,[**r0**]**

@ step 4

ldr r0**,=**DMC\_PHYCONTROL0

ldr r1**,=**0x00101003

**str** r1**,[**r0**]**

@ step 5

ldr r0**,=**DMC\_CONCONTROL

ldr r1**,=**0x0FFF1350

**str** r1**,[**r0**]**

@ step 6

ldr r0**,=**DMC\_MEMCONTROL

ldr r1**,=**DMC0\_MEMCONTROL

**str** r1**,[**r0**]**

@ step 7

ldr r0**,=**DMC\_MEMCONFIG0

ldr r1**,=**DMC0\_MEMCONFIG\_0

**str** r1**,[**r0**]**

@ step 8

ldr r0**,=**DMC\_PRECHCONFIG

ldr r1**,=**0xFF000000

**str** r1**,[**r0**]**

@ step 9.1

ldr r0**,=**DMC\_TIMINGAREF

ldr r1**,=**DMC0\_TIMINGA\_REF

**str** r1**,[**r0**]**

@ step 9.2

ldr r0**,=**DMC\_TIMINGROW

ldr r1**,=**DMC0\_TIMING\_ROW

**str** r1**,[**r0**]**

@ step 9.3

ldr r0**,=**DMC\_TIMINGDATA

ldr r1**,=**DMC0\_TIMING\_DATA

**str** r1**,[**r0**]**

@ step 9.4

ldr r0**,=**DMC\_TIMINGPOWER

ldr r1**,=**DMC0\_TIMING\_PWR

**str** r1**,[**r0**]**

@ step 11

wait\_lock**:**

ldr r0**,=**DMC\_PHYSTATUS

ldr r1**,[**r0**]**

**and** r2**,** r1**,** #0x4

**cmp** r2**,** #0x4

bne wait\_lock

@ step 14

ldr r0**,=**DMC\_DIRECTCMD

ldr r1**,=**0x07000000

**str** r1**,[**r0**]**

@ step 16

ldr r1**,=**0x01000000

**str** r1**,[**r0**]**

@ step 17

ldr r1**,=**0x00020000

**str** r1**,[**r0**]**

@ step 18

ldr r1**,=**0x00030000

**str** r1**,[**r0**]**

@ step 19

ldr r1**,=**0x00010400

**str** r1**,[**r0**]**

@ step 20

ldr r1**,=**0x00000542

**str** r1**,[**r0**]**

@ step 21

ldr r1**,=**0x01000000

**str** r1**,[**r0**]**

@ step 22.1

ldr r1**,=**0x05000000

**str** r1**,[**r0**]**

@ step 22.2

ldr r1**,=**0x05000000

**str** r1**,[**r0**]**

@ step 23

ldr r1**,=**0x00000442

**str** r1**,[**r0**]**

@ step 25.1

ldr r1**,=**0x00010780

**str** r1**,[**r0**]**

@ step 25.2

ldr r1**,=**0x00010400

**str** r1**,[**r0**]**

@ step 26**,**repeat step14**~**step25

ldr r1**,=**0x07100000

**str** r1**,[**r0**]**

ldr r1**,=**0x01100000

**str** r1**,[**r0**]**

ldr r1**,=**0x00120000

**str** r1**,[**r0**]**

ldr r1**,=**0x00130000

**str** r1**,[**r0**]**

ldr r1**,=**0x00110400

**str** r1**,[**r0**]**

ldr r1**,=**0x00100542

**str** r1**,[**r0**]**

ldr r1**,=**0x01100000

**str** r1**,[**r0**]**

ldr r1**,=**0x05100000

**str** r1**,[**r0**]**

ldr r1**,=**0x05100000

**str** r1**,[**r0**]**

ldr r1**,=**0x00100442

**str** r1**,[**r0**]**

ldr r1**,=**0x00110780

**str** r1**,[**r0**]**

ldr r1**,=**0x00110400

**str** r1**,[**r0**]**

@ step 27

ldr r0**,=**DMC\_CONCONTROL

ldr r1**,=**0x0FF02030

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_PWRDNCONFIG

ldr r1**,=**0xFFFF00FF

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_CONCONTROL

ldr r1**,=**0x00202400

**str** r1**,[**r0**]**

**mov** pc**,** lr

### led.c

#define GPC0CON \*((volatile unsigned long\*)0xE0200280)

#define GPC0DAT \*((volatile unsigned long\*)0xE0200284)

void led\_init**()**

**{**

GPC0CON **=**0x1**;**

**}**

void led\_off**()**

**{**

GPC0DAT **=**0x1**;**

**}**

void led\_on**()**

**{**

GPC0DAT **=**0x0**;**

**}**

### interrupt.c

/\*interrupt registes\*/

#define EXT\_INT\_2\_CON \*((volatile unsigned int \*)0xE0200E08)

#define EXT\_INT\_2\_MASK \*((volatile unsigned int \*)0xE0200F08)

#define VIC0INTENABLE \*((volatile unsigned int \*)0xF2000010)

#define key\_VICADDR \*((volatile unsigned int \*)0xF2000140)

#define VIC0ADDRESS \*((volatile unsigned int \*)0xF2000F00)

#define EXT\_INT\_2\_PEND \*((volatile unsigned int \*)0xE0200F48)

void key\_isr**()**

**{**

volatileunsignedint key\_code**;**

\_\_asm\_\_**(**

"sub lr, lr, #4\n"

"stmfd sp!, {r0-r12, lr}\n"

**:**

**:**

**);**

key\_code **=** EXT\_INT\_2\_PEND **&**0b11**;**

**switch(**key\_code**)**

**{**

**case**1**:**//K1

led\_on**();**

**break;**

**case**2**:**//K2

led\_off**();**

**break;**

**default:**

**break;**

**}**

/\* 清除中断 \*/

EXT\_INT\_2\_PEND **=~**0x0**;**

VIC0ADDRESS **=**0**;**

\_\_asm\_\_**(**

"ldmfd sp!, {r0-r12, pc}^ \n"

**:**

**:**

**);**

**}**

void init\_irq**()**

**{**

EXT\_INT\_2\_CON **=**0b010**|(**0b010**<<**4**);**/\*下降沿触发中断\*/

EXT\_INT\_2\_MASK **=**0**;**/\* 取消屏蔽外部中断\*/

VIC0INTENABLE **=**0b1**<<**16**;**

key\_VICADDR **=(**int**)**key\_isr**;**/\*由于key1~key4属于同一中断源，所以只需设置一个地址\*/

\_\_asm\_\_**(**

/\*开中断\*/

"mrs r0,cpsr\n"

"bic r0, r0, #0x80\n"

"msr cpsr\_c, r0\n"

**:**

**:**

**);**

**}**

### gboot.lds

OUTPUT\_ARCH**(**arm**)**

ENTRY**(**\_start**)**

SECTIONS **{**

**.=**0x20008000**;**

**.=** ALIGN**(**4**);**

**.**text **:**

**{**

start.o **(.**text**)**

**\*(.**text**)**

**}**

**.=** ALIGN**(**4**);**

**.**data **:**

**{**

**\*(.**data**)**

**}**

**.=** ALIGN**(**4**);**

bss\_start **=.;**

**.**bss **:**

**{**

**\*(.**bss**)**

**}**

bss\_end **=.;**

**}**

### button.c

#define GPH2CON \*((volatile unsigned long \*)0xe0200c40)

/\*

\* K1,K2,K3,K4对应GPH0\_1、GPH0\_2、GPH0\_3、GPH0\_4

\*/

#define GPH0\_int (0xf<<(0\*4))

#define GPH1\_int (0xf<<(1\*4))

#define GPH2\_int (0xf<<(2\*4))

#define GPH4\_int (0xf<<(4\*4))

#define GPH0\_msk (0xf<<(0\*4))

#define GPH1\_msk (0xf<<(1\*4))

#define GPH2\_msk (0xf<<(2\*4))

#define GPH4\_msk (0xf<<(4\*4))

void button\_init**()**

**{**

GPH2CON **&=~(**GPH0\_msk **|** GPH1\_msk **|** GPH2\_msk **|** GPH4\_msk**);**

GPH2CON **|=** GPH0\_int **|** GPH1\_int **|** GPH2\_int **|** GPH4\_int**;**

**}**

### Makefile

all**:** start.o mem.o main.o led.o mmu.o button.o interrupt.o flash.o

arm-linux-ld -Tgboot.lds -o gboot.elf $^

arm-linux-objcopy -O binary gboot.elf gboot.bin

%.o**:** %.S

arm-linux-gcc -g -c $^

%.o**:** %.c

arm-linux-gcc -g -c $^

.PHONY**:** clean

clean**:**

rm \*.o \*.elf \*.bin

### main.c

int gboot\_main**()**

**{**

int num**;**

int i**;**

#ifdef MMU\_ON

mmu\_init**();**

#endif

led\_init**();**

button\_init**();**

init\_irq**();**

**while(**1**)**

**{**

**;**

**}**

**return**0**;**

**}**

# 专题04-串口来做控制台

[主目录](#_目录)

## 串口核心理论

[主目录](#_目录)

串口用来做什么？

数据传输、充当控制台(发送/接收数据)

串口通讯参数

波特率、数据位、停止位、奇偶校验位

串口通讯格式-分类

同步通讯和异步通讯，我们通常使用都是异步串口。

通讯时，双方先约定好数据帧格式，即波特率、数据位、停止位、奇偶校验位等。

波特率

波特率是一个衡量通讯速度的参数，它表示每秒传送的bit个数。例如（常用）115200表示每秒发送115200bit数据。

这是一个衡量通讯速度的参数。它表示每秒传送的bit个数。例如（常用）115200表示每秒发送115200bit数据。

通讯参数-起始位

表示一帧数据的开始。当线路空闲时，电平为高。一旦检测到一个下降沿，则视为一个起始位，然后接收方按照

约定好的格式，接收这一帧数据。

通讯参数-数据位

一帧数据中实际有效数据的位数。

通讯参数-停止位

表示这帧数据的结束。

通讯参数-校验位

用于检测数据传输是否正确的位。

串口硬件引脚

我们通常使用的RS232的9帧串口，其中最重要的是2,3,5引脚。

RXD-接收数据、TXD-发送数据、GND-接地。

串口工作在哪些模式？

轮询模式、中断模式、DMA模式

起始位：当线路空闲时，电平为高。一旦检测到一个下降沿，则视为一个起始位。然后按照接收方约定好的格式，接收这一帧数据。（1位）

数据位：一帧中实际有效数据的位数。（8位）

停止位：表示这帧数据的结束。（1位）

校验位：用于检测数据传输是否正确的位。（1位）

## 串口驱动程序设计：mini2440

### start.S

**.text**

.global \_start

\_start**:**

b reset

ldr pc**,** \_undifined\_instruction

ldr pc**,** \_software\_interrupt

ldr pc**,** \_prefetch\_abort

ldr pc**,** \_data\_abort

ldr pc**,** \_not\_used

ldr pc**,** \_irq

ldr pc**,** \_fiq

\_undifined\_instruction**:** .word undifined\_instruction

\_software\_interrupt**:** .word software\_interrupt

\_prefetch\_abort**:** .word prefetch\_abort

\_data\_abort**:** .word data\_abort

\_not\_used**:** .word not\_used

\_irq**:** .word irq

\_fiq**:** .word reset

undifined\_instruction**:**

**nop**

software\_interrupt**:**

**nop**

prefetch\_abort**:**

**nop**

data\_abort**:**

**nop**

not\_used**:**

**nop**

irq**:**

**sub** lr**,** lr **,**#4

stmfd **sp**! **,** {r0**-**r12**,**lr} **/\***保存环境**\*/**

**bl** handle\_int

ldmfd **sp**!**,** {r0**-**r12**,** pc}**^/\***恢复环境**\*/**

fiq**:**

**nop**

reset**:**

**bl** set\_svc

**bl** disable\_watchdog

**bl** disable\_interrupt

**bl** disable\_mmu

**bl** init\_clock

**bl** init\_sdram

**bl** copy\_to\_ram

**bl** init\_stack

**bl** clean\_bss

ldr pc**,=**gboot\_main

@ **bl** light\_led

set\_svc**:**

mrs r0**,** cpsr

bic r0**,** r0**,**#0x1f

orr r0**,** r0**,**#0xd3

msr cpsr**,** r0

**mov** pc**,** lr

#define pWTCON 0x53000000

disable\_watchdog**:**

ldr r0**,=**pWTCON

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_interrupt**:**

mvn r1**,** #0x0

ldr r0**,=**0x4a000008

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_mmu**:**

mcr p15**,**0**,**r0**,**c7**,**c7**,**0

mrc p15**,**0**,**r0**,**c1**,**c0**,**0

bic r0**,** r0**,** #0x00000007

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

**mov** pc**,** lr

#define CLKDIVN 0x4c000014

#define MPLLCON 0x4c000008

#define MPLL\_405MHZ **((**127**<<**12**)|(**2**<<**4**)|(**1**<<**0**))**

init\_clock**:**

ldr r0**,=**CLKDIVN

**mov** r1**,** #0x5

**str** r1**,[**r0**]**

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

orr r0**,**r0**,**#0xc0000000

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

ldr r0**,=**MPLLCON

ldr r1**,=**MPLL\_405MHZ

**str** r1**,[**r0**]**

**mov** pc**,** lr

#define mem\_contrl 0x48000000

init\_sdram**:**

ldr r0**,=**mem\_contrl

**add** r3**,** r0**,** #4**\***13

adrl r1**,** mem\_data

0**:**

ldr r2**,[**r1**],** #4

**str** r2**,[**r0**],** #4

**cmp** r0**,** r3

bne 0b

**mov** pc**,** lr

copy\_to\_ram**:**

ldr r0**,=**0x0

ldr r1**,=**0x30008000

**add** r3**,** r0**,** #1024**\***4

copy\_loop**:**

ldr r2**,[**r0**],** #4

**str** r2**,[**r1**],** #4

**cmp** r0**,** r3

bne copy\_loop

**mov** pc**,** lr

init\_stack**:**

msr cpsr\_c**,** #0xd2

ldr **sp,=**0x33000000**//**初始化r13\_irq

msr cpsr\_c**,** #0xd3

ldr **sp,=**0x34000000**//**初始化R13\_svc

**mov** pc **,**lr

clean\_bss**:**

ldr r0**,=**bss\_start

ldr r1**,=**bss\_end

**cmp** r0**,** r1

moveq pc**,** lr

clean\_loop**:**

**mov** r2**,** #0

**str** r2**,[**r0**],** #4

**cmp** r0**,** r1

bne clean\_loop

**mov** pc**,** lr

mem\_data**:**

.long 0x22000000

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00018001

.long 0x00018001

.long 0x008c04f5

.long 0x000000b1

.long 0x00000030

.long 0x00000030

#define GPBCON 0x56000010

#define GPBDAT 0x56000014

light\_led**:**

ldr r0**,=**GPBCON

**mov** r1**,** #0x400

**str** r1**,[**r0**]**

ldr r0**,=**GPBDAT

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

**mov** pc**,** lr

### uart.c

#define GPHCON (\*(volatile unsigned long\*)0x56000070)

#define ULCON0 (\*(volatile unsigned long\*)0x50000000)

#define UCON0 (\*(volatile unsigned long\*)0x50000004)

#define UBRDIV0 (\*(volatile unsigned long\*)0x50000028)

#define UTRSTAT0 (\*(volatile unsigned long\*)0x50000010)

#define UTXH0 (\*(volatile unsigned long\*)0x50000020)

#define URXH0 (\*(volatile unsigned long\*)0x50000024)

#define PCLK 50000000

#define BAUD 115200

void uart\_init**()**

**{**

//1.配置引脚功能

GPHCON **&=~(**0xf**<<**4**);**

GPHCON **|=(**0xa**<<**4**);**

//2.1 设置数据格式

ULCON0 **=**0b11**;**

//2.2 设置工作模式

UCON0 **=**0b0101**;**

//3. 设置波特率

UBRDIV0 **=(**int**)(**PCLK**/(**BAUD**\***16**)-**1**);**

**}**

void putc**(**unsignedchar ch**)**

**{**

**while(!(**UTRSTAT0 **&(**1**<<**2**)));**

UTXH0 **=** ch**;**

**}**

unsignedchar getc**(**void**)**

**{**

unsignedchar ret**;**

**while(!(**UTRSTAT0 **&(**1**<<**0**)));**

// 取数据

ret **=** URXH0**;**

**if((**ret **==**0x0d**)||(**ret **==**0x0a**))**

**{**

putc**(**0x0d**);**

putc**(**0x0a**);**

**}**

**else**

putc**(**ret**);**

**return** ret**;**

**}**

### mmu.c

/\*

\* 用于段描述符的一些宏定义

\*/

#define MMU\_FULL\_ACCESS (3 << 10) /\* 访问权限 \*/

#define MMU\_DOMAIN (0 << 5) /\* 属于哪个域 \*/

#define MMU\_SPECIAL (1 << 4) /\* 必须是1 \*/

#define MMU\_CACHEABLE (1 << 3) /\* cacheable \*/

#define MMU\_BUFFERABLE (1 << 2) /\* bufferable \*/

#define MMU\_SECTION (2 << 0) /\* 段描述符 \*/

#define MMU\_SECDESC (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_SECTION)

#define MMU\_SECDESC\_WB (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_CACHEABLE | MMU\_BUFFERABLE | MMU\_SECTION)

void create\_page\_table**(**void**)**

**{**

unsignedlong**\***ttb **=(**unsignedlong**\*)**0x31000000**;**

unsignedlong vaddr**,** paddr**;**

vaddr **=**0x00000000**;**

paddr **=**0x30000000**;**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xfff00000**)|** MMU\_SECDESC\_WB**;**

vaddr **=**0x56000000**;**

paddr **=**0x56000000**;**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC**;**

vaddr **=**0x4A000000**;**

paddr **=**0x4A000000**;**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xfff00000**)|** MMU\_SECDESC**;**

vaddr **=**0x30000000**;**

paddr **=**0x30000000**;**

**while(**vaddr **<**0x34000000**)**

**{**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC\_WB**;**

vaddr **+=**0x100000**;**

paddr **+=**0x100000**;**

**}**

**}**

void mmu\_enable**()**

**{**

\_\_asm\_\_**(**

/\*设置TTB\*/

"ldr r0, =0x31000000\n"

"mcr p15, 0, r0, c2, c0, 0\n"

/\*不进行权限检查\*/

"mvn r0, #0\n"

"mcr p15, 0, r0, c3, c0, 0\n"

/\*使能MMU\*/

"mrc p15, 0, r0, c1, c0, 0\n"

"orr r0, r0, #0x0001\n"

"mcr p15, 0, r0, c1, c0, 0\n"

**:**

**:**

**);**

**}**

void mmu\_init**()**

**{**

create\_page\_table**();**

mmu\_enable**();**

**}**

### Makefile

all**:** start.o main.o mmu.o led.o interrupt.o button.o uart.o

arm-linux-ld -Tgboot.lds -o gboot.elf $^

arm-linux-objcopy -O binary gboot.elf gboot.bin

%.o**:** %.S

arm-linux-gcc -g -c $^

%.o**:** %.c

arm-linux-gcc -g -c $^

.PHONY**:** clean

clean**:**

rm \*.o \*.elf \*.bin

### main.c

int gboot\_main**()**

**{**

#ifdef MMU\_ON

mmu\_init**();**

#endif

led\_init**();**

button\_init**();**

init\_irq**();**

uart\_init**();**

**while(**1**)**

getc**();**

**return**0**;**

**}**

### led.c

#define GPBCON (volatile unsigned long\*)0x56000010

#define GPBDAT (volatile unsigned long\*)0x56000014

void led\_init**()**

**{**

**\*(**GPBCON**)=**0x400**;**

**}**

void led\_off**()**

**{**

**\*(**GPBDAT**)=**0x7ff**;**

**}**

void led\_on**()**

**{**

**\*(**GPBDAT**)=**0x0**;**

**}**

### interrupt.c

/\*interrupt registes\*/

#define SRCPND (volatile unsigned long \*)0x4A000000

#define INTMOD (volatile unsigned long \*)0x4A000004

#define INTMSK (volatile unsigned long \*)0x4A000008

#define PRIORITY (volatile unsigned long \*)0x4A00000c

#define INTPND (volatile unsigned long \*)0x4A000010

#define INTOFFSET (volatile unsigned long \*)0x4A000014

#define SUBSRCPND (volatile unsigned long \*)0x4A000018

#define INTSUBMSK (volatile unsigned long \*)0x4A00001c

#define EINTMASK (volatile unsigned long \*)0x560000a4

#define EINTPEND (volatile unsigned long \*)0x560000a8

void init\_irq**()**

**{**

// 在EINTMASK寄存器中使能它们

**\*(**EINTMASK**)&=(~(**1**<<**8**))&(~(**1**<<**11**))&(~(**1**<<**13**))&(~(**1**<<**14**));**

// EINT8\_23使能

**\*(**INTMSK**)&=~(**1**<<**5**);**

\_\_asm\_\_**(**

/\*开中断\*/

"mrs r0,cpsr\n"

"bic r0, r0, #0x80\n"

"msr cpsr\_c, r0\n"

**:**

**:**

**);**

**}**

void handle\_int**()**

**{**

/\*读取产生中断的源\*/

unsignedlong value **=\*(**EINTPEND**)&((**1**<<**8**)|(**1**<<**11**)|(**1**<<**13**)|(**1**<<**14**));**

**switch(**value**)**

**{**

**case(**1**<<**8**):**//K1

led\_on**();**

**break;**

**case(**1**<<**11**):**//K2

led\_off**();**

**break;**

**case(**1**<<**13**):**//K3

led\_on**();**

**break;**

**case(**1**<<**14**):**//K4

led\_off**();**

**break;**

**default:**

**break;**

**}**

/\* 中断清除 \*/

**\*(**EINTPEND**)=** value**;**

**\*(**SRCPND**)=**1**<<**5**;**

**\*(**INTPND**)=**1**<<**5**;**

**}**

### gboot.lds

OUTPUT\_ARCH**(**arm**)**

ENTRY**(**\_start**)**

SECTIONS **{**

**.=**0x30008000**;**

**.=** ALIGN**(**4**);**

**.**text **:**

**{**

start.o **(.**text**)**

**\*(.**text**)**

**}**

**.=** ALIGN**(**4**);**

**.**data **:**

**{**

**\*(.**data**)**

**}**

**.=** ALIGN**(**4**);**

bss\_start **=.;**

**.**bss **:**

**{**

**\*(.**bss**)**

**}**

bss\_end **=.;**

**}**

### button.c

#define GPGCON (volatile unsigned long \*)0x56000060

/\*

\* K1,K2,K3,K4对应GPG0、GPG3、GPG5、GPG6

\*/

#define GPG0\_int (0x2<<(0\*2))

#define GPG3\_int (0x2<<(3\*2))

#define GPG5\_int (0x2<<(5\*2))

#define GPG6\_int (0x2<<(6\*2))

#define GPG0\_msk (3<<(0\*2))

#define GPG3\_msk (3<<(3\*2))

#define GPG5\_msk (3<<(5\*2))

#define GPG6\_msk (3<<(6\*2))

void button\_init**()**

**{**

**\*(**GPGCON**)&=~(**GPG0\_msk **|** GPG3\_msk **|** GPG5\_msk **|** GPG6\_msk**);**

**\*(**GPGCON**)|=** GPG0\_int **|** GPG3\_int **|** GPG5\_int **|** GPG6\_int**;**

**}**

## 串口驱动程序设计：OK6410

### start.S

**.text**

.global \_start

\_start**:**

b reset

ldr pc**,** \_undifined\_instruction

ldr pc**,** \_software\_interrupt

ldr pc**,** \_prefetch\_abort

ldr pc**,** \_data\_abort

ldr pc**,** \_not\_used

ldr pc**,** \_irq

ldr pc**,** \_fiq

\_undifined\_instruction**:** .word undifined\_instruction

\_software\_interrupt**:** .word software\_interrupt

\_prefetch\_abort**:** .word prefetch\_abort

\_data\_abort**:** .word data\_abort

\_not\_used**:** .word not\_used

\_irq**:** .word irq

\_fiq**:** .word reset

undifined\_instruction**:**

**nop**

software\_interrupt**:**

**nop**

prefetch\_abort**:**

**nop**

data\_abort**:**

**nop**

not\_used**:**

**nop**

irq**:**

**nop**

fiq**:**

**nop**

reset**:**

**bl** set\_svc

**bl** set\_peri\_port

**bl** disable\_watchdog

**bl** disable\_interrupt

**bl** disable\_mmu

**bl** init\_clock

**bl** mem\_init

**bl** copy\_to\_ram

**bl** init\_stack

**bl** clean\_bss

ldr pc**,=**gboot\_main

@ **bl** light\_led

set\_peri\_port**:**

ldr r0**,=**0x70000000

orr r0**,** r0**,** #0x13

mcr p15**,**0**,**r0**,**c15**,**c2**,**4

**mov** pc**,** lr

set\_svc**:**

mrs r0**,** cpsr

bic r0**,** r0**,**#0x1f

orr r0**,** r0**,**#0xd3

msr cpsr**,** r0

**mov** pc**,** lr

#define pWTCON 0x7e004000

disable\_watchdog**:**

ldr r0**,=**pWTCON

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_interrupt**:**

mvn r1**,**#0x0

ldr r0**,=**0x71200014

**str** r1**,[**r0**]**

ldr r0**,=**0x71300014

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_mmu**:**

mcr p15**,**0**,**r0**,**c7**,**c7**,**0

mrc p15**,**0**,**r0**,**c1**,**c0**,**0

bic r0**,** r0**,** #0x00000007

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

**mov** pc**,** lr

#define CLK\_DIV0 0x7e00f020

#define OTHERS 0x7e00f900

#define MPLL\_CON 0x7e00f010

#define APLL\_CON 0x7e00f00c

#define CLK\_SRC 0x7e00f01c

#define DIV\_VAL **((**0x0**<<**0**)|(**0x1**<<**9**)|(**0x1**<<**8**)|(**0x3**<<**12**))**

#define PLL\_VAL **((**1**<<**31**)|(**266**<<**16**)|(**3**<<**8**)|(**1**<<**0**))**

init\_clock**:**

ldr r0**,=**CLK\_DIV0

ldr r1**,=**DIV\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**OTHERS

ldr r1**,[**r0**]**

bic r1**,**r1**,**#0xc0

**str** r1**,[**r0**]**

ldr r0**,=**APLL\_CON

ldr r1**,=**PLL\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**MPLL\_CON

ldr r1**,=**PLL\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**CLK\_SRC

**mov** r1**,** #0x3

**str** r1**,[**r0**]**

**mov** pc**,** lr

copy\_to\_ram**:**

ldr r0**,=**0x0c000000

ldr r1**,=**0x50008000

**add** r3**,** r0**,** #1024**\***4

copy\_loop**:**

ldr r2**,[**r0**],** #4

**str** r2**,[**r1**],** #4

**cmp** r0**,** r3

bne copy\_loop

**mov** pc**,** lr

init\_stack**:**

msr cpsr\_c**,** #0xd2

ldr **sp,=**0x53000000**//**初始化r13\_irq

msr cpsr\_c**,** #0xd3

ldr **sp,=**0x54000000**//**初始化R13\_svc

**mov** pc **,**lr

clean\_bss**:**

ldr r0**,=**bss\_start

ldr r1**,=**bss\_end

**cmp** r0**,** r1

moveq pc**,** lr

clean\_loop**:**

**mov** r2**,** #0

**str** r2**,[**r0**],** #4

**cmp** r0**,** r1

bne clean\_loop

**mov** pc**,** lr

light\_led**:**

ldr r1**,=**0x7F008820

ldr r0**,=**0x1111

**str** r0**,[**r1**]**

ldr r1**,=**0x7F008824

**mov** r0**,**#0xe

**str** r0**,[**r1**]**

**mov** pc**,** lr

### uart.c

#define GPACON (\*((volatile unsigned short \*)0x7F008000))

#define ULCON0 (\*((volatile unsigned long \*)0x7F005000))

#define UCON0 (\*((volatile unsigned long \*)0x7F005004))

#define UTRSTAT0 (\*((volatile unsigned long \*)0x7F005010))

#define UBRDIV0 (\*((volatile unsigned short \*)0x7F005028))

#define UTXH0 (\*((volatile unsigned char \*)0x7F005020))

#define URXH0 (\*((volatile unsigned char \*)0x7F005024))

#define UDIVSLOT0 (\*((volatile unsigned short \*)0x7F00502C))

#define PCLK 66500000

#define BAUD 115200

void uart\_init**()**

**{**

//1.配置引脚功能

GPACON **&=~**0xff**;**

GPACON **|=**0x22**;**

//2.1 设置数据格式

ULCON0 **=**0b11**;**

//2.2 设置工作模式

UCON0 **=**0b0101**;**

//3. 设置波特率

UBRDIV0 **=(**int**)(**PCLK**/(**BAUD**\***16**)-**1**);**//UBRDIV0保存该公式计算后的整数部分

UDIVSLOT0 **=**0x0**;**//UDISLOT0=保存该公式计算后的小数部分\*16

**}**

void putc**(**unsignedchar ch**)**

**{**

**while(!(**UTRSTAT0 **&(**1**<<**2**)));**

UTXH0 **=** ch**;**

**}**

unsignedchar getc**(**void**)**

**{**

unsignedchar ret**;**

**while(!(**UTRSTAT0 **&(**1**<<**0**)));**

// 取数据

ret **=** URXH0**;**

**if((**ret **==**0x0d**)||(**ret **==**0x0a**))**

**{**

putc**(**0x0d**);**

putc**(**0x0a**);**

**}**

**else**

putc**(**ret**);**

**return** ret**;**

**}**

### mmu.c

/\*

\* 用于段描述符的一些宏定义

\*/

#define MMU\_FULL\_ACCESS (3 << 10) /\* 访问权限 \*/

#define MMU\_DOMAIN (0 << 5) /\* 属于哪个域 \*/

#define MMU\_SPECIAL (1 << 4) /\* 必须是1 \*/

#define MMU\_CACHEABLE (1 << 3) /\* cacheable \*/

#define MMU\_BUFFERABLE (1 << 2) /\* bufferable \*/

#define MMU\_SECTION (2) /\* 表示这是段描述符 \*/

#define MMU\_SECDESC (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_SECTION)

#define MMU\_SECDESC\_WB (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_CACHEABLE | MMU\_BUFFERABLE | MMU\_SECTION)

void create\_page\_table**(**void**)**

**{**

unsignedlong**\***ttb **=(**unsignedlong**\*)**0x50000000**;**

unsignedlong vaddr**,** paddr**;**

vaddr **=**0xA0000000**;**

paddr **=**0x7f000000**;**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC**;**

vaddr **=**0x50000000**;**

paddr **=**0x50000000**;**

**while(**vaddr **<**0x54000000**)**

**{**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC\_WB**;**

vaddr **+=**0x100000**;**

paddr **+=**0x100000**;**

**}**

**}**

void mmu\_enable**()**

**{**

\_\_asm\_\_**(**

/\*设置TTB\*/

"ldr r0, =0x50000000\n"

"mcr p15, 0, r0, c2, c0, 0\n"

/\*不进行权限检查\*/

"mvn r0, #0\n"

"mcr p15, 0, r0, c3, c0, 0\n"

/\*使能MMU\*/

"mrc p15, 0, r0, c1, c0, 0\n"

"orr r0, r0, #0x0001\n"

"mcr p15, 0, r0, c1, c0, 0\n"

**:**

**:**

**);**

**}**

void mmu\_init**()**

**{**

create\_page\_table**();**

mmu\_enable**();**

**}**

### mem.S

**.text**

.global mem\_init

mem\_init**:**

ldr r0**,=**0x7e00f120

**mov** r1**,** #0x8

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001004 @内存控制命令寄存器

**mov** r1**,** #0x4 @根据手册得知需要先进入配置模式

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001010 @刷新寄存器地址

ldr r1**,=((**7800**/(**1000000000**/**133000000**)+**1**))** @设置刷新时间

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001014 @CAS latency寄存器

**mov** r1**,** #**(**3**<<**1**)**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001018 @t\_DQSS寄存器

**mov** r1**,** #0x1

**str** r1**,[**r0**]**

ldr r0**,=**0x7e00101c @T\_MRD寄存器

**mov** r1**,** #0x2

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001020 @t\_RAS寄存器

ldr r1**,=((**45**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001024 @t\_RC寄存器

ldr r1**,=((**68**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001028 @t\_RCD寄存器

ldr r1**,=((**23**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e00102c @t\_RFC寄存器

ldr r1**,=((**80**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001030 @t\_RP寄存器

ldr r1**,=((**23**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001034 @t\_rrd寄存器

ldr r1**,=((**15**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001038 @t\_wr寄存器

ldr r1**,=((**15**/(**1000000000**/**133000000**)+**1**))**

@ ldr r2**,[**r0**]**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e00103c @t\_wtr寄存器

**mov** r1**,** #0x07

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001040 @t\_xp寄存器

**mov** r1**,** #0x02

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001044 @t\_xsr寄存器

ldr r1**,=((**120**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001048 @t\_esr寄存器

ldr r1**,=((**120**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e00100c @内存控制配置寄存器

ldr r1**,=**0x00010012 @配置控制器

**str** r1**,[**r0**]**

ldr r0**,=**0x7e00104c @32位DRAM配置控制寄存器

ldr r1**,=**0x0b45

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001200 @片选寄存器

ldr r1**,=**0x150f8

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001304 @用户配置寄存器

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001008

ldr r1**,=**0x000c0000

**str** r1**,[**r0**]**

ldr r1**,=**0x00000000

**str** r1**,[**r0**]**

ldr r1**,=**0x00040000

**str** r1**,[**r0**]**

ldr r1**,=**0x000a0000

**str** r1**,[**r0**]**

ldr r1**,=**0x00080032

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001004

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

check\_dmc1\_ready**:**

ldr r0**,=**0x7e001000

ldr r1**,[**r0**]**

**mov** r2**,** #0x3

**and** r1**,** r1**,** r2

**cmp** r1**,** #0x1

bne check\_dmc1\_ready

**nop**

**mov** pc**,** lr

### Makefile

all**:** start.o main.o mmu.o led.o mem.o interrupt.o button.o uart.o

arm-linux-ld -Tgboot.lds -o gboot.elf $^

arm-linux-objcopy -O binary gboot.elf gboot.bin

%.o**:** %.S

arm-linux-gcc -g -c $^

%.o**:** %.c

arm-linux-gcc -g -c $^

.PHONY**:** clean

clean**:**

rm \*.o \*.elf \*.bin

### led.c

#define GPKCON (volatile unsigned long\*)0x7f008800

#define GPKDAT (volatile unsigned long\*)0x7f008808

void led\_init**()**

**{**

**\*(**GPKCON**)=**0x1111**;**

**}**

void led\_on**()**

**{**

**\*(**GPKDAT**)=**0x0e**;**

**}**

void led\_off**()**

**{**

**\*(**GPKDAT**)=**0x0f**;**

**}**

### interrupt.c

/\*interrupt registes\*/

#define EXT\_INT\_0\_CON \*((volatile unsigned int \*)0x7f008900)

#define EXT\_INT\_0\_MASK \*((volatile unsigned int \*)0x7f008920)

#define EXT\_INT\_0\_PEND \*((volatile unsigned int \*)0x7f008924)

#define VIC0INTENABLE \*((volatile unsigned int \*)0x71200010)

#define EINT0\_VECTADDR \*((volatile unsigned int \*)0x71200100)

#define EINT5\_VECTADDR \*((volatile unsigned int \*)0x71200104)

#define VIC0ADDRESS \*((volatile unsigned int \*)0x71200f00)

#define VIC1ADDRESS \*((volatile unsigned int \*)0x71300f00)

void key1\_handle**()**

**{**

\_\_asm\_\_**(**

"sub lr, lr, #4\n"

"stmfd sp!, {r0-r12, lr}\n"

**:**

**:**

**);**

led\_on**();**

/\* 清除中断 \*/

EXT\_INT\_0\_PEND **=~**0x0**;**

VIC0ADDRESS **=**0**;**

VIC1ADDRESS **=**0**;**

\_\_asm\_\_**(**

"ldmfd sp!, {r0-r12, pc}^ \n"

**:**

**:**

**);**

**}**

void key6\_handle**()**

**{**

\_\_asm\_\_**(**

"sub lr, lr, #4\n"

"stmfd sp!, {r0-r12, lr}\n"

**:**

**:**

**);**

led\_off**();**

/\* 清除中断 \*/

EXT\_INT\_0\_PEND **=~**0x0**;**

VIC0ADDRESS **=**0**;**

VIC1ADDRESS **=**0**;**

\_\_asm\_\_**(**

"ldmfd sp!, {r0-r12, pc}^ \n"

**:**

**:**

**);**

**}**

void init\_irq**()**

**{**

EXT\_INT\_0\_CON **=(**0b010**)|(**0b010**<<**8**);**/\* 配置为下降沿触发 \*/

EXT\_INT\_0\_MASK **=**0**;**/\* 取消屏蔽外部中断 \*/

VIC0INTENABLE **|=(**0b1**)|(**0b10**);**/\* 使能外部中断\*/

EINT0\_VECTADDR **=(**int**)**key1\_handle**;**/\* 用户按下key时，CPU就会自动的将VIC0VECTADDR0的值赋给VIC0ADDRESS并跳转到这个地址去执 \*/

EINT5\_VECTADDR **=(**int**)**key6\_handle**;**

\_\_asm\_\_**(**

"mrc p15,0,r0,c1,c0,0\n"

"orr r0,r0,#(1<<24)\n"

"mcr p15,0,r0,c1,c0,0\n"

"mrs r0,cpsr\n"

"bic r0, r0, #0x80\n"

"msr cpsr\_c, r0\n"

**:**

**:**

**);**

**}**

### gboot.lds

OUTPUT\_ARCH**(**arm**)**

ENTRY**(**\_start**)**

SECTIONS **{**

**.=**0x50008000**;**

**.=** ALIGN**(**4**);**

**.**text **:**

**{**

start.o **(.**text**)**

**\*(.**text**)**

**}**

**.=** ALIGN**(**4**);**

**.**data **:**

**{**

**\*(.**data**)**

**}**

**.=** ALIGN**(**4**);**

bss\_start **=.;**

**.**bss **:**

**{**

**\*(.**bss**)**

**}**

bss\_end **=.;**

**}**

### button.c

#define GPNCON (volatile unsigned long\*)0x7f008830

void button\_init**()**

**{**

**\*(**GPNCON**)=**0b10**|(**0b10**<<**10**);**//设置按键1与按键6

**}**

### main.c

int gboot\_main**()**

**{**

#ifdef MMU\_ON

mmu\_init**();**

#endif

led\_init**();**

button\_init**();**

init\_irq**();**

uart\_init**();**

**while(**1**)**

getc**();**

**return**0**;**

**}**

## 串口驱动程序设计：TQ210

### start.S

@**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

@File**:**start.S

@

@成都国嵌嵌入式培训中心

@**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**.text**

.global \_start

\_start**:**

b reset

ldr pc**,** \_undefined\_instruction

ldr pc**,** \_software\_interrupt

ldr pc**,** \_prefetch\_abort

ldr pc**,** \_data\_abort

ldr pc**,** \_not\_used

ldr pc**,** \_irq

ldr pc**,** \_fiq

\_undefined\_instruction**:** .word undefined\_instruction

\_software\_interrupt**:** .word software\_interrupt

\_prefetch\_abort**:** .word prefetch\_abort

\_data\_abort**:** .word data\_abort

\_not\_used**:** .word not\_used

\_irq**:** .word irq

\_fiq**:** .word fiq

undefined\_instruction**:**

**nop**

software\_interrupt**:**

**nop**

prefetch\_abort**:**

**nop**

data\_abort**:**

**nop**

not\_used**:**

**nop**

irq**:**

**nop**

fiq**:**

**nop**

reset**:**

**bl** set\_svc

**bl** disable\_watchdog

**bl** disable\_interrupt

**bl** disable\_mmu

**bl** init\_clock

**bl** mem\_init

**bl** copy\_to\_ram

**bl** init\_stack

**bl** clean\_bss

ldr pc**,=**gboot\_main

@ **bl** light\_led

set\_svc**:**

mrs r0**,** cpsr

bic r0**,** r0**,** #0x1f

orr r0**,** r0**,** #0xd3

msr cpsr**,** r0

**mov** pc**,** lr

#define pWTCON 0xE2700000

disable\_watchdog**:**

ldr r0**,=**pWTCON

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_interrupt**:**

mvn r1**,**#0x0

ldr r0**,=**0xf2000014

**str** r1**,[**r0**]**

ldr r0**,=**0xf2100014

**str** r1**,[**r0**]**

ldr r0**,=**0xf2200014

**str** r1**,[**r0**]**

ldr r0**,=**0xf2300014

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_mmu**:**

mcr p15**,**0**,**r0**,**c7**,**c5**,**0

mcr p15**,**0**,**r0**,**c7**,**c6**,**1

mrc p15**,**0**,**r0**,**c1**,**c0**,**0

bic r0**,** r0**,** #0x00000007

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

**mov** pc**,** lr

#define CLK\_DIV0 0xe0100300

#define MPLL\_CON 0xe0100108

#define APLL\_CON 0xe0100100

#define CLK\_SRC 0xe0100200

#define DIV\_VAL **((**0x0**<<**0**)|(**4**<<**4**)|(**4**<<**8**)|(**1**<<**12**)|(**3**<<**16**)|(**1**<<**20**)|(**4**<<**24**)|(**1**<<**28**))**

#define APLL\_VAL **((**1**<<**31**)|(**3**<<**8**)|(**125**<<**16**)|(**1**<<**0**))**

#define MPLL\_VAL **((**1**<<**31**)|(**12**<<**8**)|(**667**<<**16**)|(**1**<<**0**))**

init\_clock**:**

ldr r0**,=**CLK\_DIV0

ldr r1**,=**DIV\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**APLL\_CON

ldr r1**,=**APLL\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**MPLL\_CON

ldr r1**,=**MPLL\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**CLK\_SRC

ldr r1**,=**0x1111

**str** r1**,[**r0**]**

**mov** pc**,** lr

copy\_to\_ram**:**

ldr r0**,=**0xd0020010

ldr r1**,=**0x20008000

**add** r3**,** r0**,** #1024**\***4

copy\_loop**:**

ldr r2**,[**r0**],** #4

**str** r2**,[**r1**],** #4

**cmp** r0**,** r3

bne copy\_loop

**mov** pc**,** lr

init\_stack**:**

msr cpsr\_c**,** #0xd2

ldr **sp,=**0x23000000 @此处实际设置的是r13\_irq

msr cpsr\_c**,** #0xd3

ldr **sp,=**0x24000000 @此处实际设置的是r13\_svc

**mov** pc**,** lr

init\_stacktmp**:**

ldr **sp,=**0xd0030000

**mov** pc **,**lr

clean\_bss**:**

ldr r0**,=**bss\_start

ldr r1**,=**bss\_end

**cmp** r0**,** r1

moveq pc**,** lr

clean\_loop**:**

**mov** r2**,** #0

**str** r2**,[**r0**],** #4

**cmp** r0**,** r1

bne clean\_loop

**mov** pc**,** lr

light\_led**:**

ldr R0**,=**0xE0200060

ldr R1**,=**0x00011000

**str** R1**,[**R0**]**

ldr R0**,=**0xE0200064

ldr R1**,=**0x00000018

**str** R1**,[**R0**]**

**mov** pc**,** lr

### uart.c

#define GPA0CON ( \*((volatile unsigned long \*)0xE0200000) )

#define GPA1CON ( \*((volatile unsigned long \*)0xE0200020) )

#define ULCON0 ( \*((volatile unsigned long \*)0xE2900000) )

#define UCON0 ( \*((volatile unsigned long \*)0xE2900004) )

#define UBRDIV0 ( \*((volatile unsigned long \*)0xE2900028) )

#define UDIVSLOT0 ( \*((volatile unsigned long \*)0xE290002C) )

#define UTRSTAT0 ( \*((volatile unsigned long \*)0xE2900010) )

#define UTXH0 ( \*((volatile unsigned long \*)0xE2900020) )

#define URXH0 ( \*((volatile unsigned long \*)0xE2900024) )

#define PCLK\_PSYS 66000000 // HCLK\_PSYS=MPLL/(HCLK\_PSYS\_RATIO+1); PCLK\_PSYS=HCLK\_PSYS/(PCLK\_PSYS\_RATIO+1); MPLL=667MHZ(参考时钟初始化那个专题)

#define BAUD 115200

void uart\_init**()**

**{**

//1.配置引脚功能

GPA0CON **=**0x22222222**;**

GPA1CON **=**0x2222**;**

//2.1 设置数据格式

ULCON0 **=**0b11**;**

//2.2 设置工作模式

UCON0 **=**0b0101**;**

//3. 设置波特率

UBRDIV0 **=(**int**)(**PCLK\_PSYS **/(**BAUD**\***16**)-**1**);**//UBRDIV0保存该公式计算后的整数部分

UDIVSLOT0 **=**0x0**;**//UDISLOT0=保存该公式计算后的小数部分\*16

**}**

void putc**(**unsignedchar ch**)**

**{**

**while(!(**UTRSTAT0 **&(**1**<<**2**)));**

UTXH0 **=** ch**;**

**}**

unsignedchar getc**(**void**)**

**{**

unsignedchar ret**;**

**while(!(**UTRSTAT0 **&(**1**<<**0**)));**

// 取数据

ret **=** URXH0**;**

**if((**ret **==**0x0d**)||(**ret **==**0x0a**))**

**{**

putc**(**0x0d**);**

putc**(**0x0a**);**

**}**

**else**

putc**(**ret**);**

**return** ret**;**

**}**

### mmu.c

/\*

\* 用于段描述符的一些宏定义

\*/

#define MMU\_FULL\_ACCESS (3 << 10) /\* 访问权限 \*/

#define MMU\_DOMAIN (0 << 5) /\* 属于哪个域 \*/

#define MMU\_SPECIAL (1 << 4) /\* 必须是1 \*/

#define MMU\_CACHEABLE (1 << 3) /\* cacheable \*/

#define MMU\_BUFFERABLE (1 << 2) /\* bufferable \*/

#define MMU\_SECTION (2) /\* 表示这是段描述符 \*/

#define MMU\_SECDESC (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_SECTION)

#define MMU\_SECDESC\_WB (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_CACHEABLE | MMU\_BUFFERABLE | MMU\_SECTION)

void create\_page\_table**(**void**)**

**{**

unsignedlong**\***ttb **=(**unsignedlong**\*)**0x20000000**;**

unsignedlong vaddr**,** paddr**;**

vaddr **=**0xA0000000**;**

paddr **=**0xE0200000**;**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC**;**

vaddr **=**0x20000000**;**

paddr **=**0x20000000**;**

**while(**vaddr **<**0x24000000**)**

**{**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC\_WB**;**

vaddr **+=**0x100000**;**

paddr **+=**0x100000**;**

**}**

**}**

void mmu\_enable**()**

**{**

\_\_asm\_\_**(**

/\*设置TTB\*/

"ldr r0, =0x20000000\n"

"mcr p15, 0, r0, c2, c0, 0\n"

/\*不进行权限检查\*/

"mvn r0, #0\n"

"mcr p15, 0, r0, c3, c0, 0\n"

/\*使能MMU\*/

"mrc p15, 0, r0, c1, c0, 0\n"

"orr r0, r0, #0x0001\n"

"mcr p15, 0, r0, c1, c0, 0\n"

**:**

**:**

**);**

**}**

void mmu\_init**()**

**{**

create\_page\_table**();**

mmu\_enable**();**

**}**

### mem.S

#define DMC\_PHYCONTROL0 0xf0000018

#define DMC\_PHYCONTROL1 0xf000001c

#define DMC\_CONCONTROL 0xf0000000

#define DMC\_MEMCONTROL 0xf0000004

#define DMC\_MEMCONFIG0 0xf0000008

#define DMC\_MEMCONFIG1 0xf000000c

#define DMC\_PRECHCONFIG 0xf0000014

#define DMC\_TIMINGAREF 0xf0000030

#define DMC\_TIMINGROW 0xf0000034

#define DMC\_TIMINGDATA 0xf0000038

#define DMC\_TIMINGPOWER 0xf000003c

#define DMC\_PHYSTATUS 0xf0000040

#define DMC\_DIRECTCMD 0xf0000010

#define DMC\_PWRDNCONFIG 0xf0000028

#define DMC0\_MEMCONTROL 0x00202400

#define DMC0\_MEMCONFIG\_0 0x20F00313

#define DMC0\_MEMCONFIG\_1 0x00F00313

#define DMC0\_TIMINGA\_REF 0x00000618

#define DMC0\_TIMING\_ROW 0x2B34438A

#define DMC0\_TIMING\_DATA 0x24240000

#define DMC0\_TIMING\_PWR 0x0BDC0343

.globl mem\_init

mem\_init**:**

ldr r0**,=**DMC\_PHYCONTROL0

ldr r1**,=**0x00101000

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_PHYCONTROL1

ldr r1**,=**0x00000086

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_PHYCONTROL0

ldr r1**,=**0x00101002

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_PHYCONTROL0

ldr r1**,=**0x00101003

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_CONCONTROL

ldr r1**,=**0x0FFF2010

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_MEMCONTROL

ldr r1**,=**DMC0\_MEMCONTROL

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_MEMCONFIG0

ldr r1**,=**DMC0\_MEMCONFIG\_0

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_MEMCONFIG1

ldr r1**,=**DMC0\_MEMCONFIG\_1

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_PRECHCONFIG

ldr r1**,=**0xFF000000

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_TIMINGAREF

ldr r1**,=**DMC0\_TIMINGA\_REF

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_TIMINGROW

ldr r1**,=**DMC0\_TIMING\_ROW

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_TIMINGDATA

ldr r1**,=**DMC0\_TIMING\_DATA

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_TIMINGPOWER

ldr r1**,=**DMC0\_TIMING\_PWR

**str** r1**,[**r0**]**

wait\_lock**:**

**//Loop** until DLL is locked

ldr r0**,=**DMC\_PHYSTATUS

ldr r1**,[**r0**]**

**and** r2**,** r1**,** #0x4

**cmp** r2**,** #0x4

bne wait\_lock

ldr r0**,=**DMC\_DIRECTCMD

ldr r1**,=**0x07000000

**str** r1**,[**r0**]**

ldr r1**,=**0x01000000

**str** r1**,[**r0**]**

ldr r1**,=**0x00020000

**str** r1**,[**r0**]**

ldr r1**,=**0x00030000

**str** r1**,[**r0**]**

ldr r1**,=**0x00010400

**str** r1**,[**r0**]**

ldr r1**,=**0x00000542

**str** r1**,[**r0**]**

ldr r1**,=**0x01000000

**str** r1**,[**r0**]**

ldr r1**,=**0x05000000

**str** r1**,[**r0**]**

ldr r1**,=**0x05000000

**str** r1**,[**r0**]**

ldr r1**,=**0x00000442

**str** r1**,[**r0**]**

ldr r1**,=**0x00010780

**str** r1**,[**r0**]**

ldr r1**,=**0x00010400

**str** r1**,[**r0**]**

ldr r1**,=**0x07100000

**str** r1**,[**r0**]**

ldr r1**,=**0x01100000

**str** r1**,[**r0**]**

ldr r1**,=**0x00120000

**str** r1**,[**r0**]**

ldr r1**,=**0x00130000

**str** r1**,[**r0**]**

ldr r1**,=**0x00110400

**str** r1**,[**r0**]**

ldr r1**,=**0x00100542

**str** r1**,[**r0**]**

ldr r1**,=**0x01100000

**str** r1**,[**r0**]**

ldr r1**,=**0x05100000

**str** r1**,[**r0**]**

ldr r1**,=**0x05100000

**str** r1**,[**r0**]**

ldr r1**,=**0x00100442

**str** r1**,[**r0**]**

ldr r1**,=**0x00110780

**str** r1**,[**r0**]**

ldr r1**,=**0x00110400

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_CONCONTROL

ldr r1**,=**0x0FF02030

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_PWRDNCONFIG

ldr r1**,=**0xFFFF00FF

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_CONCONTROL

ldr r1**,=**0x00202400

**str** r1**,[**r0**]**

**mov** pc**,** lr

### Makefile

all**:** start.o mem.o main.o button.o mmu.o interrupt.o led.o uart.o

arm-linux-ld -Tgboot.lds -o gboot.elf $^

arm-linux-objcopy -O binary gboot.elf gboot.bin

%.o**:** %.S

arm-linux-gcc -g -c $^

%.o**:** %.c

arm-linux-gcc -g -c $^

.PHONY**:** clean

clean**:**

rm \*.o \*.elf \*.bin

### led.c

#define GPC0CON \*((volatile unsigned long\*)0xE0200060)

#define GPC0DAT \*((volatile unsigned long\*)0xE0200064)

void led\_init**()**

**{**

GPC0CON **=**0x00011000**;**

**}**

void led\_off**()**

**{**

GPC0DAT **=**0x0**;**

**}**

void led\_on**()**

**{**

GPC0DAT **=**0x00000018**;**

**}**

### interrupt.c

/\*interrupt registes\*/

#define EXT\_INT\_0\_CON \*((volatile unsigned int \*)0xE0200E00)

#define EXT\_INT\_0\_MASK \*((volatile unsigned int \*)0xE0200F00)

#define VIC0INTSELECT \*((volatile unsigned int \*)0xF200000C)

#define VIC0INTENABLE \*((volatile unsigned int \*)0xF2000010)

#define key1\_VICADDR \*((volatile unsigned int \*)0xF2000100)

#define key2\_VICADDR \*((volatile unsigned int \*)0xF2000104)

#define VIC0ADDRESS \*((volatile unsigned int \*)0xF2000F00)

#define EXT\_INT\_0\_PEND \*((volatile unsigned int \*)0xE0200F40)

void key1\_isr**()**

**{**

\_\_asm\_\_**(**

"sub lr, lr, #4\n"

"stmfd sp!, {r0-r12, lr}\n"

**:**

**:**

**);**

led\_on**();**

/\* 清除中断 \*/

EXT\_INT\_0\_PEND **=~**0x0**;**

VIC0ADDRESS **=**0**;**

\_\_asm\_\_**(**

"ldmfd sp!, {r0-r12, pc}^ \n"

**:**

**:**

**);**

**}**

void key2\_isr**()**

**{**

\_\_asm\_\_**(**

"sub lr, lr, #4\n"

"stmfd sp!, {r0-r12, lr}\n"

**:**

**:**

**);**

led\_off**();**

/\* 清除中断 \*/

EXT\_INT\_0\_PEND **=~**0x0**;**

VIC0ADDRESS **=**0**;**

\_\_asm\_\_**(**

"ldmfd sp!, {r0-r12, pc}^ \n"

**:**

**:**

**);**

**}**

void init\_irq**()**

**{**

EXT\_INT\_0\_CON **&=~(**0xFF**<<**0**);**

EXT\_INT\_0\_CON **|=**2**|(**2**<<**4**);**/\* 配置EXT\_INT[0]和EXT\_INT[1]为下降沿触发 \*/

EXT\_INT\_0\_MASK **&=~**3**;**/\* 取消屏蔽外部中断EXT\_INT[0]和EXT\_INT[1] \*/

VIC0INTENABLE **|=**3**;**/\* 使能外部中断EXT\_INT[0]和EXT\_INT[1] \*/

key1\_VICADDR **=(**int**)**key1\_isr**;**/\* 当EXT\_INT[0]触发中断，即用户按下key时，

CPU就会自动的将VIC0VECTADDR0的值赋给VIC0ADDRESS并跳转到这个地址去执 \*/

key2\_VICADDR **=(**int**)**key2\_isr**;**

\_\_asm\_\_**(**

/\*开中断\*/

"mrs r0,cpsr\n"

"bic r0, r0, #0x80\n"

"msr cpsr\_c, r0\n"

**:**

**:**

**);**

**}**

### gboot.lds

OUTPUT\_ARCH**(**arm**)**

ENTRY**(**\_start**)**

SECTIONS **{**

**.=**0x20008000**;**

**.=** ALIGN**(**4**);**

**.**text **:**

**{**

start.o **(.**text**)**

**\*(.**text**)**

**}**

**.=** ALIGN**(**4**);**

**.**data **:**

**{**

**\*(.**data**)**

**}**

**.=** ALIGN**(**4**);**

bss\_start **=.;**

**.**bss **:**

**{**

**\*(.**bss**)**

**}**

bss\_end **=.;**

**}**

### button.c

#define GPH0CON \*((volatile unsigned long \*)0xe0200c00)

/\*

\* K1,K2,K3,K4对应GPH0\_1、GPH0\_2、GPH0\_3、GPH0\_4

\*/

#define GPH0\_int (0xf<<(0\*4))

#define GPH1\_int (0xf<<(1\*4))

#define GPH2\_int (0xf<<(2\*4))

#define GPH4\_int (0xf<<(4\*4))

#define GPH0\_msk (0xf<<(0\*4))

#define GPH1\_msk (0xf<<(1\*4))

#define GPH2\_msk (0xf<<(2\*4))

#define GPH4\_msk (0xf<<(4\*4))

void button\_init**()**

**{**

GPH0CON **&=~(**GPH0\_msk **|** GPH1\_msk **|** GPH2\_msk **|** GPH4\_msk**);**

GPH0CON **|=** GPH0\_int **|** GPH1\_int **|** GPH2\_int **|** GPH4\_int**;**

**}**

### main.c

int gboot\_main**()**

**{**

#ifdef MMU\_ON

mmu\_init**();**

#endif

led\_init**();**

button\_init**();**

init\_irq**();**

uart\_init**();**

**while(**1**)**

getc**();**

**return**0**;**

**}**

## 串口控制台建立：mini2440

控制台框架搭建

串口控制台分类

菜单型控制台和解析型控制台

printf/scanf实现

程序结构优化

标准库的移植参考：linux/lib/xxx.c

### start.S

**.text**

.global \_start

\_start**:**

b reset

ldr pc**,** \_undifined\_instruction

ldr pc**,** \_software\_interrupt

ldr pc**,** \_prefetch\_abort

ldr pc**,** \_data\_abort

ldr pc**,** \_not\_used

ldr pc**,** \_irq

ldr pc**,** \_fiq

\_undifined\_instruction**:** .word undifined\_instruction

\_software\_interrupt**:** .word software\_interrupt

\_prefetch\_abort**:** .word prefetch\_abort

\_data\_abort**:** .word data\_abort

\_not\_used**:** .word not\_used

\_irq**:** .word irq

\_fiq**:** .word reset

undifined\_instruction**:**

**nop**

software\_interrupt**:**

**nop**

prefetch\_abort**:**

**nop**

data\_abort**:**

**nop**

not\_used**:**

**nop**

irq**:**

**sub** lr**,** lr **,**#4

stmfd **sp**! **,** {r0**-**r12**,**lr} **/\***保存环境**\*/**

**bl** handle\_int

ldmfd **sp**!**,** {r0**-**r12**,** pc}**^/\***恢复环境**\*/**

fiq**:**

**nop**

reset**:**

**bl** set\_svc

**bl** disable\_watchdog

**bl** disable\_interrupt

**bl** disable\_mmu

**bl** init\_clock

**bl** init\_sdram

**bl** init\_stack

**bl** nandflash\_init

**bl** copy\_to\_ram

**bl** clean\_bss

ldr pc**,=**gboot\_main

@ **bl** light\_led

set\_svc**:**

mrs r0**,** cpsr

bic r0**,** r0**,**#0x1f

orr r0**,** r0**,**#0xd3

msr cpsr**,** r0

**mov** pc**,** lr

#define pWTCON 0x53000000

disable\_watchdog**:**

ldr r0**,=**pWTCON

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_interrupt**:**

mvn r1**,** #0x0

ldr r0**,=**0x4a000008

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_mmu**:**

mcr p15**,**0**,**r0**,**c7**,**c7**,**0

mrc p15**,**0**,**r0**,**c1**,**c0**,**0

bic r0**,** r0**,** #0x00000007

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

**mov** pc**,** lr

#define CLKDIVN 0x4c000014

#define MPLLCON 0x4c000004

#define MPLL\_405MHZ **((**127**<<**12**)|(**2**<<**4**)|(**1**<<**0**))**

init\_clock**:**

ldr r0**,=**CLKDIVN

**mov** r1**,** #0x5

**str** r1**,[**r0**]**

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

orr r0**,**r0**,**#0xc0000000

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

ldr r0**,=**MPLLCON

ldr r1**,=**MPLL\_405MHZ

**str** r1**,[**r0**]**

**mov** pc**,** lr

#define mem\_contrl 0x48000000

init\_sdram**:**

ldr r0**,=**mem\_contrl

**add** r3**,** r0**,** #4**\***13

adrl r1**,** mem\_data

0**:**

ldr r2**,[**r1**],** #4

**str** r2**,[**r0**],** #4

**cmp** r0**,** r3

bne 0b

**mov** pc**,** lr

copy\_to\_ram**:**

**mov** r0**,**#0x00

ldr r1**,=**\_start

ldr r2**,=**bss\_end

**sub** r2**,**r2**,**r1

**mov** ip**,**lr

**bl** nand\_to\_ram

**mov** lr**,**ip

**mov** pc**,**lr

init\_stack**:**

msr cpsr\_c**,** #0xd2

ldr **sp,=**0x33000000**//**初始化r13\_irq

msr cpsr\_c**,** #0xd3

ldr **sp,=**0x34000000**//**初始化R13\_svc

**mov** pc **,**lr

clean\_bss**:**

ldr r0**,=**bss\_start

ldr r1**,=**bss\_end

**cmp** r0**,** r1

moveq pc**,** lr

clean\_loop**:**

**mov** r2**,** #0

**str** r2**,[**r0**],** #4

**cmp** r0**,** r1

bne clean\_loop

**mov** pc**,** lr

mem\_data**:**

.long 0x22000000

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00000700

.long 0x00018001

.long 0x00018001

.long 0x008c04f5

.long 0x000000b1

.long 0x00000030

.long 0x00000030

#define GPBCON 0x56000010

#define GPBDAT 0x56000014

light\_led**:**

ldr r0**,=**GPBCON

**mov** r1**,** #0x400

**str** r1**,[**r0**]**

ldr r0**,=**GPBDAT

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

**mov** pc**,** lr

### Makefile

OBJS**:=** start.o main.o dev/dev.o lib/lib.o

CFLAGS**:=** -nostdinc -fno-builtin -I$(shell pwd)/include

export CFLAGS

gboot.bin**:** gboot.elf

arm-linux-objcopy -O binary gboot.elf gboot.bin

gboot.elf**:**$(OBJS)

arm-linux-ld -Tgboot.lds -o gboot.elf $^

%.o**:** %.S

arm-linux-gcc -g -c $^

%.o**:** %.c

arm-linux-gcc $(CFLAGS) -c $^

lib/lib.o**:**

make -C lib all

dev/dev.o**:**

make -C dev all

.PHONY**:** clean

clean**:**

rm \*.o \*.elf \*.bin

make -C lib clean

make -C dev clean

### main.c

int gboot\_main**()**

**{**

int num**;**

#ifdef MMU\_ON

mmu\_init**();**

#endif

led\_init**();**

button\_init**();**

init\_irq**();**

uart\_init**();**

**while(**1**)**

**{**

printf**(**"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r"**);**

printf**(**"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*GBOOT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r"**);**

printf**(**"1:Download Linux Kernel from TFTP Server!\n\r"**);**

printf**(**"2:Boot Linux from RAM!\n\r"**);**

printf**(**"3:Boor Linux from Nand Flash!\n\r"**);**

printf**(**"\n Plese Select:"**);**

scanf**(**"%d"**,&**num**);**

**switch(**num**)**

**{**

**case**1**:**

//tftp\_load();

**break;**

**case**2**:**

//boot\_linux\_ram();

**break;**

**case**3**:**

//boot\_linux\_nand();

**break;**

**default:**

printf**(**"Error: wrong selection!\n\r"**);**

**break;**

**}**

**}**

**return**0**;**

**}**

### gboot.lds

OUTPUT\_ARCH**(**arm**)**

ENTRY**(**\_start**)**

SECTIONS **{**

**.=**0x30008000**;**

**.=** ALIGN**(**4**);**

**.**text **:**

**{**

start.o **(.**text**)**

**\*(.**text**)**

**}**

**.=** ALIGN**(**4**);**

**.**data **:**

**{**

**\*(.**data**)**

**}**

**.=** ALIGN**(**4**);**

bss\_start **=.;**

**.**bss **:**

**{**

**\*(.**bss**)**

**}**

bss\_end **=.;**

**}**

### dev/uart.c

#define GPHCON (\*(volatile unsigned long\*)0x56000070)

#define ULCON0 (\*(volatile unsigned long\*)0x50000000)

#define UCON0 (\*(volatile unsigned long\*)0x50000004)

#define UBRDIV0 (\*(volatile unsigned long\*)0x50000028)

#define UTRSTAT0 (\*(volatile unsigned long\*)0x50000010)

#define UTXH0 (\*(volatile unsigned long\*)0x50000020)

#define URXH0 (\*(volatile unsigned long\*)0x50000024)

#define PCLK 50000000

#define BAUD 115200

void uart\_init**()**

**{**

//1.配置引脚功能

GPHCON **&=~(**0xf**<<**4**);**

GPHCON **|=(**0xa**<<**4**);**

//2.1 设置数据格式

ULCON0 **=**0b11**;**

//2.2 设置工作模式

UCON0 **=**0b0101**;**

//3. 设置波特率

UBRDIV0 **=(**int**)(**PCLK**/(**BAUD**\***16**)-**1**);**

**}**

void putc**(**unsignedchar ch**)**

**{**

**while(!(**UTRSTAT0 **&(**1**<<**1**)));**

UTXH0 **=** ch**;**

**}**

unsignedchar getc**(**void**)**

**{**

unsignedchar ret**;**

**while(!(**UTRSTAT0 **&(**1**<<**0**)));**

// 取数据

ret **=** URXH0**;**

**if((**ret **==**0x0d**)||(**ret **==**0x0a**))**

**{**

putc**(**0x0d**);**

putc**(**0x0a**);**

**}**

**else**

putc**(**ret**);**

**return** ret**;**

**}**

### dev/flash.c

#define NFCONF (\*(volatile unsigned long\*)0x4E000000)

#define NFCONT (\*(volatile unsigned long\*)0x4E000004)

#define NFCMD (\*(volatile unsigned char\*)0x4E000008)

#define NFADDR (\*(volatile unsigned char\*)0x4E00000C)

#define NFDATA (\*(volatile unsigned char\*)0x4E000010)

#define NFSTAT (\*(volatile unsigned char\*)0x4E000020)

#define TACLS 1

#define TWRPH0 2

#define TWRPH1 1

void select\_chip**()**

**{**

NFCONT **&=~(**1**<<**1**);**

**}**

void deselect\_chip**()**

**{**

NFCONT **|=(**1**<<**1**);**

**}**

void clear\_RnB**()**

**{**

NFSTAT **|=(**1**<<**2**);**

**}**

void send\_cmd**(**unsigned cmd**)**

**{**

NFCMD **=** cmd**;**

**}**

void send\_addr**(**unsigned addr**)**

**{**

NFADDR **=** addr**;**

**}**

void wait\_RnB**()**

**{**

**while(!(**NFSTAT**&(**1**<<**2**)))**

**{**

**;**

**}**

**}**

void nand\_reset**()**

**{**

//选中flash

select\_chip**();**

//清除RnB

clear\_RnB**();**

//发送0xff命令

send\_cmd**(**0xff**);**

//等待RnB

wait\_RnB**();**

//取消选中flash

deselect\_chip**();**

**}**

void nandflash\_init**()**

**{**

//初始化NFCONF

NFCONF **=(**TACLS**<<**12**)|(**TWRPH0**<<**8**)|(**TWRPH1**<<**4**);**

//初始化NFCONT

NFCONT **=(**1**<<**0**)|(**1**<<**1**);**

//复位

nand\_reset**();**

**}**

void NF\_PageRead**(**unsignedlong addr**,**unsignedchar**\*** buff**)**

**{**

int i**;**

//选中nandflash芯片

select\_chip**();**

//清除RnB

clear\_RnB**();**

//发送命令0x00

send\_cmd**(**0x00**);**

//发送列地址

send\_addr**(**0x00**);**

send\_addr**(**0x00**);**

//发送行地址

send\_addr**(**addr**&**0xff**);**

send\_addr**((**addr**>>**8**)&**0xff**);**

send\_addr**((**addr**>>**16**)&**0xff**);**

//发送命令0x30

send\_cmd**(**0x30**);**

//等待RnB

wait\_RnB**();**

//读取数据

**for(**i**=**0**;**i**<**2048**;**i**++)**

**{**

buff**[**i**]=** NFDATA**;**

**}**

//取消选中nandflash芯片

deselect\_chip**();**

**}**

void nand\_to\_ram**(**unsignedlong start\_addr**,**unsignedchar**\*** sdram\_addr**,**int size**)**

**{**

int i**;**

**for(** i**=(**start\_addr **>>**11**);** size**>**0**;)**

**{**

NF\_PageRead**(**i**,**sdram\_addr**);**

size **-=**2048**;**

sdram\_addr **+=**2048**;**

i**++;**

**}**

**}**

int NF\_Erase**(**unsignedlong addr**)**

**{**

int ret**;**

//选中flash芯片

select\_chip**();**

//清除RnB

clear\_RnB**();**

//发送命令0x60

send\_cmd**(**0x60**);**

//发送行地址

send\_addr**(**addr**&**0xff**);**

send\_addr**((**addr**>>**8**)&**0xff**);**

send\_addr**((**addr**>>**16**)&**0xff**);**

//发送命令D0

send\_cmd**(**0xD0**);**

//等待RnB

wait\_RnB**();**

//发送命令0x70

send\_cmd**(**0x70**);**

//读取擦除结果

ret **=** NFDATA**;**

//取消选中flash芯片

deselect\_chip**();**

**return** ret**;**

**}**

int NF\_WritePage**(**unsignedlong addr**,**unsignedchar**\***buff**)**

**{**

unsignedint i**,**ret **=**0**;**

//选中nandflash

select\_chip**();**

//清除RnB

clear\_RnB**();**

//发送0x80命令

send\_cmd**(**0x80**);**

//发送2个列地址

send\_addr**(**0x00**);**

send\_addr**(**0x00**);**

//发送3个行地址

send\_addr**(**addr**&**0xff**);**

send\_addr**((**addr**>>**8**)&**0xff**);**

send\_addr**((**addr**>>**16**)&**0xff**);**

//发送数据

**for(**i**=**0**;**i**<**2048**;**i**++)**

**{**

NFDATA **=** buff**[**i**];**

**}**

//发送0x10命令

send\_cmd**(**0x10**);**

//等待RnB

wait\_RnB**();**

//发送0x70命令

send\_cmd**(**0x70**);**

//读取写入结果

ret **=** NFDATA**;**

//关闭nandflash

deselect\_chip**();**

**return** ret**;**

**}**

### dev/mmu.c

/\*

\* 用于段描述符的一些宏定义

\*/

#define MMU\_FULL\_ACCESS (3 << 10) /\* 访问权限 \*/

#define MMU\_DOMAIN (0 << 5) /\* 属于哪个域 \*/

#define MMU\_SPECIAL (1 << 4) /\* 必须是1 \*/

#define MMU\_CACHEABLE (1 << 3) /\* cacheable \*/

#define MMU\_BUFFERABLE (1 << 2) /\* bufferable \*/

#define MMU\_SECTION (2 << 0) /\* 段描述符 \*/

#define MMU\_SECDESC (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_SECTION)

#define MMU\_SECDESC\_WB (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_CACHEABLE | MMU\_BUFFERABLE | MMU\_SECTION)

void create\_page\_table**(**void**)**

**{**

unsignedlong**\***ttb **=(**unsignedlong**\*)**0x31000000**;**

unsignedlong vaddr**,** paddr**;**

vaddr **=**0x00000000**;**

paddr **=**0x30000000**;**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xfff00000**)|** MMU\_SECDESC\_WB**;**

vaddr **=**0x56000000**;**

paddr **=**0x56000000**;**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC**;**

vaddr **=**0x4A000000**;**

paddr **=**0x4A000000**;**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xfff00000**)|** MMU\_SECDESC**;**

vaddr **=**0x30000000**;**

paddr **=**0x30000000**;**

**while(**vaddr **<**0x34000000**)**

**{**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC\_WB**;**

vaddr **+=**0x100000**;**

paddr **+=**0x100000**;**

**}**

**}**

void mmu\_enable**()**

**{**

\_\_asm\_\_**(**

/\*设置TTB\*/

"ldr r0, =0x31000000\n"

"mcr p15, 0, r0, c2, c0, 0\n"

/\*不进行权限检查\*/

"mvn r0, #0\n"

"mcr p15, 0, r0, c3, c0, 0\n"

/\*使能MMU\*/

"mrc p15, 0, r0, c1, c0, 0\n"

"orr r0, r0, #0x0001\n"

"mcr p15, 0, r0, c1, c0, 0\n"

**:**

**:**

**);**

**}**

void mmu\_init**()**

**{**

create\_page\_table**();**

mmu\_enable**();**

**}**

### dev/led.c

#define GPBCON (volatile unsigned long\*)0x56000010

#define GPBDAT (volatile unsigned long\*)0x56000014

void led\_init**()**

**{**

**\*(**GPBCON**)=**0x400**;**

**}**

void led\_off**()**

**{**

**\*(**GPBDAT**)=**0x7ff**;**

**}**

void led\_on**()**

**{**

**\*(**GPBDAT**)=**0x0**;**

**}**

### dev/interrupt.c

/\*interrupt registes\*/

#define SRCPND (volatile unsigned long \*)0x4A000000

#define INTMOD (volatile unsigned long \*)0x4A000004

#define INTMSK (volatile unsigned long \*)0x4A000008

#define PRIORITY (volatile unsigned long \*)0x4A00000c

#define INTPND (volatile unsigned long \*)0x4A000010

#define INTOFFSET (volatile unsigned long \*)0x4A000014

#define SUBSRCPND (volatile unsigned long \*)0x4A000018

#define INTSUBMSK (volatile unsigned long \*)0x4A00001c

#define EINTMASK (volatile unsigned long \*)0x560000a4

#define EINTPEND (volatile unsigned long \*)0x560000a8

void init\_irq**()**

**{**

// 在EINTMASK寄存器中使能它们

**\*(**EINTMASK**)&=(~(**1**<<**8**))&(~(**1**<<**11**))&(~(**1**<<**13**))&(~(**1**<<**14**));**

// EINT8\_23使能

**\*(**INTMSK**)&=~(**1**<<**5**);**

\_\_asm\_\_**(**

/\*开中断\*/

"mrs r0,cpsr\n"

"bic r0, r0, #0x80\n"

"msr cpsr\_c, r0\n"

**:**

**:**

**);**

**}**

void handle\_int**()**

**{**

/\*读取产生中断的源\*/

unsignedlong value **=\*(**EINTPEND**)&((**1**<<**8**)|(**1**<<**11**)|(**1**<<**13**)|(**1**<<**14**));**

**switch(**value**)**

**{**

**case(**1**<<**8**):**//K1

led\_on**();**

**break;**

**case(**1**<<**11**):**//K2

led\_off**();**

**break;**

**case(**1**<<**13**):**//K3

led\_on**();**

**break;**

**case(**1**<<**14**):**//K4

led\_off**();**

**break;**

**default:**

**break;**

**}**

/\* 中断清除 \*/

**\*(**EINTPEND**)=** value**;**

**\*(**SRCPND**)=**1**<<**5**;**

**\*(**INTPND**)=**1**<<**5**;**

**}**

### dev/button.c

#define GPGCON (volatile unsigned long \*)0x56000060

/\*

\* K1,K2,K3,K4对应GPG0、GPG3、GPG5、GPG6

\*/

#define GPG0\_int (0x2<<(0\*2))

#define GPG3\_int (0x2<<(3\*2))

#define GPG5\_int (0x2<<(5\*2))

#define GPG6\_int (0x2<<(6\*2))

#define GPG0\_msk (3<<(0\*2))

#define GPG3\_msk (3<<(3\*2))

#define GPG5\_msk (3<<(5\*2))

#define GPG6\_msk (3<<(6\*2))

void button\_init**()**

**{**

**\*(**GPGCON**)&=~(**GPG0\_msk **|** GPG3\_msk **|** GPG5\_msk **|** GPG6\_msk**);**

**\*(**GPGCON**)|=** GPG0\_int **|** GPG3\_int **|** GPG5\_int **|** GPG6\_int**;**

**}**

### dev/Makefile

objs**:=** button.o interrupt.o led.o mmu.o nandflash.o uart.o

all**:**$(objs)

arm-linux-ld -r -o dev.o $^

%.o**:** %.c

arm-linux-gcc ${CFLAGS} -c $^

%.o**:** %.S

arm-linux-gcc ${CFLAGS} -c $^

clean**:**

rm -f \*.o

### lib/printf.c

#include "vsprintf.h"

unsignedchar outbuf**[**1024**];**

unsignedchar inbuf**[**1024**];**

int printf**(**constchar**\*** fmt**,...)**

**{**

unsignedint i**;**

va\_list args**;**

//1.将变参转化为字符串

va\_start**(**args**,**fmt**);**//fmt转化为变参列表

vsprintf**((**char**\*)**outbuf**,**fmt**,**args**);**// 变参列表转化为字符串

va\_end**();**//转化结束

//2.打印字符到串口

**for(**i**=**0**;**i**<**strlen**((**constchar**\*)**outbuf**);**i**++)**

**{**

putc**(**outbuf**[**i**]);**

**}**

**return** i**;**

**}**

int scanf**(**constchar**\*** fmt**,...)**

**{**

unsignedchar c**;**

int i **=**0**;**

va\_list args**;**

//1. 获取输入的字符串

**while(**1**)**

**{**

c **=** getc**();**

**if((**c**==**0x0d**)||(**c**==**0x0a**))**

**{**

inbuf**[**i**]=**'\n'**;**

**break;**

**}**

**else**

**{**

inbuf**[**i**++]=** c**;**

**}**

**}**

//2. 格式转化

va\_start**(**args**,** fmt**);**

vsscanf**((**char**\*)**inbuf**,**fmt**,**args**);**

va\_end**(**args**);**

**return** i**;**

**}**

include/ctype.h

include/gcclib.h

include/kernel.h

include/string.h

include/types.h

lib/printf.c

lib/printf.h

lib/ctype.c

lib/div64.S

lib/div64.h

lib/lib1funcs.S

lib/muldi3.c

lib/string.c

lib/vsprintf.c

lib/vsprintf.h

lib/Makefile

## 串口控制台建立：OK6410

### start.S

**.text**

.global \_start

\_start**:**

b reset

ldr pc**,** \_undifined\_instruction

ldr pc**,** \_software\_interrupt

ldr pc**,** \_prefetch\_abort

ldr pc**,** \_data\_abort

ldr pc**,** \_not\_used

ldr pc**,** \_irq

ldr pc**,** \_fiq

\_undifined\_instruction**:** .word undifined\_instruction

\_software\_interrupt**:** .word software\_interrupt

\_prefetch\_abort**:** .word prefetch\_abort

\_data\_abort**:** .word data\_abort

\_not\_used**:** .word not\_used

\_irq**:** .word irq

\_fiq**:** .word reset

undifined\_instruction**:**

**nop**

software\_interrupt**:**

**nop**

prefetch\_abort**:**

**nop**

data\_abort**:**

**nop**

not\_used**:**

**nop**

irq**:**

**nop**

fiq**:**

**nop**

reset**:**

**bl** set\_svc

**bl** set\_peri\_port

**bl** disable\_watchdog

**bl** disable\_interrupt

**bl** disable\_mmu

**bl** init\_clock

**bl** mem\_init

**bl** init\_stack

**bl** nand\_init

**bl** copy\_to\_ram

**bl** clean\_bss

ldr pc**,=**gboot\_main

@ **bl** light\_led

set\_peri\_port**:**

ldr r0**,=**0x70000000

orr r0**,** r0**,** #0x13

mcr p15**,**0**,**r0**,**c15**,**c2**,**4

**mov** pc**,** lr

set\_svc**:**

mrs r0**,** cpsr

bic r0**,** r0**,**#0x1f

orr r0**,** r0**,**#0xd3

msr cpsr**,** r0

**mov** pc**,** lr

#define pWTCON 0x7e004000

disable\_watchdog**:**

ldr r0**,=**pWTCON

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_interrupt**:**

mvn r1**,**#0x0

ldr r0**,=**0x71200014

**str** r1**,[**r0**]**

ldr r0**,=**0x71300014

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_mmu**:**

mcr p15**,**0**,**r0**,**c7**,**c7**,**0

mrc p15**,**0**,**r0**,**c1**,**c0**,**0

bic r0**,** r0**,** #0x00000007

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

**mov** pc**,** lr

#define CLK\_DIV0 0x7e00f020

#define OTHERS 0x7e00f900

#define MPLL\_CON 0x7e00f010

#define APLL\_CON 0x7e00f00c

#define CLK\_SRC 0x7e00f01c

#define DIV\_VAL **((**0x0**<<**0**)|(**0x1**<<**9**)|(**0x1**<<**8**)|(**0x3**<<**12**))**

#define PLL\_VAL **((**1**<<**31**)|(**266**<<**16**)|(**3**<<**8**)|(**1**<<**0**))**

init\_clock**:**

ldr r0**,=**CLK\_DIV0

ldr r1**,=**DIV\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**OTHERS

ldr r1**,[**r0**]**

bic r1**,**r1**,**#0xc0

**str** r1**,[**r0**]**

ldr r0**,=**APLL\_CON

ldr r1**,=**PLL\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**MPLL\_CON

ldr r1**,=**PLL\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**CLK\_SRC

**mov** r1**,** #0x3

**str** r1**,[**r0**]**

**mov** pc**,** lr

copy\_to\_ram**:**

**mov** r0**,**#0

ldr r1**,=**\_start

ldr r2**,=**bss\_end

**sub** r2**,**r2**,**r1

**mov** ip**,**lr

**bl** nand\_to\_ram

**mov** lr**,**ip

**mov** pc**,**lr

init\_stack**:**

msr cpsr\_c**,** #0xd2

ldr **sp,=**0x53000000**//**初始化r13\_irq

msr cpsr\_c**,** #0xd3

ldr **sp,=**0x54000000**//**初始化R13\_svc

**mov** pc **,**lr

clean\_bss**:**

ldr r0**,=**bss\_start

ldr r1**,=**bss\_end

**cmp** r0**,** r1

moveq pc**,** lr

clean\_loop**:**

**mov** r2**,** #0

**str** r2**,[**r0**],** #4

**cmp** r0**,** r1

bne clean\_loop

**mov** pc**,** lr

light\_led**:**

ldr r1**,=**0x7F008820

ldr r0**,=**0x1111

**str** r0**,[**r1**]**

ldr r1**,=**0x7F008824

**mov** r0**,**#0xe

**str** r0**,[**r1**]**

**mov** pc**,** lr

### mem.S

**.text**

.global mem\_init

mem\_init**:**

ldr r0**,=**0x7e00f120

**mov** r1**,** #0x8

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001004 @内存控制命令寄存器

**mov** r1**,** #0x4 @根据手册得知需要先进入配置模式

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001010 @刷新寄存器地址

ldr r1**,=((**7800**/(**1000000000**/**133000000**)+**1**))** @设置刷新时间

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001014 @CAS latency寄存器

**mov** r1**,** #**(**3**<<**1**)**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001018 @t\_DQSS寄存器

**mov** r1**,** #0x1

**str** r1**,[**r0**]**

ldr r0**,=**0x7e00101c @T\_MRD寄存器

**mov** r1**,** #0x2

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001020 @t\_RAS寄存器

ldr r1**,=((**45**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001024 @t\_RC寄存器

ldr r1**,=((**68**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001028 @t\_RCD寄存器

ldr r1**,=((**23**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e00102c @t\_RFC寄存器

ldr r1**,=((**80**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001030 @t\_RP寄存器

ldr r1**,=((**23**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001034 @t\_rrd寄存器

ldr r1**,=((**15**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001038 @t\_wr寄存器

ldr r1**,=((**15**/(**1000000000**/**133000000**)+**1**))**

@ ldr r2**,[**r0**]**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e00103c @t\_wtr寄存器

**mov** r1**,** #0x07

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001040 @t\_xp寄存器

**mov** r1**,** #0x02

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001044 @t\_xsr寄存器

ldr r1**,=((**120**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001048 @t\_esr寄存器

ldr r1**,=((**120**/(**1000000000**/**133000000**)+**1**))**

**str** r1**,[**r0**]**

ldr r0**,=**0x7e00100c @内存控制配置寄存器

ldr r1**,=**0x00010012 @配置控制器

**str** r1**,[**r0**]**

ldr r0**,=**0x7e00104c @32位DRAM配置控制寄存器

ldr r1**,=**0x0b45

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001200 @片选寄存器

ldr r1**,=**0x150f8

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001304 @用户配置寄存器

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001008

ldr r1**,=**0x000c0000

**str** r1**,[**r0**]**

ldr r1**,=**0x00000000

**str** r1**,[**r0**]**

ldr r1**,=**0x00040000

**str** r1**,[**r0**]**

ldr r1**,=**0x000a0000

**str** r1**,[**r0**]**

ldr r1**,=**0x00080032

**str** r1**,[**r0**]**

ldr r0**,=**0x7e001004

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

check\_dmc1\_ready**:**

ldr r0**,=**0x7e001000

ldr r1**,[**r0**]**

**mov** r2**,** #0x3

**and** r1**,** r1**,** r2

**cmp** r1**,** #0x1

bne check\_dmc1\_ready

**nop**

**mov** pc**,** lr

### Makefile

OBJS**:=** start.o main.o mem.o dev/dev.o lib/lib.o

CFLAGS**:=** -fno-builtin -I$(shell pwd)/include

export CFLAGS

gboot.bin**:** gboot.elf

arm-linux-objcopy -O binary gboot.elf gboot.bin

gboot.elf**:**$(OBJS)

arm-linux-ld -Tgboot.lds -o gboot.elf $^

%.o**:** %.S

arm-linux-gcc -g -c $^

%.o**:** %.c

arm-linux-gcc -g -fno-builtin -c $^

lib/lib.o**:**

make -C lib all

dev/dev.o**:**

make -C dev all

.PHONY**:** clean

clean**:**

rm \*.o \*.elf \*.bin

make -C lib clean

make -C dev clean

### main.c

//char \* buf="Hello World ->\n";

int gboot\_main**()**

**{**

int num**;**

//unsigned char buf[1024\*4];

#ifdef MMU\_ON

mmu\_init**();**

#endif

uart\_init**();**

led\_init**();**

button\_init**();**

init\_irq**();**

led\_on**();**

/\*

NF\_Erase(128\*1+1);

buf[0] = 100;

NF\_WritePage(128\*1+1,buf);

buf[0] = 10;

NF\_PageRead(128\*1+1,buf);

if( buf[0] == 100 )

led\_on();

\*/

//putc(0x0d);

//putc(0x0a);

//putc('H');

//printf(buf);

//printf("Hello GBOOT!\n");

**while(**1**)**

**{**

//getc();

printf**(**"\n\r\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r"**);**

printf**(**"\n\r\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*GBOOT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r"**);**

printf**(**"1:Download Linux Kernel from TFTP Server!\n\r"**);**

printf**(**"2:Boot Linux from RAM!\n\r"**);**

printf**(**"3:Boor Linux from Nand Flash!\n\r"**);**

printf**(**"\n Plese Select:"**);**

scanf**(**"%d"**,&**num**);**

**switch(**num**)**

**{**

**case**1**:**

//tftp\_load();

**break;**

**case**2**:**

//boot\_linux\_ram();

**break;**

**case**3**:**

//boot\_linux\_nand();

**break;**

**default:**

printf**(**"Error: wrong selection!\n\r"**);**

**break;**

**}**

**}**

**return**0**;**

**}**

### gboot.lds

OUTPUT\_ARCH**(**arm**)**

ENTRY**(**\_start**)**

SECTIONS **{**

**.=**0x50008000**;**

**.=** ALIGN**(**4**);**

**.**text **:**

**{**

start.o **(.**text**)**

**\*(.**text**)**

**}**

**.=** ALIGN**(**4**);**

**.**data **:**

**{**

**\*(.**data**)**

**}**

**.=** ALIGN**(**4**);**

bss\_start **=.;**

**.**bss **:**

**{**

**\*(.**bss**)**

**}**

bss\_end **=.;**

**}**

### lib/printf.c

#include "vsprintf.h"

unsignedchar outbuf**[**1024**];**

unsignedchar inbuf**[**1024**];**

int printf**(**constchar**\*** fmt**,...)**

**{**

unsignedint i**;**

va\_list args**;**

//1.将变参转化为字符串

va\_start**(**args**,**fmt**);**//fmt转化为变参列表

vsprintf**((**char**\*)**outbuf**,**fmt**,**args**);**// 变参列表转化为字符串

va\_end**();**//转化结束

//2.打印字符到串口

**for(**i**=**0**;**i**<**strlen**((**constchar**\*)**outbuf**);**i**++)**

**{**

putc**(**outbuf**[**i**]);**

**}**

**return** i**;**

**}**

int scanf**(**constchar**\*** fmt**,...)**

**{**

unsignedchar c**;**

int i **=**0**;**

va\_list args**;**

//1. 获取输入的字符串

**while(**1**)**

**{**

c **=** getc**();**

**if((**c**==**0x0d**)||(**c**==**0x0a**))**

**{**

inbuf**[**i**]=**'\n'**;**

**break;**

**}**

**else**

**{**

inbuf**[**i**++]=** c**;**

**}**

**}**

//2. 格式转化

va\_start**(**args**,** fmt**);**

vsscanf**((**char**\*)**inbuf**,**fmt**,**args**);**

va\_end**(**args**);**

**return** i**;**

**}**

include/ctype.h

include/gcclib.h

include/kernel.h

include/string.h

include/types.h

lib/printf.c

lib/printf.h

lib/ctype.c

lib/div64.S

lib/div64.h

lib/lib1funcs.S

lib/muldi3.c

lib/string.c

lib/vsprintf.c

lib/vsprintf.h

lib/Makefile

### lib/Makefile

objs**:=** div64.o lib1funcs.o ctype.o muldi3.o printf.o string.o vsprintf.o

all**:**$(objs)

arm-linux-ld -r -o lib.o $^

%.o**:** %.c

arm-linux-gcc ${CFLAGS} -c $^

%.o**:** %.S

arm-linux-gcc ${CFLAGS} -c $^

clean**:**

rm -f \*.o

include/ctype.h

include/gcclib.h

include/kernel.h

include/string.h

include/types.h

lib/printf.c

lib/printf.h

lib/ctype.c

lib/div64.S

lib/div64.h

lib/lib1funcs.S

lib/muldi3.c

lib/string.c

lib/vsprintf.c

lib/vsprintf.h

lib/Makefile

### dev/button.c

#define GPNCON (volatile unsigned long\*)0x7f008830

void button\_init**()**

**{**

**\*(**GPNCON**)=**0b10**|(**0b10**<<**10**);**//设置按键1与按键6

**}**

### dev/interrupt.c

/\*interrupt registes\*/

#define EXT\_INT\_0\_CON \*((volatile unsigned int \*)0x7f008900)

#define EXT\_INT\_0\_MASK \*((volatile unsigned int \*)0x7f008920)

#define EXT\_INT\_0\_PEND \*((volatile unsigned int \*)0x7f008924)

#define VIC0INTENABLE \*((volatile unsigned int \*)0x71200010)

#define EINT0\_VECTADDR \*((volatile unsigned int \*)0x71200100)

#define EINT5\_VECTADDR \*((volatile unsigned int \*)0x71200104)

#define VIC0ADDRESS \*((volatile unsigned int \*)0x71200f00)

#define VIC1ADDRESS \*((volatile unsigned int \*)0x71300f00)

void key1\_handle**()**

**{**

\_\_asm\_\_**(**

"sub lr, lr, #4\n"

"stmfd sp!, {r0-r12, lr}\n"

**:**

**:**

**);**

led\_on**();**

/\* 清除中断 \*/

EXT\_INT\_0\_PEND **=~**0x0**;**

VIC0ADDRESS **=**0**;**

VIC1ADDRESS **=**0**;**

\_\_asm\_\_**(**

"ldmfd sp!, {r0-r12, pc}^ \n"

**:**

**:**

**);**

**}**

void key6\_handle**()**

**{**

\_\_asm\_\_**(**

"sub lr, lr, #4\n"

"stmfd sp!, {r0-r12, lr}\n"

**:**

**:**

**);**

led\_off**();**

/\* 清除中断 \*/

EXT\_INT\_0\_PEND **=~**0x0**;**

VIC0ADDRESS **=**0**;**

VIC1ADDRESS **=**0**;**

\_\_asm\_\_**(**

"ldmfd sp!, {r0-r12, pc}^ \n"

**:**

**:**

**);**

**}**

void init\_irq**()**

**{**

EXT\_INT\_0\_CON **=(**0b010**)|(**0b010**<<**8**);**/\* 配置为下降沿触发 \*/

EXT\_INT\_0\_MASK **=**0**;**/\* 取消屏蔽外部中断 \*/

VIC0INTENABLE **|=(**0b1**)|(**0b10**);**/\* 使能外部中断\*/

EINT0\_VECTADDR **=(**int**)**key1\_handle**;**/\* 用户按下key时，CPU就会自动的将VIC0VECTADDR0的值赋给VIC0ADDRESS并跳转到这个地址去执 \*/

EINT5\_VECTADDR **=(**int**)**key6\_handle**;**

\_\_asm\_\_**(**

"mrc p15,0,r0,c1,c0,0\n"

"orr r0,r0,#(1<<24)\n"

"mcr p15,0,r0,c1,c0,0\n"

"mrs r0,cpsr\n"

"bic r0, r0, #0x80\n"

"msr cpsr\_c, r0\n"

**:**

**:**

**);**

**}**

### dev/led.c

#define GPMCON (volatile unsigned long\*)0x7f008820

#define GPMDAT (volatile unsigned long\*)0x7f008824

void led\_init**()**

**{**

**\*(**GPMCON**)=**0x1111**;**

**}**

void led\_on**()**

**{**

**\*(**GPMDAT**)=**0x0e**;**

**}**

void led\_off**()**

**{**

**\*(**GPMDAT**)=**0x0f**;**

**}**

### dev/mmu.c

/\*

\* 用于段描述符的一些宏定义

\*/

#define MMU\_FULL\_ACCESS (3 << 10) /\* 访问权限 \*/

#define MMU\_DOMAIN (0 << 5) /\* 属于哪个域 \*/

#define MMU\_SPECIAL (1 << 4) /\* 必须是1 \*/

#define MMU\_CACHEABLE (1 << 3) /\* cacheable \*/

#define MMU\_BUFFERABLE (1 << 2) /\* bufferable \*/

#define MMU\_SECTION (2) /\* 表示这是段描述符 \*/

#define MMU\_SECDESC (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_SECTION)

#define MMU\_SECDESC\_WB (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_CACHEABLE | MMU\_BUFFERABLE | MMU\_SECTION)

void create\_page\_table**(**void**)**

**{**

unsignedlong**\***ttb **=(**unsignedlong**\*)**0x50000000**;**

unsignedlong vaddr**,** paddr**;**

vaddr **=**0xA0000000**;**

paddr **=**0x7f000000**;**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC**;**

vaddr **=**0x50000000**;**

paddr **=**0x50000000**;**

**while(**vaddr **<**0x54000000**)**

**{**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC\_WB**;**

vaddr **+=**0x100000**;**

paddr **+=**0x100000**;**

**}**

**}**

void mmu\_enable**()**

**{**

\_\_asm\_\_**(**

/\*设置TTB\*/

"ldr r0, =0x50000000\n"

"mcr p15, 0, r0, c2, c0, 0\n"

/\*不进行权限检查\*/

"mvn r0, #0\n"

"mcr p15, 0, r0, c3, c0, 0\n"

/\*使能MMU\*/

"mrc p15, 0, r0, c1, c0, 0\n"

"orr r0, r0, #0x0001\n"

"mcr p15, 0, r0, c1, c0, 0\n"

**:**

**:**

**);**

**}**

void mmu\_init**()**

**{**

create\_page\_table**();**

mmu\_enable**();**

**}**

### dev/flash.c

#define NFCONF (\*((volatile unsigned long\*)0x70200000))

#define NFCONT (\*((volatile unsigned long\*)0x70200004))

#define NFCMMD (\*((volatile unsigned char\*)0x70200008))

#define NFSTAT (\*((volatile unsigned char\*)0x70200028))

#define NFADDR (\*((volatile unsigned char\*)0x7020000c))

#define NFDATA (\*((volatile unsigned char\*)0x70200010))

void select\_ship**(**void**)**

**{**

NFCONT **&=~(**1**<<**1**);**

**}**

void delselect\_ship**(**void**)**

**{**

NFCONT **|=(**1**<<**1**);**

**}**

void clean\_RnB**()**

**{**

NFSTAT **|=(**1**<<**4**);**

**}**

void nand\_cmd**(**unsignedchar cmd**)**

**{**

NFCMMD **=** cmd**;**

**}**

void wait\_RnB**(**void**)**

**{**

**while(!(**NFSTAT **&**0x1**));**

**}**

void nand\_addr**(**unsignedchar addr**)**

**{**

NFADDR **=** addr**;**

**}**

void nand\_reset**(**void**)**

**{**

/\* 选中 \*/

select\_ship**();**

/\* 清除RnB \*/

clean\_RnB**();**

/\* 发出复位信号 \*/

nand\_cmd**(**0xff**);**

/\* 等待就绪 \*/

wait\_RnB**();**

/\* 取消选中 \*/

delselect\_ship**();**

**}**

void nand\_init**(**void**)**

**{**

/\*

HCLK的频率为100MHZ，周期就为10ns

TACLS > 0 ns

TWRPH0 > 15ns

TWRPH1 > 5ns

TACLS的值 = HCLK x TACLS > 0ns

TWRPH0的值 = HCLK x (TWRPH0 + 1) > 15ns

TWRPH1的值 = HCLK x (TWRPH1 +1) > 5ns

\*/

/\* 设置时间参数 \*/

#define TACLS 1

#define TWRPH0 2

#define TWRPH1 1

NFCONF **&=~((**7**<<**12**)|(**7**<<**8**)|(**7**<<**4**));**

NFCONF **|=(**TACLS**<<**12**)|(**TWRPH0**<<**8**)|(**TWRPH1**<<**4**);**

/\* 使能 nandflash controller\*/

NFCONT **=**1**|(**1**<<**1**);**

/\* 复位 \*/

nand\_reset**();**

**}**

void NF\_PageRead**(**unsignedlong addr**,**unsignedchar**\*** buff**)**

**{**

int i**;**

/\* 选中芯片 \*/

select\_ship**();**

/\* 清除RnB \*/

clean\_RnB**();**

/\* 发出命令0x00 \*/

nand\_cmd**(**0x00**);**

/\* 发出列地址 \*/

nand\_addr**(**0x00**);**

nand\_addr**(**0x00**);**

/\* 发出行地址 \*/

nand\_addr**(**addr**&**0xff**);**

nand\_addr**((**addr **>>**8**)&(**0xff**));**

nand\_addr**((**addr **>>**16**)&(**0xff**));**

/\* 发出命令0x30 \*/

nand\_cmd**(**0x30**);**

/\* 等待就绪 \*/

wait\_RnB**();**

/\* 读数据 \*/

**for(**i **=**0**;** i**<**1024**\***4**;** i**++)**

**{**

buff**[**i**]=** NFDATA**;**

**}**

/\* 取消片选 \*/

delselect\_ship**();**

**}**

void nand\_to\_ram**(**unsignedlong start\_addr**,**unsignedchar**\*** sdram\_addr**,**int size**)**

**{**

/\* i为页号、sdram\_addr为内存中的位置、size拷贝数据的大小 \*/

int i**;**

/\*

S3C6410启动时拷贝的8K代码不是存储在Nand flash的第一页上，

而是存储在Nand flash的前4页上，每页2K，总共8K，

\*/

**for(**i **=**0**;** i **<**4**;** i**++,** sdram\_addr**+=**2048**)**

**{**

NF\_PageRead**(**i**,**sdram\_addr**);**

**}**

size **-=**1024**\***8**;**

**for(** i**=**4**;** size**>**0**;)**

**{**

NF\_PageRead**(**i**,**sdram\_addr**);**

size **-=**4096**;**

sdram\_addr **+=**4096**;**

i**++;**

**}**

**}**

int NF\_Erase**(**unsignedlong addr**)**

**{**

int ret**;**

//选中flash芯片

select\_ship**();**

//清除RnB

clean\_RnB**();**

//发送命令60

nand\_cmd**(**0x60**);**

//发送行地址（3个周期）

nand\_addr**(**addr**&**0xff**);**

nand\_addr**((**addr **>>**8**)&(**0xff**));**

nand\_addr**((**addr **>>**16**)&(**0xff**));**

//发送命令D0

nand\_cmd**(**0xD0**);**

//等待RnB

wait\_RnB**();**

//发送命令70

nand\_cmd**(**0x70**);**

//读取擦除结果

ret **=** NFDATA**;**

//取消选中flash芯片

delselect\_ship**();**

**return** ret**;**

**}**

int NF\_WritePage**(**unsignedlong addr**,**unsignedchar**\*** buff**)**

**{**

int ret**,**i**;**

//选中flash芯片

select\_ship**();**

//清除RnB

clean\_RnB**();**

//发送命令80

nand\_cmd**(**0x80**);**

//发送列地址（2个周期）

nand\_addr**(**0x00**);**

nand\_addr**(**0x00**);**

//发送行地址（3个周期）

nand\_addr**(**addr**&**0xff**);**

nand\_addr**((**addr **>>**8**)&(**0xff**));**

nand\_addr**((**addr **>>**16**)&(**0xff**));**

//写入数据

**for(**i**=**0**;**i**<**1024**\***4**;**i**++)**

**{**

NFDATA **=** buff**[**i**];**

**}**

//发送命令10

nand\_cmd**(**0x10**);**

//等待RnB

wait\_RnB**();**

//发送命令70

nand\_cmd**(**0x70**);**

//读取写入结果

ret **=** NFDATA**;**

//取消选中flash芯片

delselect\_ship**();**

**return** ret**;**

**}**

### dev/uart.c

#define ULCON0 (\*((volatile unsigned long \*)0x7F005000))

#define UCON0 (\*((volatile unsigned long \*)0x7F005004))

#define UFCON0 (\*((volatile unsigned long \*)0x7F005008))

#define UMCON0 (\*((volatile unsigned long \*)0x7F00500C))

#define UTRSTAT0 (\*((volatile unsigned long \*)0x7F005010))

#define UFSTAT0 (\*((volatile unsigned long \*)0x7F005018))

#define UTXH0 (\*((volatile unsigned char \*)0x7F005020))

#define URXH0 (\*((volatile unsigned char \*)0x7F005024))

#define UBRDIV0 (\*((volatile unsigned short \*)0x7F005028))

#define UDIVSLOT0 (\*((volatile unsigned short \*)0x7F00502C))

#define GPACON (\*((volatile unsigned long \*)0x7F008000))

#define PCLK 66500000

#define BOUD 115200

void uart\_init**(**void**)**

**{**

GPACON **&=~**0xff**;**

GPACON **|=**0x22**;**

/\* ULCON0 \*/

ULCON0 **=**0x3**;**/\* 数据位:8, 无较验, 停止位: 1, 8n1 \*/

UCON0 **=**0x5**;**/\* 使能UART发送、接收 \*/

UFCON0 **=**0x01**;**/\* FIFO ENABLE \*/

UMCON0 **=**0**;**

/\* 波特率 \*/

/\* DIV\_VAL = (PCLK / (bps x 16 ) ) - 1

\* bps = 115200

\* DIV\_VAL = (66000000 / (115200 x 16 ) ) - 1

\* = 35.08

\*/

UBRDIV0 **=(**int**)(**PCLK**/(**BOUD**\***16**)-**1**);**

/\* x/16 = 0.08

\* x = 1

\*/

UDIVSLOT0 **=**0x1**;**

**}**

void putc**(**unsignedchar c**)**

**{**

**while(**UFSTAT0 **&(**1**<<**14**));**/\* 如果TX FIFO满，等待 \*/

UTXH0 **=** c**;**/\* 写数据 \*/

**}**

unsignedchar getc**(**void**)**

**{**

unsignedchar ret**;**

**while((**UFSTAT0 **&**0x7f**)==**0**);**/\* 如果RX FIFO空，等待 \*/

ret **=** URXH0**;**/\* 取数据 \*/

**if((**ret **==**0x0d**)||(**ret **==**0x0a**))**

**{**

putc**(**0x0d**);**

putc**(**0x0a**);**

**}**

**else**

**{**

putc**(**ret**);**

**}**

**return** ret**;**

**}**

/\*

void uart\_send\_string(char \*str)

{

while(\*str)

{

putc(\*str++);

}

putc(0x0d);

putc(0x0a);

}

\*/

### dev/Makefile

objs**:=** button.o interrupt.o led.o mmu.o flash.o uart.o

all**:**$(objs)

arm-linux-ld -r -o dev.o $^

%.o**:** %.c

arm-linux-gcc ${CFLAGS} -c $^

%.o**:** %.S

arm-linux-gcc ${CFLAGS} -c $^

clean**:**

rm -f \*.o \*.bak

## 串口控制台建立：smart210

### start.S

@**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

@File**:**start.S

@

@成都国嵌嵌入式培训中心

@**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**.text**

.global \_start

\_start**:**

b reset

ldr pc**,** \_undefined\_instruction

ldr pc**,** \_software\_interrupt

ldr pc**,** \_prefetch\_abort

ldr pc**,** \_data\_abort

ldr pc**,** \_not\_used

ldr pc**,** \_irq

ldr pc**,** \_fiq

\_undefined\_instruction**:** .word undefined\_instruction

\_software\_interrupt**:** .word software\_interrupt

\_prefetch\_abort**:** .word prefetch\_abort

\_data\_abort**:** .word data\_abort

\_not\_used**:** .word not\_used

\_irq**:** .word irq

\_fiq**:** .word fiq

undefined\_instruction**:**

**nop**

software\_interrupt**:**

**nop**

prefetch\_abort**:**

**nop**

data\_abort**:**

**nop**

not\_used**:**

**nop**

irq**:**

**nop**

fiq**:**

**nop**

reset**:**

**bl** set\_svc

**bl** disable\_watchdog

**bl** disable\_interrupt

**bl** disable\_mmu

**bl** init\_clock

**bl** mem\_init

**bl** copy\_to\_ram

**bl** init\_stack

**bl** clean\_bss

ldr pc**,=**gboot\_main

@ **bl** light\_led

set\_svc**:**

mrs r0**,** cpsr

bic r0**,** r0**,** #0x1f

orr r0**,** r0**,** #0xd3

msr cpsr**,** r0

**mov** pc**,** lr

#define pWTCON 0xE2700000

disable\_watchdog**:**

ldr r0**,=**pWTCON

**mov** r1**,** #0x0

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_interrupt**:**

mvn r1**,**#0x0

ldr r0**,=**0xf2000014

**str** r1**,[**r0**]**

ldr r0**,=**0xf2100014

**str** r1**,[**r0**]**

ldr r0**,=**0xf2200014

**str** r1**,[**r0**]**

ldr r0**,=**0xf2300014

**str** r1**,[**r0**]**

**mov** pc**,** lr

disable\_mmu**:**

mcr p15**,**0**,**r0**,**c7**,**c5**,**0

mcr p15**,**0**,**r0**,**c7**,**c6**,**1

mrc p15**,**0**,**r0**,**c1**,**c0**,**0

bic r0**,** r0**,** #0x00000007

mcr p15**,**0**,**r0**,**c1**,**c0**,**0

**mov** pc**,** lr

#define CLK\_DIV0 0xe0100300

#define MPLL\_CON 0xe0100108

#define APLL\_CON 0xe0100100

#define CLK\_SRC 0xe0100200

#define DIV\_VAL **((**0x0**<<**0**)|(**4**<<**4**)|(**4**<<**8**)|(**1**<<**12**)|(**3**<<**16**)|(**1**<<**20**)|(**4**<<**24**)|(**1**<<**28**))**

#define APLL\_VAL **((**1**<<**31**)|(**3**<<**8**)|(**125**<<**16**)|(**1**<<**0**))**

#define MPLL\_VAL **((**1**<<**31**)|(**12**<<**8**)|(**667**<<**16**)|(**1**<<**0**))**

init\_clock**:**

ldr r0**,=**CLK\_DIV0

ldr r1**,=**DIV\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**APLL\_CON

ldr r1**,=**APLL\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**MPLL\_CON

ldr r1**,=**MPLL\_VAL

**str** r1**,[**r0**]**

ldr r0**,=**CLK\_SRC

ldr r1**,=**0x1111

**str** r1**,[**r0**]**

**mov** pc**,** lr

copy\_to\_ram**:**

ldr r0**,=**0xd0020010

ldr r1**,=**0x20008000

**add** r3**,** r0**,** #1024**\***4

copy\_loop**:**

ldr r2**,[**r0**],** #4

**str** r2**,[**r1**],** #4

**cmp** r0**,** r3

bne copy\_loop

**mov** pc**,** lr

init\_stack**:**

msr cpsr\_c**,** #0xd2

ldr **sp,=**0x23000000 @此处实际设置的是r13\_irq

msr cpsr\_c**,** #0xd3

ldr **sp,=**0x24000000 @此处实际设置的是r13\_svc

**mov** pc**,** lr

clean\_bss**:**

ldr r0**,=**bss\_start

ldr r1**,=**bss\_end

**cmp** r0**,** r1

moveq pc**,** lr

clean\_loop**:**

**mov** r2**,** #0

**str** r2**,[**r0**],** #4

**cmp** r0**,** r1

bne clean\_loop

**mov** pc**,** lr

light\_led**:**

ldr R0**,=**0xE0200280

**mov** R1**,**#0x1

**str** R1**,[**R0**]**

ldr R0**,=**0xE0200284

**mov** R1**,**#0x0

**str** R1**,[**R0**]**

**mov** pc**,** lr

### mem.S

#define DMC\_PHYCONTROL0 0xf0000018

#define DMC\_PHYCONTROL1 0xf000001c

#define DMC\_CONCONTROL 0xf0000000

#define DMC\_MEMCONTROL 0xf0000004

#define DMC\_MEMCONFIG0 0xf0000008

#define DMC\_MEMCONFIG1 0xf000000c

#define DMC\_PRECHCONFIG 0xf0000014

#define DMC\_TIMINGAREF 0xf0000030

#define DMC\_TIMINGROW 0xf0000034

#define DMC\_TIMINGDATA 0xf0000038

#define DMC\_TIMINGPOWER 0xf000003c

#define DMC\_PHYSTATUS 0xf0000040

#define DMC\_DIRECTCMD 0xf0000010

#define DMC\_PWRDNCONFIG 0xf0000028

#define DMC0\_MEMCONTROL 0x00202400

#define DMC0\_MEMCONFIG\_0 0x20F00313

#define DMC0\_MEMCONFIG\_1 0x00F00313

#define DMC0\_TIMINGA\_REF 0x00000618

#define DMC0\_TIMING\_ROW 0x2B34438A

#define DMC0\_TIMING\_DATA 0x24240000

#define DMC0\_TIMING\_PWR 0x0BDC0343

.globl mem\_init

mem\_init**:**

@ step 2.1

ldr r0**,=**DMC\_PHYCONTROL0

ldr r1**,=**0x00101000

**str** r1**,[**r0**]**

@ step 2.2

ldr r0**,=**DMC\_PHYCONTROL0

ldr r1**,=**0x00101002

**str** r1**,[**r0**]**

@ step 4

ldr r0**,=**DMC\_PHYCONTROL0

ldr r1**,=**0x00101003

**str** r1**,[**r0**]**

@ step 5

ldr r0**,=**DMC\_CONCONTROL

ldr r1**,=**0x0FFF1350

**str** r1**,[**r0**]**

@ step 6

ldr r0**,=**DMC\_MEMCONTROL

ldr r1**,=**DMC0\_MEMCONTROL

**str** r1**,[**r0**]**

@ step 7

ldr r0**,=**DMC\_MEMCONFIG0

ldr r1**,=**DMC0\_MEMCONFIG\_0

**str** r1**,[**r0**]**

@ step 8

ldr r0**,=**DMC\_PRECHCONFIG

ldr r1**,=**0xFF000000

**str** r1**,[**r0**]**

@ step 9.1

ldr r0**,=**DMC\_TIMINGAREF

ldr r1**,=**DMC0\_TIMINGA\_REF

**str** r1**,[**r0**]**

@ step 9.2

ldr r0**,=**DMC\_TIMINGROW

ldr r1**,=**DMC0\_TIMING\_ROW

**str** r1**,[**r0**]**

@ step 9.3

ldr r0**,=**DMC\_TIMINGDATA

ldr r1**,=**DMC0\_TIMING\_DATA

**str** r1**,[**r0**]**

@ step 9.4

ldr r0**,=**DMC\_TIMINGPOWER

ldr r1**,=**DMC0\_TIMING\_PWR

**str** r1**,[**r0**]**

@ step 11

wait\_lock**:**

ldr r0**,=**DMC\_PHYSTATUS

ldr r1**,[**r0**]**

**and** r2**,** r1**,** #0x4

**cmp** r2**,** #0x4

bne wait\_lock

@ step 14

ldr r0**,=**DMC\_DIRECTCMD

ldr r1**,=**0x07000000

**str** r1**,[**r0**]**

@ step 16

ldr r1**,=**0x01000000

**str** r1**,[**r0**]**

@ step 17

ldr r1**,=**0x00020000

**str** r1**,[**r0**]**

@ step 18

ldr r1**,=**0x00030000

**str** r1**,[**r0**]**

@ step 19

ldr r1**,=**0x00010400

**str** r1**,[**r0**]**

@ step 20

ldr r1**,=**0x00000542

**str** r1**,[**r0**]**

@ step 21

ldr r1**,=**0x01000000

**str** r1**,[**r0**]**

@ step 22.1

ldr r1**,=**0x05000000

**str** r1**,[**r0**]**

@ step 22.2

ldr r1**,=**0x05000000

**str** r1**,[**r0**]**

@ step 23

ldr r1**,=**0x00000442

**str** r1**,[**r0**]**

@ step 25.1

ldr r1**,=**0x00010780

**str** r1**,[**r0**]**

@ step 25.2

ldr r1**,=**0x00010400

**str** r1**,[**r0**]**

@ step 26**,**repeat step14**~**step25

ldr r1**,=**0x07100000

**str** r1**,[**r0**]**

ldr r1**,=**0x01100000

**str** r1**,[**r0**]**

ldr r1**,=**0x00120000

**str** r1**,[**r0**]**

ldr r1**,=**0x00130000

**str** r1**,[**r0**]**

ldr r1**,=**0x00110400

**str** r1**,[**r0**]**

ldr r1**,=**0x00100542

**str** r1**,[**r0**]**

ldr r1**,=**0x01100000

**str** r1**,[**r0**]**

ldr r1**,=**0x05100000

**str** r1**,[**r0**]**

ldr r1**,=**0x05100000

**str** r1**,[**r0**]**

ldr r1**,=**0x00100442

**str** r1**,[**r0**]**

ldr r1**,=**0x00110780

**str** r1**,[**r0**]**

ldr r1**,=**0x00110400

**str** r1**,[**r0**]**

@ step 27

ldr r0**,=**DMC\_CONCONTROL

ldr r1**,=**0x0FF02030

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_PWRDNCONFIG

ldr r1**,=**0xFFFF00FF

**str** r1**,[**r0**]**

ldr r0**,=**DMC\_CONCONTROL

ldr r1**,=**0x00202400

**str** r1**,[**r0**]**

**mov** pc**,** lr

### Makefile

OBJS**:=** start.o main.o mem.o dev/dev.o lib/lib.o

CFLAGS**:=** -fno-builtin -I$(shell pwd)/include

export CFLAGS

gboot.bin**:** gboot.elf

arm-linux-objcopy -O binary gboot.elf gboot.bin

gboot.elf**:**$(OBJS)

arm-linux-ld -Tgboot.lds -o gboot.elf $^

%.o**:** %.S

arm-linux-gcc -g -c $^

%.o**:** %.c

arm-linux-gcc $(CFLAGS) -c $^

lib/lib.o**:**

make -C lib all

dev/dev.o**:**

make -C dev all

.PHONY**:** clean

clean**:**

rm \*.o \*.elf \*.bin

make -C lib clean

make -C dev clean

### main.c

int gboot\_main**()**

**{**

int num**;**

#ifdef MMU\_ON

mmu\_init**();**

#endif

led\_init**();**

button\_init**();**

init\_irq**();**

uart\_init**();**

**while(**1**)**

**{**

printf**(**"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r"**);**

printf**(**"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*GBOOT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r"**);**

printf**(**"1:Download Linux Kernel from TFTP Server!\n\r"**);**

printf**(**"2:Boot Linux from RAM!\n\r"**);**

printf**(**"3:Boor Linux from Nand Flash!\n\r"**);**

printf**(**"\n Plese Select:"**);**

scanf**(**"%d"**,&**num**);**

**switch(**num**)**

**{**

**case**1**:**

//tftp\_load();

**break;**

**case**2**:**

//boot\_linux\_ram();

**break;**

**case**3**:**

//boot\_linux\_nand();

**break;**

**default:**

printf**(**"Error: wrong selection!\n\r"**);**

**break;**

**}**

**}**

**return**0**;**

**}**

### gboot.lds

OUTPUT\_ARCH**(**arm**)**

ENTRY**(**\_start**)**

SECTIONS **{**

**.=**0x20008000**;**

**.=** ALIGN**(**4**);**

**.**text **:**

**{**

start.o **(.**text**)**

**\*(.**text**)**

**}**

**.=** ALIGN**(**4**);**

**.**data **:**

**{**

**\*(.**data**)**

**}**

**.=** ALIGN**(**4**);**

bss\_start **=.;**

**.**bss **:**

**{**

**\*(.**bss**)**

**}**

bss\_end **=.;**

**}**

### lib/printf.c

#include "vsprintf.h"

unsignedchar outbuf**[**1024**];**

unsignedchar inbuf**[**1024**];**

int printf**(**constchar**\*** fmt**,...)**

**{**

unsignedint i**;**

va\_list args**;**

//1.将变参转化为字符串

va\_start**(**args**,**fmt**);**//fmt转化为变参列表

vsprintf**((**char**\*)**outbuf**,**fmt**,**args**);**// 变参列表转化为字符串

va\_end**();**//转化结束

//2.打印字符到串口

**for(**i**=**0**;**i**<**strlen**((**constchar**\*)**outbuf**);**i**++)**

**{**

putc**(**outbuf**[**i**]);**

**}**

**return** i**;**

**}**

int scanf**(**constchar**\*** fmt**,...)**

**{**

unsignedchar c**;**

int i **=**0**;**

va\_list args**;**

//1. 获取输入的字符串

**while(**1**)**

**{**

c **=** getc**();**

**if((**c**==**0x0d**)||(**c**==**0x0a**))**

**{**

inbuf**[**i**]=**'\n'**;**

**break;**

**}**

**else**

**{**

inbuf**[**i**++]=** c**;**

**}**

**}**

//2. 格式转化

va\_start**(**args**,** fmt**);**

vsscanf**((**char**\*)**inbuf**,**fmt**,**args**);**

va\_end**(**args**);**

**return** i**;**

**}**

include/ctype.h

include/gcclib.h

include/kernel.h

include/string.h

include/types.h

lib/printf.c

lib/printf.h

lib/ctype.c

lib/div64.S

lib/div64.h

lib/lib1funcs.S

lib/muldi3.c

lib/string.c

lib/vsprintf.c

lib/vsprintf.h

lib/Makefile

### lib/Makefile

objs**:=** div64.o lib1funcs.o ctype.o muldi3.o printf.o string.o vsprintf.o

all**:**$(objs)

arm-linux-ld -r -o lib.o $^

%.o**:** %.c

arm-linux-gcc ${CFLAGS} -c $^

%.o**:** %.S

arm-linux-gcc ${CFLAGS} -c $^

clean**:**

rm -f \*.o

### dev/button.c

#define GPH2CON \*((volatile unsigned long \*)0xe0200c40)

/\*

\* K1,K2,K3,K4对应GPH0\_1、GPH0\_2、GPH0\_3、GPH0\_4

\*/

#define GPH0\_int (0xf<<(0\*4))

#define GPH1\_int (0xf<<(1\*4))

#define GPH2\_int (0xf<<(2\*4))

#define GPH4\_int (0xf<<(4\*4))

#define GPH0\_msk (0xf<<(0\*4))

#define GPH1\_msk (0xf<<(1\*4))

#define GPH2\_msk (0xf<<(2\*4))

#define GPH4\_msk (0xf<<(4\*4))

void button\_init**()**

**{**

GPH2CON **&=~(**GPH0\_msk **|** GPH1\_msk **|** GPH2\_msk **|** GPH4\_msk**);**

GPH2CON **|=** GPH0\_int **|** GPH1\_int **|** GPH2\_int **|** GPH4\_int**;**

**}**

### dev/interrupt.c

/\*interrupt registes\*/

#define EXT\_INT\_2\_CON \*((volatile unsigned int \*)0xE0200E08)

#define EXT\_INT\_2\_MASK \*((volatile unsigned int \*)0xE0200F08)

#define VIC0INTENABLE \*((volatile unsigned int \*)0xF2000010)

#define key\_VICADDR \*((volatile unsigned int \*)0xF2000140)

#define VIC0ADDRESS \*((volatile unsigned int \*)0xF2000F00)

#define EXT\_INT\_2\_PEND \*((volatile unsigned int \*)0xE0200F48)

void key\_isr**()**

**{**

volatileunsignedint key\_code**;**

\_\_asm\_\_**(**

"sub lr, lr, #4\n"

"stmfd sp!, {r0-r12, lr}\n"

**:**

**:**

**);**

key\_code **=** EXT\_INT\_2\_PEND **&**0b11**;**

**switch(**key\_code**)**

**{**

**case**1**:**//K1

led\_on**();**

**break;**

**case**2**:**//K2

led\_off**();**

**break;**

**default:**

**break;**

**}**

/\* 清除中断 \*/

EXT\_INT\_2\_PEND **=~**0x0**;**

VIC0ADDRESS **=**0**;**

\_\_asm\_\_**(**

"ldmfd sp!, {r0-r12, pc}^ \n"

**:**

**:**

**);**

**}**

void init\_irq**()**

**{**

EXT\_INT\_2\_CON **=**0b010**|(**0b010**<<**4**);**/\*下降沿触发中断\*/

EXT\_INT\_2\_MASK **=**0**;**/\* 取消屏蔽外部中断\*/

VIC0INTENABLE **=**0b1**<<**16**;**

key\_VICADDR **=(**int**)**key\_isr**;**/\*由于key1~key4属于同一中断源，所以只需设置一个地址\*/

\_\_asm\_\_**(**

/\*开中断\*/

"mrs r0,cpsr\n"

"bic r0, r0, #0x80\n"

"msr cpsr\_c, r0\n"

**:**

**:**

**);**

**}**

### dev/led.c

#define GPC0CON \*((volatile unsigned long\*)0xE0200280)

#define GPC0DAT \*((volatile unsigned long\*)0xE0200284)

void led\_init**()**

**{**

GPC0CON **=**0x1**;**

**}**

void led\_off**()**

**{**

GPC0DAT **=**0x1**;**

**}**

void led\_on**()**

**{**

GPC0DAT **=**0x0**;**

**}**

### dev/mmu.c

/\*

\* 用于段描述符的一些宏定义

\*/

#define MMU\_FULL\_ACCESS (3 << 10) /\* 访问权限 \*/

#define MMU\_DOMAIN (0 << 5) /\* 属于哪个域 \*/

#define MMU\_SPECIAL (1 << 4) /\* 必须是1 \*/

#define MMU\_CACHEABLE (1 << 3) /\* cacheable \*/

#define MMU\_BUFFERABLE (1 << 2) /\* bufferable \*/

#define MMU\_SECTION (2) /\* 表示这是段描述符 \*/

#define MMU\_SECDESC (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_SECTION)

#define MMU\_SECDESC\_WB (MMU\_FULL\_ACCESS | MMU\_DOMAIN | MMU\_SPECIAL | MMU\_CACHEABLE | MMU\_BUFFERABLE | MMU\_SECTION)

void create\_page\_table**(**void**)**

**{**

unsignedlong**\***ttb **=(**unsignedlong**\*)**0x20000000**;**

unsignedlong vaddr**,** paddr**;**

vaddr **=**0xA0000000**;**

paddr **=**0xE0200000**;**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC**;**

vaddr **=**0x20000000**;**

paddr **=**0x20000000**;**

**while(**vaddr **<**0x24000000**)**

**{**

**\*(**ttb **+(**vaddr **>>**20**))=(**paddr **&**0xFFF00000**)|** MMU\_SECDESC\_WB**;**

vaddr **+=**0x100000**;**

paddr **+=**0x100000**;**

**}**

**}**

void mmu\_enable**()**

**{**

\_\_asm\_\_**(**

/\*设置TTB\*/

"ldr r0, =0x20000000\n"

"mcr p15, 0, r0, c2, c0, 0\n"

/\*不进行权限检查\*/

"mvn r0, #0\n"

"mcr p15, 0, r0, c3, c0, 0\n"

/\*使能MMU\*/

"mrc p15, 0, r0, c1, c0, 0\n"

"orr r0, r0, #0x0001\n"

"mcr p15, 0, r0, c1, c0, 0\n"

**:**

**:**

**);**

**}**

void mmu\_init**()**

**{**

create\_page\_table**();**

mmu\_enable**();**

**}**

### dev/flash.c

### dev/uart.c

#define GPA0CON ( \*((volatile unsigned long \*)0xE0200000) )

#define GPA1CON ( \*((volatile unsigned long \*)0xE0200020) )

#define ULCON0 ( \*((volatile unsigned long \*)0xE2900000) )

#define UCON0 ( \*((volatile unsigned long \*)0xE2900004) )

#define UBRDIV0 ( \*((volatile unsigned long \*)0xE2900028) )

#define UDIVSLOT0 ( \*((volatile unsigned long \*)0xE290002C) )

#define UTRSTAT0 ( \*((volatile unsigned long \*)0xE2900010) )

#define UTXH0 ( \*((volatile unsigned long \*)0xE2900020) )

#define URXH0 ( \*((volatile unsigned long \*)0xE2900024) )

#define PCLK\_PSYS 66000000 // HCLK\_PSYS=MPLL/(HCLK\_PSYS\_RATIO+1); PCLK\_PSYS=HCLK\_PSYS/(PCLK\_PSYS\_RATIO+1); MPLL=667MHZ(参考时钟初始化那个专题)

#define BAUD 115200

void uart\_init**()**

**{**

//1.配置引脚功能

GPA0CON **=**0x22222222**;**

GPA1CON **=**0x2222**;**

//2.1 设置数据格式

ULCON0 **=**0b11**;**

//2.2 设置工作模式

UCON0 **=**0b0101**;**

//3. 设置波特率

UBRDIV0 **=(**int**)(**PCLK\_PSYS **/(**BAUD**\***16**)-**1**);**//UBRDIV0保存该公式计算后的整数部分

UDIVSLOT0 **=**0x0**;**//UDISLOT0=保存该公式计算后的小数部分\*16

**}**

void putc**(**unsignedchar ch**)**

**{**

**while(!(**UTRSTAT0 **&(**1**<<**2**)));**

UTXH0 **=** ch**;**

**}**

unsignedchar getc**(**void**)**

**{**

unsignedchar ret**;**

**while(!(**UTRSTAT0 **&(**1**<<**0**)));**

// 取数据

ret **=** URXH0**;**

**if((**ret **==**0x0d**)||(**ret **==**0x0a**))**

**{**

putc**(**0x0d**);**

putc**(**0x0a**);**

**}**

**else**

putc**(**ret**);**

**return** ret**;**

**}**

### dev/Makefile

objs**:=** button.o interrupt.o led.o mmu.o uart.o

all**:**$(objs)

arm-linux-ld -r -o dev.o $^

%.o**:** %.c

arm-linux-gcc ${CFLAGS} -c $^

%.o**:** %.S

arm-linux-gcc ${CFLAGS} -c $^

clean**:**

rm -f \*.o

## 210串口控制台故障分析

控制台窗口输出乱码。。。

可能原因：

\*\*波特率不对：putc()的测试

\*\*printf()函数：格式化 + 输出

\*\*假设printf()的输出有问题，我们先注释掉printf()的格式化代码块。

发现putc()，当传入buf[i]时候输出就为乱码了，于是我们排除putc()，最终集中到这个变量的问题上了。

# 专题05-快车道DMA

[主目录](#_目录)

## DMA核心理论

为什么需要DMA？

我们通过串口发送字符串的函数实现来讲解：

void uart\_send\_ptr (char \*str)

{

while (\*str != 0) putc(\*(str++));

}

上述函数集成到main.c中测试，然后我们来分析，数据的传送过程。源头在内存中的一个字符串，然后CPU读取出来然后传送给外设。

然而你们有没有考虑到，因为CPU一直在取和送，所以CPU是忙碌的，不能干其他事情，如果我们从内存中读取大量数据的话这个缺陷就凸显出来了。

我们于是出来一个硬件DMA，我们只需要告诉DMA从哪里取，然后送到哪里去。这样呢CPU就得到解放。比如这个数据传输需要10分钟，那么这10分钟CPU就可以空闲出来去干其他事情了。DMA(direction memory access):直接存储器访问。

DMA控制器的介绍

参考:芯片手册:第8章DMA:

有多少通道数

**OVERVIEW**

The S3C2440A supports four-channel DMA controller located between the system bus and the peripheral bus.

哪些请求源对应哪些通道

**Table 8-1. DMA Request Sources for Each Channel**

基本时序

**Figure 8-1. Basic DMA Timing Diagram**

工作模式

Demand模式(请求模式):如果DMA完成一次请求后如果request仍然有效，那么DMA就认为这是下一次DMA请求，并立即开始下一次传输。

Handshake模式(握手模式)：DMA完成一次请求后等待request信号无效，如果request无效，DMA会无效ACK两个时钟周期，再等待下一次request.

## 2440DMA程序设计

DMA控制器配置要点：

\*\*源地址

\*\*目的地址 **DMA初始化其实主要就是源和目的的初始化**

### dev/dma.c:

char \*buf = "Hello World!";

#define DISRC0 (\*(volatile unsigned long\*)0x4B000000)

#define DISRCC0 (\*(volatile unsigned long\*)0x4B000004)

#define DIDST0 (\*(volatile unsigned long\*)0x4B000008)

#define DIDSTC0 (\*(volatile unsigned long\*)0x4B00000C)

#define DCON0 (\*(volatile unsigned long\*)0x4B000010)

#define DMASKTRIG0 (\*(volatile unsigned long\*)0x4B000020)

#define UTXH0 (volatile unsigned long\*)0x50000020

void dma\_init()

{

//初始化源地址,往这个寄存器中填写源地址

DISRC0 = (unsigned int)buf;

DISRCC0 = (0<<1)| (0<<0); //源地址控制寄存器

//我们源是在内存上,

//内存是在AHB总线上还是APB总线上我们可参考product overview

//初始化目的地址

DIDST0 = UTXH0; //目的地址是串口的UTXH0

DIDSTC0 = (1<<1)| (1<<0);

DCON0 = (1<<24)| (1<<23)| (1<<22)| (12<<0);

}

void dma\_start() //使能DMA控制器

{

DMASKTRIG0 = (1<<1);

}

### dev/Makefile:

objs := button.o interrupt.o led.o mmu.o uart.o nand.o dma.o

all : $(objs)

arm-linux-ld -r -o dev.o $^

%.o : %.c

arm-linux-gcc ${CFLAGS} -c $^

%.o : %.S

arm-linux-gcc ${CFLAGS} -c $^

clean:

rm -f \*.o \*.bak

### 测试main.c:

//char \*buf = "hello world";

int gboot\_main()

{

int num;

#ifdef MMU\_ON

mmu\_init();

#endif

led\_init();

button\_init();

init\_irq();

uart\_init();

dma\_init();

dma\_start();

//uart\_send\_string(buf);

while(1)

{

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*GBOOT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");

printf("1:Download Linux Kernel from TFTP Server!\n\r");

printf("2:Boot Linux from RAM!\n\r");

printf("3:Boor Linux from Nand Flash!\n\r");

printf("\n Plese Select:");

scanf("%d",&num);

switch (num)

{

case 1:

//tftp\_load();

break;

case 2:

//boot\_linux\_ram();

break;

case 3:

//boot\_linux\_nand();

break;

default:

printf("Error: wrong selection!\n\r");

break;

}

}

return 0;

}

## 6410DMA程序设计

# 专题06-LCD亮起来

## LCD核心理论

LCD硬件体系架构

（ARM处理器-LCD控制器）---LCD驱动芯片---LCD液晶屏

什么是液晶？

液晶属于一种有机化合物，分子形状为长棒状，在不同电流作用下，分子会做有规律的旋转，这样对光线（背光灯产生）产生了一定的控制，形成一个像素，而很多像素又可构成完整图像。利用分子对光线的折射形成一个像素点，那么利用很多分子就可以形成很多像素点。

LCD液晶屏种类？

LCD（Liquid Crystal Display）。液晶显示器按驱动方式可分为STN, GF, TFT, TFD等种类，其中TFT屏在嵌入式开发中经常使用。

LCD控制器？

液晶要能正确显示图像，除了要有LCD驱动芯片，还要有LCD控制器来给LCD驱动芯片提供必要的数据和控制信号。多数ARM处理器内部都集成有LCD控制器。

**LCD控制器**------》（数据、控制、时钟）------》LCD驱动芯片（执行者）

参考：芯片手册：15. LCD Controller：

内存中的图像数据（帧缓存）------DMA------》》（LCD控制器）

LCD控制器组成部分

芯片手册: 15. LCD Controller:**Figure 15-1. LCD Controller Block Diagram**

REGBANK:17个寄存器组

TIMEGEN：时钟产生

LPC3600：

LCC3600：

LCDCDMA：VIDPRCS：VD[23:0]

LCD显示与信号

LCD显示流程

LCD屏幕由像素点构成，当我们要去显示一个图像时候，首先从第一行显示第1个点，第2个点，。。。直到第1行显示满了,然后到第2行开始继续一个点一个点的显示，直到整个屏幕最后一行的最后一个像素点显示完了，那么这才叫做一帧图像。

LCD时序信号

芯片手册: 15. LCD Controller: **Figure 15-1. LCD Controller Block Diagram**

VCLK：显示两个像素点之间的时间间隔

HSYNC：显示一行所用的时间（这些都是可控制的）

VSYNC：整个屏幕显示所用时间

VD[23:0]:LCD像素数据输出端口

VDEN：数据使能信号

VEND：行结束信号

为什么要做引脚初始化呢？

从LCD原理图模块图看出，因为CPU上的有些引脚是多功能的需要配置。

根据原理图配置这些引脚，让它呈现LCD所需要的引脚功能。

原理图:LCD接口模块

哪些时序信号需要我们初始化呢？

参考:芯片手册: 15. LCD Controller: **Figure 15-6. TFT LCD Timing Examples**

通过时序图看出哪些时序信号需要初始化：

VSPW

VBPD

LINEVAL

VFPD

VCLK

HSPW

HBPD

HOZVAL

HFPD

第2步：找到在哪些寄存器中我们找到这些时序信号。

## TQ2440LCD驱动程序设计

LCD驱动程序设计

\*\*LCD初始化

\*\*引脚初始化

\*\*时序初始化

\*\*帧缓冲初始化

\*\*分配这块帧缓存内存

\*\*告知：把这块内存地址告诉给LCD\_DMA

\*\*杂项初始化

\*\*LCD图形显示

\*\*数据显示

\*\*单色显示

### dev/lcd.c:

#define GPCCON (\*(volatile unsigned long \*)0x56000020)

#define GPDCON (\*(volatile unsigned long \*)0x56000030)

#define GPGCON (\*(volatile unsigned long \*)0x56000060)

#define LCDCON1 (\*(volatile unsigned long \*)0x4D000000)

#define LCDCON2 (\*(volatile unsigned long \*)0x4D000004)

#define LCDCON3 (\*(volatile unsigned long \*)0x4D000008)

#define LCDCON4 (\*(volatile unsigned long \*)0x4D00000C)

#define LCDCON5 (\*(volatile unsigned long \*)0x4D000010)

#define LCDSADDR1 (\*(volatile unsigned long \*)0x4D000014)

#define LCDSADDR2 (\*(volatile unsigned long \*)0x4D000018)

#define LCDSADDR3 (\*(volatile unsigned long \*)0x4D00001C)

#define TPAL (\*(volatile unsigned long \*)0x4D000050)

#define VSPW 9

#define VBPD 1

#define LINEVAL 271

#define VFPD 1

#define CLKVAL 4 //10 = 100/((CLKVAL+1)\*2)

#define HSPW 40

#define HBPD 1

#define HOZVAL 479

#define HFPD 1

unsigned short LCDBUFFER[272][480]; //帧缓存:屏幕像素行列参数(每个点为2B)

typedef unsigned int U32;

typedef unsigned short U16;

typedef unsigned char U8;

void lcd\_port\_init() //LCD引脚(端口)初始化

{

GPDCON = 0xaaaaaaaa; //配置引脚功能一般为控制寄存器

GPCCON = 0xaaaaaaaa;

}

void lcd\_control\_init() //LCD控制器组初始化

{

//时序初始化(最复杂的一项初始化)

LCDCON1 = (CLKVAL<<8)|(0x3<<5)|(0xc<<1)|(0<<0);

LCDCON2 = (VBPD<<24)|(LINEVAL<<14)|(VFPD<<6)|(VSPW);

LCDCON3 = (HBPD<<19)|(HOZVAL<<8)|(HFPD);

LCDCON4 = (HSPW);

LCDCON5 = (1<<11)|(1<<9)|(1<<8);

//\*\*帧缓冲初始化

LCDSADDR1=(((U32)LCDBUFFER>>22)<<21) | (((U32)LCDBUFFER>>1)&0x1fffff);

LCDSADDR2 = (((U32)LCDBUFFER+272\*480\*2)>>1)&0x1fffff;

LCDSADDR3 = (0<<11) | (480\*2/2);

TPAL = 0;

}

void lcd\_init() //LCD初始化主函数

{

lcd\_port\_init(); //LCD引脚初始化

lcd\_control\_init(); //LCD控制器初始化

//打开LCD电源 //打开LCD电源

GPGCON |= (0x3<<8);

LCDCON5 |= (1<<3);

LCDCON1 |= 1;

}

//像素点函数:x,y,颜色

void point(unsigned int x,unsigned int y,unsigned int color)

{

unsigned int red, green, blue;

red = (color>>19) & 0x1f;

green = (color>>10) & 0x3f;

blue = (color>>3) & 0x1f;

LCDBUFFER[y][x] = (unsigned short)((red<<11)|(green<<5)|blue);

}

//画位图

void Paint\_Bmp(U16 x0,U16 y0,U16 wide,U16 high,const U8 \*bmp)

{

U16 x,y;

U16 c;

U32 p = 0;

for( y = y0 ; y < y0+high ; y++ )

{

for( x = x0 ; x < x0+wide ; x++ )

{

c = bmp[p] | (bmp[p+1]<<8) ;

if ( ( x < 480) && ( y < 272) )

LCDBUFFER[y][x] = c ;

p = p + 2 ;

}

}

}

void clearSrc(unsigned int color) //清除源

{

TPAL = (1<<24)|(color&0xffffff);

}

extern unsigned char bmp[90200];

void lcd\_test() //LCD驱动测试函数

{

int x;

for(x=0;x<480;x++)

point(x++,150,0xff0000);

Paint\_Bmp(0,0,220,220,bmp); //bmp.c:bmp[]

clearSrc(0xff0000);

}

//http://www.114la.com/other/rgb.htm

单色显示：

方法1：设置帧缓存像素点都为一个颜色（这方法比较笨）

方法2：设置临时调色板寄存器

### dev/Makefile:

objs := button.o interrupt.o led.o mmu.o uart.o nand.o dma.o lcd.o bmp.o

all : $(objs)

arm-linux-ld -r -o dev.o $^

%.o : %.c

arm-linux-gcc ${CFLAGS} -c $^

%.o : %.S

arm-linux-gcc ${CFLAGS} -c $^

clean:

rm -f \*.o \*.bak

### 测试main.c:

//char \*buf = "hello world";

int gboot\_main()

{

int num;

#ifdef MMU\_ON

mmu\_init();

#endif

led\_init();

button\_init();

init\_irq();

uart\_init();

lcd\_init();

lcd\_test();

while(1)

{

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*GBOOT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");

printf("1:Download Linux Kernel from TFTP Server!\n\r");

printf("2:Boot Linux from RAM!\n\r");

printf("3:Boor Linux from Nand Flash!\n\r");

printf("\n Plese Select:");

scanf("%d",&num);

switch (num)

{

case 1:

//tftp\_load();

break;

case 2:

//boot\_linux\_ram();

break;

case 3:

//boot\_linux\_nand();

break;

default:

printf("Error: wrong selection!\n\r");

break;

}

}

return 0;

}

# 专题07-触摸屏显身手

## 硬件核心理论

电阻屏工作原理

当手指触摸屏幕时，两个相互绝缘的导电层在触摸点处连接，顶层的5V电压就会加压到底层的触摸点处，底层该点的电压会发生改变。

控制器检测到该点的变化后，将该点的电压进行A/D转换，得到的值与5V相比，再乘以该轴总长度即可得到触摸点靠哪一端的坐标。

电容屏工作原理

给工作面通上一个很低的电压，当用户触摸屏幕时，手指头吸收走一个很小的电流，这个电流分别从触摸屏四个角上的电极中流出，并且理论上流经这四个电极的电流与手指到四个角的距离成比例，控制器通过对这四个电流比例的精密计算，得出触摸点的位置。

## 电阻屏驱动程序设计

电阻屏驱动流程

\*\*初始化

\*\*设置AD转换时钟

\*\*中断屏蔽设置

\*\*TC

\*\*ADC

\*\*进入等待中断模式（按下）

\*\*TC中断处理（按下）

\*\*启动XY坐标自动转化

\*\*等待转化完成，获取坐标

\*\*清除按下中断TC

\*\*进入等待TC中断模式（弹起）

\*\*清除弹起中断TC

\*\*进入等待中断模式（按下）

### dev/ts.c:

### dev/Makefile:

### 测试:main.c:

# 专题08-网卡搭建新通道

[主目录](#_目录)

## 硬件核心理论

网络模型

\*\*OSI七层模型

OSI(Open System Interconnection):

物理层,数据链路层,网络层,传输层,会话层,表示层,应用层

MAC头---LLC头---IP头---TCP头---上层数据

\*\*Linux四层模型

网络接口层,IP层,传输层,应用层

网卡硬件结构:开发光盘DM9000.pdf芯片手册

\*\*硬件总体架构

\*\*MAC:数据链路层

MAC主要负责数据帧的构建、数据差错检查、传送控制等。

网络协议栈》IP包》MAC》以太网包》PHY

\*\*PHY:物理层

PHY是物理接口收发器，属于物理层，当它接收到MAC发过来的数据时，它会加上校验码，然后按照物理层的规则进行数据编码，再发送到传输介质上。

MAC》》以太网包》》PHY》》数据编码》》传输介质。

\*\*MII接口

\*\*Media Independent Interface

媒体独立接口，表明MAC一定情况下，任何类型的PHY设备都可以正常工作。

MAC》》MII》》PHY,起到统一接口的规范作用。

DM9000虽然有很多寄存器，我们能看见它们的存在，但是我们不能直接访问，给我们提供的仅2个编程接口，一个是INDEX接口，一个是数据接口。在2440上这两个接口分别是0x20000300和0x20000304。其中这两个地址数字可不简单，通过2我们可以知道片选信息，通过3我们可知道IO BASE信息，通过4我们可知道引脚信息。

DM9000工作特性

\*\*片选

从DM900模块原理图上nLAN\_CS引脚定位到处理器上的nGCS4引脚,通过芯片手册上的nGCS4得知为片选4，而片选4的起始地址是0x20000000。

\*\*IO Base:DM9000手册:关键字300H

\*\*CMD引脚

为什么我们这里DM9000驱动采用移植？

我们这里采用移植方法来设计DM9000驱动程序，移植源是uboot的DM9000驱动，移植目标是我们自己编写的gboot。以后你们进入工作以后，真正让你从零写驱动的情况很少，更多的时候是移植。gboot的编写是为了让我们更深刻去理解uboot的工作原理和各个硬件的基本原理。

移植方法细节

从硬件原理或特性来分析，既然是网卡，一个网卡最主要的功能是发送数据，其次是接收数据，另外任何一个硬件还应该有个硬件初始化。所以我们根据上面这些来找这三个函数。

DM9000驱动索引方法:DM9000驱动属于驱动，所以在uboot/drivers目录,然后根据DM9000属于网络驱动，所以我们进而找到uboot/drivers/net/，于是找到dm9000x.c和dm9000x.h

ARP协议的介绍

\*\*以太网通信格式

\*\*dMAC(6)+sMAC(6)+类型(2)+数据(IP包:46~1500)+CRC(4)s

\*\*ARP协议功能

在以太网络中，每台计算机唯一的身份证是MAC地址（物理层的地址），两台计算机要通讯，也必须知道对方的MAC地址，但是用户通常只知道对方的IP地址，这时候就可以利用ARP（地址解析协议）来向局域网中的所有计算机发送ARP请求包，收到请求包且满足条件的计算机将回复ARP应答包，告知其MAC地址。所以ARP是一种利用IP地址或MAC地址的协议。

\*\*ARP通信格式

\*\*

ETH头部+硬件类型(2)+协议(2)+硬件长度(1)+协议长度+OP+sMAC+sIP+dMAC+dIP

\*\*其中ARP包分为请求包和应答包，通过OP字段来区别,分别是OP=1,OP=2

\*\*ARP属于网络层

如何实现ARP协议？

首先要知道包格式，然后要知道它们之间的通信细节。

typedef struct eth\_hdr //以太网首部

{

u8 d\_mac[6]; //dst\_MAC(1)

u8 s\_mac[6]; //src\_MAC(1)

u16 type; //IP or others(2)

}ETH\_HDR;

typedef struct arp\_hdr //ARP首部

{

ETH\_HDR ethhdr; //以太网首部

u16 hwtype; //硬件类型(2B)

u16 protocol; //协议(2B)

u8 hwlen; //硬件长度(1B)

u8 protolen; //协议长度(1B)

u16 opcode; //OP:1(req), 2(ack)(2B)

u8 smac[6]; //src\_MAC(1Bx6)

u8 sipaddr[4]; //src\_IP(1Bx4)

u8 dmac[6]; //dst\_MAC(1Bx6)

u8 dipaddr[4]; //dst\_IP(1Bx4)

}ARP\_HDR;s

## DM9000驱动程序设计

DM9000驱动流程分析设计

\*\*硬件初始化实现

\*\*设置片选

\*\*中断初始化（我们这里采用中断方式）

\*\*复位设备

\*\*捕获DM9000

\*\*MAC初始化

\*\*填充MAC地址

\*\*激活DM9000

\*\*数据发送实现

\*\*禁止中断

\*\*写入发送数据长度

\*\*写入待发送的数据

\*\*启动发送

\*\*等待发送结束

\*\*清除发送状态

\*\*恢复中断使能

\*\*数据接收实现

\*\*判断是否产生中断,且清除中断

\*\*空读

\*\*读取状态

\*\*读取包长度

\*\*读取包数据

### dev/dm9000.c:

#include "dm9000.h"

#define DM\_ADD (\*((volatile unsigned short \*)0x20000300))

#define DM\_DAT (\*((volatile unsigned short \*)0x20000304))

#define GPFCON (\*(volatile unsigned \*)0x56000050) //Port F control

#define EXTINT0 (\*(volatile unsigned \*)0x56000088) //External interrupt control register 0

#define EINTMASK (\*(volatile unsigned \*)0x560000a4) //External interrupt mask

#define SRCPND (\*(volatile unsigned \*)0x4a000000) //Interrupt request status

#define INTPND (\*(volatile unsigned \*)0x4a000010) //Interrupt request status

#define INTMSK (\*(volatile unsigned \*)0x4a000008) //Interrupt mask control

#define EINTPEND (\*(volatile unsigned \*)0x560000a8) //External interrupt pending

#define BWSCON (\*(volatile unsigned \*)0x48000000) //Bus width & wait status

#define BANKCON4 (\*(volatile unsigned \*)0x48000014) //BANK4 control

typedef unsigned int u32;

typedef unsigned short u16;

typedef unsigned char u8;

u8 mac\_addr[6] = {9,8,7,6,5,4};

u8 buffer[1000];

void cs\_init()

{

BWSCON = BWSCON & (~(0x3<<16));

BWSCON = BWSCON |(0x1<<16);

BANKCON4=\

(0x0<<13)|(0x0<<11)|(0x7<<8)|(0x1<<6)|(0x0<<4)|(0x0<<2)|(0x0<<0);

}

void int\_init()

{

GPFCON = GPFCON &(~(0x3<<14));

GPFCON = GPFCON |(0x2<<14);

EXTINT0 = EXTINT0 & (~(0x7<<28));

EXTINT0 = EXTINT0 | (0x1<<28);

INTMSK = INTMSK &(~(1<<4));

EINTMASK = EINTMASK & (~(0x1<<7));

SRCPND = (1<<4);

INTPND = (1<<4);

}

void dm9000\_reg\_write(u16 reg,u16 data)

{

DM\_ADD = reg;

DM\_DAT = data;

}

u8 dm9000\_reg\_read(u16 reg)

{

DM\_ADD = reg;

return DM\_DAT;

}

void dm9000\_reset()

{

dm9000\_reg\_write(DM9000\_GPCR, GPCR\_GPIO0\_OUT);

dm9000\_reg\_write(DM9000\_GPR, 0);

dm9000\_reg\_write(DM9000\_NCR, (NCR\_LBK\_INT\_MAC | NCR\_RST));

dm9000\_reg\_write(DM9000\_NCR, 0);

dm9000\_reg\_write(DM9000\_NCR, (NCR\_LBK\_INT\_MAC | NCR\_RST));

dm9000\_reg\_write(DM9000\_NCR, 0);

}

void dm9000\_probe(void)

{

u32 id\_val;

id\_val = dm9000\_reg\_read(DM9000\_VIDL);

id\_val |= dm9000\_reg\_read(DM9000\_VIDH) << 8;

id\_val |= dm9000\_reg\_read(DM9000\_PIDL) << 16;

id\_val |= dm9000\_reg\_read(DM9000\_PIDH) << 24;

if (id\_val == DM9000\_ID) {

printf("dm9000 is found !\n");

return ;

} else {

printf("dm9000 is not found !\n");

return ;

}

}

void dm9000\_init()

{

u32 i;

/\*设置片选\*/

cs\_init();

/\*中断初始化\*/

int\_init();

/\*复位设备\*/

dm9000\_reset();

/\*捕获dm9000\*/

dm9000\_probe();

/\*MAC初始化\*/

/\* Program operating register, only internal phy supported \*/

dm9000\_reg\_write(DM9000\_NCR, 0x0);

/\* TX Polling clear \*/

dm9000\_reg\_write(DM9000\_TCR, 0);

/\* Less 3Kb, 200us \*/

dm9000\_reg\_write(DM9000\_BPTR, BPTR\_BPHW(3) | BPTR\_JPT\_600US);

/\* Flow Control : High/Low Water \*/

dm9000\_reg\_write(DM9000\_FCTR, FCTR\_HWOT(3) | FCTR\_LWOT(8));

/\* SH FIXME: This looks strange! Flow Control \*/

dm9000\_reg\_write(DM9000\_FCR, 0x0);

/\* Special Mode \*/

dm9000\_reg\_write(DM9000\_SMCR, 0);

/\* clear TX status \*/

dm9000\_reg\_write(DM9000\_NSR, NSR\_WAKEST | NSR\_TX2END | NSR\_TX1END);

/\* Clear interrupt status \*/

dm9000\_reg\_write(DM9000\_ISR, ISR\_ROOS | ISR\_ROS | ISR\_PTS | ISR\_PRS);

/\*填充MAC地址\*/

for (i = 0; i < 6; i++)

dm9000\_reg\_write(DM9000\_PAR+i, mac\_addr[i]);

/\*激活DM9000\*/

dm9000\_reg\_write(DM9000\_RCR, RCR\_DIS\_LONG | RCR\_DIS\_CRC | RCR\_RXEN);

/\* Enable TX/RX interrupt mask \*/

dm9000\_reg\_write(DM9000\_IMR, IMR\_PAR);

}

void dm9000\_tx(u8 \*data,u32 length)

{

u32 i;

/\*禁止中断\*/

dm9000\_reg\_write(DM9000\_IMR,0x80);

/\*写入发送数据的长度\*/

dm9000\_reg\_write(DM9000\_TXPLL, length & 0xff);

dm9000\_reg\_write(DM9000\_TXPLH, (length >> 8) & 0xff);

/\*写入待发送的数据\*/

DM\_ADD = DM9000\_MWCMD;

for(i=0;i<length;i+=2)

{

DM\_DAT = data[i] | (data[i+1]<<8);

}

/\*启动发送\*/

dm9000\_reg\_write(DM9000\_TCR, TCR\_TXREQ);

/\*等待发送结束\*/

while(1)

{

u8 status;

status = dm9000\_reg\_read(DM9000\_TCR);

if((status&0x01)==0x00)

break;

}

/\*清除发送状态\*/

dm9000\_reg\_write(DM9000\_NSR,0x2c);

/\*恢复中断使能\*/

dm9000\_reg\_write(DM9000\_IMR,0x81);

}

#define PTK\_MAX\_LEN 1522

u32 dm9000\_rx(u8 \*data)

{

u8 status,len;

u16 tmp;

u32 i;

/\*判断是否产生中断，且清除\*/

if(dm9000\_reg\_read(DM9000\_ISR) & 0x01)

dm9000\_reg\_write(DM9000\_ISR,0x01);

else

return 0;

/\*空读\*/

dm9000\_reg\_read(DM9000\_MRCMDX);

/\*读取状态\*/

status = dm9000\_reg\_read(DM9000\_MRCMD);

/\*读取包长度\*/

len = DM\_DAT;

/\*读取包数据\*/

if(len<PTK\_MAX\_LEN)

{

for(i=0;i<len;i+=2)

{

tmp = DM\_DAT;

data[i] = tmp & 0x0ff;

data[i+1] = (tmp>>8)&0x0ff;

}

}

}

void int\_issue()

{

u32 i;

i = dm9000\_rx(buffer);

SRCPND = (1<<4);

INTPND = (1<<4);

EINTPEND |= 1<<7;

}

### dev/arp.c:

ARP请求包和应答包运作流程

\*\*ARP请求包

\*\*构建ARP请求包

\*\*调用DM9000驱动发送函数来发送请求包

\*\*ARP应答包

typedef unsigned int u32;

typedef unsigned short u16;

typedef unsigned char u8;

typedef struct eth\_hdr //以太网首部

{

u8 d\_mac[6];

u8 s\_mac[6];

u16 type;

}ETH\_HDR;

typedef struct arp\_hdr //ARP首部

{

ETH\_HDR ethhdr; //以太网首部

u16 hwtype;

u16 protocol;

u8 hwlen;

u8 protolen;

u16 opcode;

u8 smac[6];

u8 sipaddr[4];

u8 dmac[6];

u8 dipaddr[4];

}ARP\_HDR;

ARP\_HDR arpbuf;

extern u8 host\_mac\_addr[6];

extern u8 mac\_addr[6];

extern u8 ip\_addr[4];

extern u8 host\_ip\_addr[4];

extern u16 packet\_len;

extern u8 \*buffer;

#include "arp.h"

#define HON(n) ((((u16)((n) & 0xff)) << 8) | (((n) & 0xff00) >> 8))

/\*1.发送arp请求包\*/

void arp\_request()

{

/\*1.构成arp请求包\*/

memcpy(arpbuf.ethhdr.d\_mac,host\_mac\_addr,6);

memcpy(arpbuf.ethhdr.s\_mac,mac\_addr,6);

arpbuf.ethhdr.type = HON(0x0806);

arpbuf.hwtype = HON(1);

arpbuf.protocol = HON(0x0800);

arpbuf.hwlen = 6;

arpbuf.protolen = 4;

arpbuf.opcode = HON(1);

memcpy(arpbuf.smac,mac\_addr,6);

memcpy(arpbuf.sipaddr,ip\_addr,4);

memcpy(arpbuf.dipaddr,host\_ip\_addr,4);

packet\_len = 14+28;

/\*2.调用dm9000发送函数，发送应答包\*/

dm9000\_tx(buffer,packet\_len);

}

/\*2.解析arp应答包，提取mac\*/

u8 arp\_process()

{

u32 i;

if (packet\_len<28)

return 0;

memcpy(host\_ip\_addr,arpbuf.sipaddr,4);

printf("host ip is : ");

for(i=0;i<4;i++)

printf("%03d ",host\_ip\_addr[i]);

printf("\n\r");

memcpy(host\_mac\_addr,arpbuf.smac,6);

printf("host mac is : ");

for(i=0;i<6;i++)

printf("%02x ",host\_mac\_addr[i]);

printf("\n\r");

}

### >>dev/dm9000.c:

文件底部添加测试函数:

void dm9000\_arp()

{

while(1)

arp\_request();

}

### >>dev/interrupt.c:

void handle\_int()中添加:

case 4: //EINT4~K2

int\_issue();

break;

### 测试:main.c

//char \*buf = "hello world";

int gboot\_main()

{

int num;

#ifdef MMU\_ON

mmu\_init();

#endif

led\_init();

button\_init();

init\_irq();

uart\_init();

ts\_init();

//lcd\_init();

//lcd\_test();

dm9000\_init();

dm9000\_arp();

while(1)

{

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*GBOOT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");

printf("1:Download Linux Kernel from TFTP Server!\n\r");

printf("2:Boot Linux from RAM!\n\r");

printf("3:Boor Linux from Nand Flash!\n\r");

printf("\n Plese Select:");

scanf("%d",&num);

switch (num)

{

case 1:

//tftp\_load();

break;

case 2:

//boot\_linux\_ram();

break;

case 3:

//boot\_linux\_nand();

break;

default:

printf("Error: wrong selection!\n\r");

break;

}

}

return 0;

}

# 专题09-移植tftp客户端

[主目录](#_目录)

## 核心理论

[主目录](#_目录)

tftp作用？

tftp是无需OS的支持的一种网络协议。所以它的客户端的使用可以不带操作系统。

客户端向服务器发tftp请求包之前还得向服务器发送ARP请求包以获取服务器的MAC地址。我们把网络协议相关代码放到net目录下。

## Tftp报文格式

[主目录](#_目录)

IP数据报文：IP首部（20B）+UDP首部（8B）+操作码（2B）+文件名+0（1B）+模式（NB）+0（1B）

UDP数据报文：UDP首部（8B）+操作码（2B）+文件名+0（1B）+模式（NB）+0（1B）

TFTP数据报文：操作码（2B）+文件名+0（1B）+模式（NB）+0（1B）

操作码：（1-RRQ，2-WRQ，3-data，4-ACK，5-data（error））

操作码=1时：操作码（2B）+文件名+0（1B）+模式（NB）+0（1B）

操作码=2时：操作码（2B）+文件名+0（1B）+模式（NB）+0（1B）

操作码=3时：操作码（2B）+块编号（2B）+数据（0~512B）

操作码=4时：操作码（2B）+块编号（2B）

操作码=5时：操作码（2B）+差错码（2B）+差错信息（NB）+0（1B）

我们学习一种网络协议，就需要知道它的协议结构以及交互过程。

下面是各种协议包的结构体以及它们之间的关系：

typedef struct eth\_hdr { //sizeof=14

u8 d\_mac[6];

u8 s\_mac[6];

u16 type;

}ETH\_HDR;

typedef struct arp\_hdr { //

ETH\_HDR ethhdr;

u16 hwtype;

u16 protocol;

u8 hwlen;

u8 protolen;

u16 opcode;

u8 smac[6];

u8 sipaddr[4];

u8 dmac[6];

u8 dipaddr[4];

}ARP\_HDR;

typedef struct ip\_hdr {

ETH\_HDR ethhdr;

u8 vhl;

u8 tos;

u16 len;

u16 ipid;

u16 ipoffset;

u8 ttl;

u8 proto;

u16 ipchksum;

u8 srcipaddr[4];

u8 destipaddr[4];

} IP\_HDR;

typedef struct udp\_hdr {

IP\_HDR iphdr;

u16 sport;

u16 dport;

u16 len;

u16 udpchksum;

} UDP\_HDR;

typedef struct tftp\_package {

u16 opcode;

u16 blocknum;

u8 data[0];

} TFTP\_PACK;

## tftp网络协议实现

### dev/dm9000.c:

#include "dm9000.h"

#include "arp.h"

#define DM\_ADD (\*((volatile unsigned short \*)0x20000300))

#define DM\_DAT (\*((volatile unsigned short \*)0x20000304))

#define GPFCON (\*(volatile unsigned \*)0x56000050) //Port F control

#define EXTINT0 (\*(volatile unsigned \*)0x56000088) //External interrupt control register 0

#define EINTMASK (\*(volatile unsigned \*)0x560000a4) //External interrupt mask

#define SRCPND (\*(volatile unsigned \*)0x4a000000) //Interrupt request status

#define INTPND (\*(volatile unsigned \*)0x4a000010) //Interrupt request status

#define INTMSK (\*(volatile unsigned \*)0x4a000008) //Interrupt mask control

#define EINTPEND (\*(volatile unsigned \*)0x560000a8) //External interrupt pending

#define BWSCON (\*(volatile unsigned \*)0x48000000) //Bus width & wait status

#define BANKCON4 (\*(volatile unsigned \*)0x48000014) //BANK4 control

u8 buffer[1500];

u8 host\_mac\_addr[6] = {0xff,0xff,0xff,0xff,0xff,0xff};

u8 mac\_addr[6] = {9,8,7,6,5,4};

u8 ip\_addr[4] = {192,168,1,30};

u8 host\_ip\_addr[4] = {192,168,1,100};

u16 packet\_len;

void cs\_init()

{

BWSCON = BWSCON & (~(0x3<<16));

BWSCON = BWSCON |(0x1<<16);

BANKCON4 = (0x0<<13)|(0x0<<11)|(0x7<<8)|(0x1<<6)|(0x0<<4)|(0x0<<2)|(0x0<<0);

}

void int\_init()

{

GPFCON = GPFCON &(~(0x3<<14));

GPFCON = GPFCON |(0x2<<14);

EXTINT0 = EXTINT0 & (~(0x7<<28));

EXTINT0 = EXTINT0 | (0x1<<28);

INTMSK = INTMSK &(~(1<<4));

EINTMASK = EINTMASK & (~(0x1<<7));

SRCPND = (1<<4);

INTPND = (1<<4);

}

void dm9000\_reg\_write(u16 reg,u16 data)

{

DM\_ADD = reg;

DM\_DAT = data;

}

u8 dm9000\_reg\_read(u16 reg)

{

DM\_ADD = reg;

return DM\_DAT;

}

void dm9000\_reset()

{

dm9000\_reg\_write(DM9000\_GPCR, GPCR\_GPIO0\_OUT);

dm9000\_reg\_write(DM9000\_GPR, 0);

dm9000\_reg\_write(DM9000\_NCR, (NCR\_LBK\_INT\_MAC | NCR\_RST));

dm9000\_reg\_write(DM9000\_NCR, 0);

dm9000\_reg\_write(DM9000\_NCR, (NCR\_LBK\_INT\_MAC | NCR\_RST));

dm9000\_reg\_write(DM9000\_NCR, 0);

}

void dm9000\_probe(void)

{

u32 id\_val;

id\_val = dm9000\_reg\_read(DM9000\_VIDL);

id\_val |= dm9000\_reg\_read(DM9000\_VIDH) << 8;

id\_val |= dm9000\_reg\_read(DM9000\_PIDL) << 16;

id\_val |= dm9000\_reg\_read(DM9000\_PIDH) << 24;

if (id\_val == DM9000\_ID) {

printf("dm9000 is found !\n");

return ;

} else {

printf("dm9000 is not found !\n");

return ;

}

}

void dm9000\_init()

{

u32 i;

/\*设置片选\*/

cs\_init();

/\*中断初始化\*/

int\_init();

/\*复位设备\*/

dm9000\_reset();

/\*捕获dm9000\*/

dm9000\_probe();

/\*MAC初始化\*/

/\* Program operating register, only internal phy supported \*/

dm9000\_reg\_write(DM9000\_NCR, 0x0);

/\* TX Polling clear \*/

dm9000\_reg\_write(DM9000\_TCR, 0);

/\* Less 3Kb, 200us \*/

dm9000\_reg\_write(DM9000\_BPTR, BPTR\_BPHW(3) | BPTR\_JPT\_600US);

/\* Flow Control : High/Low Water \*/

dm9000\_reg\_write(DM9000\_FCTR, FCTR\_HWOT(3) | FCTR\_LWOT(8));

/\* SH FIXME: This looks strange! Flow Control \*/

dm9000\_reg\_write(DM9000\_FCR, 0x0);

/\* Special Mode \*/

dm9000\_reg\_write(DM9000\_SMCR, 0);

/\* clear TX status \*/

dm9000\_reg\_write(DM9000\_NSR, NSR\_WAKEST | NSR\_TX2END | NSR\_TX1END);

/\* Clear interrupt status \*/

dm9000\_reg\_write(DM9000\_ISR, ISR\_ROOS | ISR\_ROS | ISR\_PTS | ISR\_PRS);

/\*填充MAC地址\*/

for (i = 0; i < 6; i++)

dm9000\_reg\_write(DM9000\_PAR+i, mac\_addr[i]);

/\*激活DM9000\*/

dm9000\_reg\_write(DM9000\_RCR, RCR\_DIS\_LONG | RCR\_ALL | RCR\_DIS\_CRC | RCR\_RXEN);

/\* Enable TX/RX interrupt mask \*/

dm9000\_reg\_write(DM9000\_IMR, IMR\_PAR);

}

void dm9000\_tx(u8 \*data,u32 length)

{

u32 i;

/\*禁止中断\*/

dm9000\_reg\_write(DM9000\_IMR,0x80);

/\*写入发送数据的长度\*/

dm9000\_reg\_write(DM9000\_TXPLL, length & 0xff);

dm9000\_reg\_write(DM9000\_TXPLH, (length >> 8) & 0xff);

/\*写入待发送的数据\*/

DM\_ADD = DM9000\_MWCMD;

for(i=0;i<length;i+=2)

{

DM\_DAT = data[i] | (data[i+1]<<8);

}

/\*启动发送\*/

dm9000\_reg\_write(DM9000\_TCR, TCR\_TXREQ);

/\*等待发送结束\*/

while(1)

{

u8 status;

status = dm9000\_reg\_read(DM9000\_TCR);

if((status&0x01)==0x00)

break;

}

/\*清除发送状态\*/

dm9000\_reg\_write(DM9000\_NSR,0x2c);

/\*恢复中断使能\*/

dm9000\_reg\_write(DM9000\_IMR,0x81);

}

#define PTK\_MAX\_LEN 1522

u32 dm9000\_rx(u8 \*data)

{

u16 status,len;

u16 tmp;

u32 i;

/\*判断是否产生中断，且清除\*/

if(dm9000\_reg\_read(DM9000\_ISR) & 0x01)

dm9000\_reg\_write(DM9000\_ISR,0x01);

else

return 0;

/\*空读\*/

dm9000\_reg\_read(DM9000\_MRCMDX);

/\*读取状态\*/

status = dm9000\_reg\_read(DM9000\_MRCMD);

/\*读取包长度\*/

len = DM\_DAT;

/\*读取包数据\*/

if(len<PTK\_MAX\_LEN)

{

for(i=0;i<len;i+=2)

{

tmp = DM\_DAT;

data[i] = tmp & 0x0ff;

data[i+1] = (tmp>>8)&0x0ff;

}

}

return len;

}

**void int\_issue() //添加事件中断入口**

**{**

**packet\_len = dm9000\_rx(&buffer[0]);**

**net\_handle(&buffer[0],packet\_len);**

**SRCPND = (1<<4);**

**INTPND = (1<<4);**

**EINTPEND |= 1<<7;**

**}**

void dm9000\_arp()

{

while(1)

arp\_request();

}

### net/arp.c:

#include "arp.h"

#define HON(n) ((((u16)((n) & 0xff)) << 8) | (((n) & 0xff00) >> 8))

/\*1.发送arp请求包\*/

void arp\_request()

{

/\*1.构成arp请求包\*/

memcpy(arpbuf.ethhdr.d\_mac,host\_mac\_addr,6);

memcpy(arpbuf.ethhdr.s\_mac,mac\_addr,6);

arpbuf.ethhdr.type = HON(0x0806);

arpbuf.hwtype = HON(1);

arpbuf.protocol = HON(0x0800);

arpbuf.hwlen = 6;

arpbuf.protolen = 4;

arpbuf.opcode = HON(1);

memcpy(arpbuf.smac,mac\_addr,6);

memcpy(arpbuf.sipaddr,ip\_addr,4);

memcpy(arpbuf.dipaddr,host\_ip\_addr,4);

packet\_len = 14+28;

/\*2.调用dm9000发送函数，发送应答包\*/

dm9000\_tx(&arpbuf,packet\_len);

}

/\*2.解析arp应答包，提取mac\*/

u8 arp\_process(u8 \*buf, u32 len)

{

u32 i;

ARP\_HDR \*arp\_p = (ARP\_HDR \*)buf;

if (packet\_len<28)

return 0;

switch (HON(arp\_p->opcode))

{

case 2: /\*arp响应包\*/

/\*发送端发送ARP请求包\*/

memcpy(host\_ip\_addr,arp\_p->sipaddr,4);

printf("host ip is : ");

for(i=0;i<4;i++)

printf("%03d ",host\_ip\_addr[i]);

printf("\n\r");

memcpy(host\_mac\_addr,arp\_p->smac,6);

printf("host mac is : ");

for(i=0;i<6;i++)

printf("%02x ",host\_mac\_addr[i]);

printf("\n\r");

break;

case 1: /\*arp请求包\*/

/\*接收端发送arp响应包\*/

memcpy(arpbuf.ethhdr.d\_mac,arp\_p->smac,6);

memcpy(arpbuf.ethhdr.s\_mac,mac\_addr,6);

arpbuf.ethhdr.type = HON(0x0806);

arpbuf.hwtype = HON(1);

arpbuf.protocol = HON(0x0800);

arpbuf.hwlen = 6;

arpbuf.protolen = 4;

arpbuf.opcode = HON(2);

memcpy(arpbuf.smac,mac\_addr,6);

memcpy(arpbuf.sipaddr,ip\_addr,4);

memcpy(arpbuf.dmac,arp\_p->smac,6);

memcpy(arpbuf.dipaddr,arp\_p->sipaddr,4);

packet\_len = 14+28;

/\*2.调用dm9000发送函数，发送应答包\*/

dm9000\_tx(&arpbuf,packet\_len);

break;

}

}

void udp\_process(u8\* buf, u32 len)

{

UDP\_HDR \*udphdr = (UDP\_HDR \*)buf;

tftp\_process(buf,len,HON(udphdr->sport));

}

void ip\_process(u8 \*buf, u32 len)

{

IP\_HDR \*p = (IP\_HDR \*)buf;

switch(p->proto)

{

case PROTO\_UDP:

udp\_process(buf,len);

break;

default:

break;

}

}

void net\_handle(u8 \*buf, u32 len)

{

ETH\_HDR \*p = (ETH\_HDR \*)buf;

switch (HON(p->type))

{

case PROTO\_ARP:

arp\_process(buf,len);

break;

case PROTO\_IP:

ip\_process(buf,len);

break;

default:

break;

}

}

### net/tftp.c:

#include "string.h"

#include "arp.h"

u8 sendbuf[1024];

u8\* tftp\_down\_addr = 0x31000000;

u16 serverport = 0;

u16 curblock = 1;

#define HON(n) ((((u16)((n) & 0xff)) << 8) | (((n) & 0xff00) >> 8))

u16 checksum(u8 \*ptr, int len)

{

u32 sum = 0;

u16 \*p = (u16 \*)ptr;

while (len > 1)

{

sum += \*p++;

len -= 2;

}

if(len == 1)

sum += \*(u8 \*)p;

while(sum>>16)

sum = (sum&0xffff) + (sum>>16);

return (u16)((~sum)&0xffff);

}

void tftp\_send\_request(const char \*filename)

{

u8 \*ptftp = &sendbuf[200];

u32 tftp\_len = 0;

UDP\_HDR \*udphdr;

u8 \*iphdr;

ptftp[0] = 0x00;

ptftp[1] = 0x01;

tftp\_len += 2 ;

sprintf(&ptftp[tftp\_len],"%s",filename);

tftp\_len += strlen(filename);

ptftp[tftp\_len] = '\0';

tftp\_len += 1;

sprintf(&ptftp[tftp\_len],"%s","octet");

tftp\_len += strlen("octect");

ptftp[tftp\_len] = '\0';

tftp\_len += 1;

udphdr = ptftp-sizeof(UDP\_HDR);

iphdr = ptftp-sizeof(UDP\_HDR)+ sizeof(ETH\_HDR);

/\*UDP帧头信息\*/

udphdr->sport = HON(48915);

udphdr->dport = HON(69);

udphdr->len = HON(tftp\_len+sizeof(UDP\_HDR)-sizeof(IP\_HDR));

udphdr->udpchksum = 0x00;

/\*IP帧头信息\*/

udphdr->iphdr.vhl = 0x45;

udphdr->iphdr.tos = 0x00;

udphdr->iphdr.len = HON(tftp\_len+sizeof(UDP\_HDR)-sizeof(ETH\_HDR));

udphdr->iphdr.ipid = HON(0x00);

udphdr->iphdr.ipoffset = HON(0x4000);

udphdr->iphdr.ttl = 0xff;

udphdr->iphdr.proto = 17;

memcpy(udphdr->iphdr.srcipaddr,ip\_addr,4);

memcpy(udphdr->iphdr.destipaddr,host\_ip\_addr,4);

udphdr->iphdr.ipchksum = 0;

udphdr->iphdr.ipchksum = checksum(iphdr,20);

memcpy(udphdr->iphdr.ethhdr.s\_mac,mac\_addr,6);

memcpy(udphdr->iphdr.ethhdr.d\_mac,host\_mac\_addr,6);

udphdr->iphdr.ethhdr.type = HON(PROTO\_IP);

dm9000\_tx((u32 \*)udphdr,sizeof(UDP\_HDR)+tftp\_len);

}

void tftp\_send\_ack(u16 blocknum)

{

u8 \*ptftp = &sendbuf[200];

u32 tftp\_len = 0;

UDP\_HDR \*udphdr;

u8 \*iphdr;

ptftp[0] = 0x00;

ptftp[1] = 0x04;

tftp\_len += 2 ;

ptftp[2] = (blocknum&0xff00)>>8;

ptftp[3] = (blocknum&0xff);

tftp\_len += 2 ;

udphdr = ptftp-sizeof(UDP\_HDR);

iphdr = ptftp-sizeof(UDP\_HDR)+ sizeof(ETH\_HDR);

/\*UDP帧头信息\*/

udphdr->sport = HON(48915);

udphdr->dport = HON(serverport);

udphdr->len = HON(tftp\_len+sizeof(UDP\_HDR)-sizeof(IP\_HDR));

udphdr->udpchksum = 0x00;

/\*IP帧头信息\*/

udphdr->iphdr.vhl = 0x45;

udphdr->iphdr.tos = 0x00;

udphdr->iphdr.len = HON(tftp\_len+sizeof(UDP\_HDR)-sizeof(ETH\_HDR));

udphdr->iphdr.ipid = HON(0x00);

udphdr->iphdr.ipoffset = HON(0x4000);

udphdr->iphdr.ttl = 0xff;

udphdr->iphdr.proto = 17;

memcpy(udphdr->iphdr.srcipaddr,ip\_addr,4);

memcpy(udphdr->iphdr.destipaddr,host\_ip\_addr,4);

udphdr->iphdr.ipchksum = 0;

udphdr->iphdr.ipchksum = checksum(iphdr,20);

memcpy(udphdr->iphdr.ethhdr.s\_mac,mac\_addr,6);

memcpy(udphdr->iphdr.ethhdr.d\_mac,host\_mac\_addr,6);

udphdr->iphdr.ethhdr.type = HON(PROTO\_IP);

dm9000\_tx((u32 \*)udphdr,sizeof(UDP\_HDR)+tftp\_len);

}

void tftp\_process(u8 \*buf, u32 len, u16 port)

{

u32 i;

u32 tftp\_len;

serverport = port;

TFTP\_PAK \*ptftp = buf + sizeof(UDP\_HDR);

tftp\_len = len - sizeof(UDP\_HDR);

if(HON(ptftp->opcode) == 3)

{

if (HON(ptftp->blocknum) == curblock)

{

for (i = 0;i<(tftp\_len-4);i++)

{

\*(tftp\_down\_addr) = \*(ptftp->data+i);

}

tftp\_send\_ack(HON(ptftp->blocknum));

curblock++;

if ((tftp\_len-4)<512)

curblock = 1;

}

}

}

### net/Makefile:

objs := arp.o tftp.o

all : $(objs)

arm-linux-ld -r -o net.o $^

%.o : %.c

arm-linux-gcc ${CFLAGS} -c $^

%.o : %.S

arm-linux-gcc ${CFLAGS} -c $^

clean:

rm -f \*.o \*.bak

### Makefile:

OBJS := start.o main.o dev/dev.o lib/lib.o net/net.o

CFLAGS := -fno-builtin -I$(shell pwd)/include

export CFLAGS

gboot.bin : gboot.elf

arm-linux-objcopy -O binary gboot.elf gboot.bin

gboot.elf : $(OBJS)

arm-linux-ld -Tgboot.lds -o gboot.elf $^

%.o : %.S

arm-linux-gcc -g -c $^

%.o : %.c

arm-linux-gcc $(CFLAGS) -c $^

lib/lib.o :

make -C lib all

dev/dev.o :

make -C dev all

net/net.o :

make -C net all

.PHONY: clean

clean:

make -C lib clean

make -C dev clean

make -C net clean

rm -f \*.o \*.elf \*.bin \*.bak

### main.c:

//char \*buf = "hello world";

int gboot\_main()

{

int num;

#ifdef MMU\_ON

mmu\_init();

#endif

led\_init();

button\_init();

init\_irq();

uart\_init();

ts\_init();

//lcd\_init();

//lcd\_test();

dm9000\_init();

while(1)

{

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*GBOOT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");

printf("1:Download Linux Kernel from TFTP Server!\n\r");

printf("2:Boot Linux from RAM!\n\r");

printf("3:Boor Linux from Nand Flash!\n\r");

printf("\n Plese Select:");

scanf("%d",&num);

switch (num)

{

case 1:

//tftp\_load();

tftp\_send\_request("start.o");

break;

case 2:

//boot\_linux\_ram();

arp\_request();

break;

case 3:

//boot\_linux\_nand();

break;

default:

printf("Error: wrong selection!\n\r");

break;

}

}

return 0;

}

# 专题10-移植bootm命令启动内核

[主目录](#_目录)

## 核心理论

[主目录](#_目录)

Bootm职能

检查信息头：压缩、OS、幻数

根据OS类型，调用相应处理函数（如Linux系统）

获取Linux内核地址

要知道CPU型号

设置启动参数

uImage = 信息头 + zImage

我们的bootm命令只能使用uImage，因为我们的bootm在启动时会检查这个头，如果没有这个头就会出错退出。

Linux内核中执行bootm命令的函数名:do\_bootm()

分析do\_bootm()函数得到流程信息：

\*\*检查信息头:压缩信息、OS信息、幻数

\*\*根据OS类型，调用相应处理函数（如Linux系统）

\*\*获取Linux内核位置:内核入口

\*\*arch:处理器信号参数

\*\*param:启动参数

## bootm命令移植启动内核

### atag.h:

#define ATAG\_CORE 0x54410001

#define ATAG\_MEM 0x54410002

#define ATAG\_CMDLINE 0x54410009

#define ATAG\_NONE 0x00000000

struct tag\_header {

unsigned int size;

unsigned int tag;

};

struct tag\_core {

unsigned int flags;

unsigned int pagesize;

unsigned int rootdev;

};

struct tag\_mem32 {

unsigned int size;

unsigned int start;

};

struct tag\_cmdline {

char cmdline[1];

};

struct tag {

struct tag\_header hdr;

union {

struct tag\_core core;

struct tag\_mem32 mem;

struct tag\_cmdline cmdline;

} u;

};

#define tag\_size(type) ((sizeof(struct tag\_header) + sizeof(struct type)) >> 2)

#define tag\_next(t) ((struct tag \*)((unsigned int \*)(t) + (t)->hdr.size))

### boot.c:

#include "atag.h"

#include "string.h"

#define SDRAM\_KERNEL\_START 0x31000000 //内核起始地址

#define SDRAM\_TAGS\_START 0x30000100 //参数起始地址

#define SDRAM\_ADDR\_START 0x30000000 //内存起始地址

#define SDRAM\_TOTAL\_SIZE 0x04000000 //内存总大小64MB

void (\*theKernel)(int , int , unsigned int );

struct tag \*pCurTag;

const char \*cmdline = "console=ttySAC0,115200 init=/init";

void setup\_core\_tag()

{

pCurTag = (struct tag \*)SDRAM\_TAGS\_START;

pCurTag->hdr.tag = ATAG\_CORE;

pCurTag->hdr.size = tag\_size(tag\_core);

pCurTag->u.core.flags = 0;

pCurTag->u.core.pagesize = 4096;

pCurTag->u.core.rootdev = 0;

pCurTag = tag\_next(pCurTag);

}

void setup\_mem\_tag()

{

pCurTag->hdr.tag = ATAG\_MEM;

pCurTag->hdr.size = tag\_size(tag\_mem32);

pCurTag->u.mem.start = SDRAM\_ADDR\_START;

pCurTag->u.mem.size = SDRAM\_TOTAL\_SIZE;

pCurTag = tag\_next(pCurTag);

}

void setup\_cmdline\_tag()

{

int linelen = strlen(cmdline);

pCurTag->hdr.tag = ATAG\_CMDLINE;

pCurTag->hdr.size = (sizeof(struct tag\_header)+linelen+1+4)>>2;

strcpy(pCurTag->u.cmdline.cmdline,cmdline);

pCurTag = tag\_next(pCurTag);

}

void setup\_end\_tag()

{

pCurTag->hdr.tag = ATAG\_NONE;

pCurTag->hdr.size = 0;

}

void boot\_linux()

{

//1. 获取Linux启动地址

theKernel = (void (\*)(int , int , unsigned int ))SDRAM\_KERNEL\_START;

//2. 设置启动参数

//2.1 设置核心启动参数

setup\_core\_tag();

//2.2 设置内存参数

setup\_mem\_tag();

//2.3 设置命令行参数

setup\_cmdline\_tag();

//2.4 设置参数结束标志

setup\_end\_tag();

//3. 启动Linux系统

theKernel(0,168,SDRAM\_TAGS\_START);

// theKernel(0,处理器编号,SDRAM\_TAGS\_START);

//此处处理器编号必须和Linux内核里的处理器编号相同内核才能正常启动

//Linux内核处理器编号索引：arch/arm/tools/mach-types

}

### Makefile:

OBJS := start.o main.o boot.o dev/dev.o lib/lib.o net/net.o

CFLAGS := -fno-builtin -I$(shell pwd)/include

export CFLAGS

**gboot.bin : gboot.elf**

**arm-linux-objcopy -O binary gboot.elf gboot.bin**

**gboot.elf : $(OBJS)**

**arm-linux-ld -Tgboot.lds -o gboot.elf $^**

%.o : %.S

arm-linux-gcc -g -c $^

%.o : %.c

arm-linux-gcc $(CFLAGS) -c $^

**lib/lib.o :**

**make -C lib all**

**dev/dev.o :**

**make -C dev all**

**net/net.o :**

**make -C net all**

.PHONY: clean

clean:

make -C lib clean

make -C dev clean

make -C net clean

rm -f \*.o \*.elf \*.bin \*.bak

### main.c:

//char \*buf = "hello world";

int gboot\_main()

{

int num;

#ifdef MMU\_ON

mmu\_init();

#endif

led\_init();

button\_init();

init\_irq();

uart\_init();

ts\_init();

//lcd\_init();

//lcd\_test();

dm9000\_init();

while(1)

{

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*GBOOT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");

printf("1:Download Linux Kernel from TFTP Server!\n\r");

printf("2:Boot Linux from RAM!\n\r");

printf("3:Send ARP to PC!\n\r");

printf("\n Plese Select:");

scanf("%d",&num);

switch (num)

{

case 1:

tftp\_send\_request("zImage");

break;

case 2:

boot\_linux();

break;

case 3:

arp\_request();

break;

default:

printf("Error: wrong selection!\n\r");

break;

}

}

return 0;

}

## boot.c

[主目录](#_目录)

#include "atag.h"

#include "string.h"

#define SDRAM\_KERNEL\_START 0x31000000 //内核起始地址

#define SDRAM\_TAGS\_START 0x30000100 //参数起始地址

#define SDRAM\_ADDR\_START 0x30000000 //内存起始地址

#define SDRAM\_TOTAL\_SIZE 0x04000000 //内存总大小64MB

void (\*theKernel)(int , int , unsigned int );

struct tag \*pCurTag;

const char \*cmdline = "console=ttySAC0,115200 init=/init";

void setup\_core\_tag()

{

//第一个参数起始地址

pCurTag = (struct tag \*)SDRAM\_TAGS\_START;

pCurTag->hdr.tag = ATAG\_CORE;

pCurTag->hdr.size = tag\_size(tag\_core); //参考于uboot

//参考于uboot

pCurTag->u.core.flags = 0;

pCurTag->u.core.pagesize = 4096;

pCurTag->u.core.rootdev = 0;

//参考于uboot：指向下一个参数区

pCurTag = tag\_next(pCurTag);

}

void setup\_mem\_tag()

{

pCurTag->hdr.tag = ATAG\_MEM;

pCurTag->hdr.size = tag\_size(tag\_mem32);

pCurTag->u.mem.start = SDRAM\_ADDR\_START;

pCurTag->u.mem.size = SDRAM\_TOTAL\_SIZE;

pCurTag = tag\_next(pCurTag);

}

void setup\_cmdline\_tag()

{

int linelen = strlen(cmdline);

pCurTag->hdr.tag = ATAG\_CMDLINE;

pCurTag->hdr.size = (sizeof(struct tag\_header)+linelen+1+4)>>2;

strcpy(pCurTag->u.cmdline.cmdline,cmdline);

pCurTag = tag\_next(pCurTag);

}

void setup\_end\_tag()

{

pCurTag->hdr.tag = ATAG\_NONE;

pCurTag->hdr.size = 0;

}

void boot\_linux()

{

//1. 获取Linux启动地址

theKernel = (void (\*)(int , int , unsigned int ))SDRAM\_KERNEL\_START;

//2. 设置启动参数

//2.1 设置核心启动参数

setup\_core\_tag();

//2.2 设置内存参数

setup\_mem\_tag();

//2.3 设置命令行参数

setup\_cmdline\_tag();

//2.4 设置参数结束标志

setup\_end\_tag();

//3. 启动Linux系统

// theKernel(0,处理器编号,SDRAM\_TAGS\_START);

theKernel(0,168,SDRAM\_TAGS\_START);

//此处处理器编号必须和Linux内核里的处理器编号相同内核才能正常启动

//Linux内核处理器编号索引：arch/arm/tools/mach-types

}

[主目录](#_目录)

## 模块分离测试办法

[主目录](#_目录)

由于前面的TFTP客户端移植无法保证其正确性，而今天的bootm移植，从而加大了测试难度；所以必须采用模块分离法来测试，

测试方法如下：

先把测试的gboot.bin下载到开发板中，

下一步通过NOR/SD卡方式启动，

下一步在NOR/SD卡的uboot中通过TFTP下载内核文件zImage到0x31000000内核地址中，

下一步将启动方式改为nandflash方式启动，

下一步迅速重启开发板（此时因为还没掉电，所以内存0x31000000开始处的数据还是存在的），然后来测试移植的bootm命令的正确性。

[主目录](#_目录)

## 内核启动后无法继续向下启动

[主目录](#_目录)

谢老师说：这种情况以后会经常遇到，特别是uboot移植和Linux内核移植的时候，这个时候就要特别注意到处理器编号（处理器参数）；由于bootloader在启动时会把处理器编号通过参数方式传给Linux内核，而Linux内核本身也会有处理器参数，Linux内核会把传过来的参数和Linux内核里的处理器参数作对比，如果相同则继续向下执行，如果不同则停止启动。（Linux内核处理器参数：arch/arm/tools/mach-types）

(arch/<cpu>/tools/mach-types)

[主目录](#_目录)

# 专题11-自我检测:移植USB驱动

# 第四季-内核驱动奥秘多(内核驱动)-上学期

# 专题01-课程规划与学习方法

[主目录](#_目录)

## 关键字索引

# 测试失败TEST\_ERROR！！！暂时搁置...

# 专题02-U-Boot新手入门

[主目录](#_目录)

## 核心理论

[主目录](#_目录)

如何使用uboot

如何开发uboot

BIOS(bootloader)的主要任务就是**将内核从硬盘上读到内存中，然后跳转到内核的入口点去运行，即启动操作系统**。在嵌入式系统中，通常没有像BIOS那样的固件程序，因此整个系统的加载启动任务就完全由bootloader来完成。比如在基于ARM7核的嵌入式系统中，系统在上电或复位时都是从地址0x00000000开始执行。而在这个地址处安排的通常就是系统的bootloader程序。

简单的来说，bootloader就是在OS运行之前运行的一段小程序。通过这段小程序，可以**初始化硬件设备**，从而将OS的软硬件环境带到一个合适的状态，以便为最终调用OS做好准备。

系统上电复位后，所有**CPU通常都从CPU制造商预先安排的地址处开始执行**。比如S3C2410在复位后从0x00000000起开始执行。**而嵌入式硬件系统则将固态存储设备安排在这个地址上(比如:FLASH)**,而**bootloader程序又安排在固态存储器的最前端**，这样就能保证在系统上电后，CPU首先执行bootloader程序。

bootloader启动流程

bootloader启动流程可分为单阶段和多阶段，通常多阶段的bootloader具有更复杂的功能，更好的可移植性。从固态存储设备上启动的bootloader大多采用两阶段，即启动过程可以分为stage1和stage2:stage1完成初始化硬件，为stage2准备内存空间，并将stage2复制到内存中，设置堆栈，然后跳转到stage2(跳转到stage2的C入口点)。stage2通常一下步骤:

\*\*初始化本阶段要使用到的硬件设备

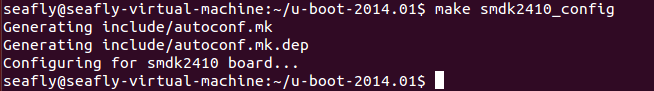
\*\*将内核映像和根文件系统映像从flash上读到RAM中

\*\*调用内核

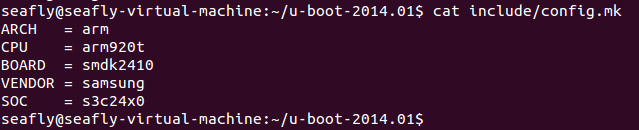
## uboot源码顶层目录说明

UBOOT源码包下载官网：http://ftp.denx.de/pub/u-boot/

|  |  |  |
| --- | --- | --- |
| **目录** | **特性** | **解释说明** |
| board | 平台依赖 | 存放电路板相关的目录文件 |
| cpu | 平台依赖 | 存放CPU相关的目录文件 |
| lib\_xxx | 平台依赖 | 存放与体系结构相关的库文件，如lib\_arm |
| include | 通用 | uboot使用的头文件，还有支持各种硬件平台的汇编文件、系统的配置文件和支持文件系统的文件。该目录下的configs目录中有与开发板配置相关的配置头文件 |
| common | 通用 | uboot支持的命令，每一条命令对应一个文件，如bootm命令对应cmd\_bootm.c文件。 |
| lib\_generic | 通用 | 通用库函数的实现 |
| net | 通用 | 与网络协议栈相关的代码，bootp、tftp、rarp和NFS文件系统的实现。 |
| fs | 通用 | 支持的文件系统，如cramfs,fat,fdos,jffs2和registerfs。 |
| drivers | 通用 | uboot支持的设备驱动都放在该目录中，如各种网卡、串口、USB等。 |
| disk | 通用 | 对磁盘的支持。 |
| doc | 文档 | 文档目录，uboot有非常完善的文档，推荐大家参考阅读 |
| tools | 通用 | 生成uboot工具，如mkimage、CRC等。 |
| examples | 应用例程 | 一些独立运行的应用程序的例子。 |



执行上述命令后，uboot通过uboot顶层目录下的mkconfig脚本生成配置文件include/config.mk，内容如下：



根据上面得知根据硬件平台依赖的目录文件可以根据这些定义来确定。SMDK2410平台相关目录如下：

board/samsung/smdk2410/ (board/$VENDOR/$BOARD)

cpu/arm920t/

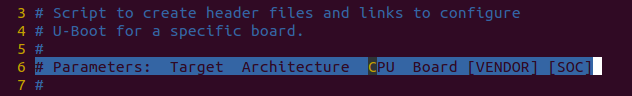
cpu/arm920t/s3c24x0/ (cpu/$CPU/$SOC)

lib\_arm/

include/asm-arm/

include/configs/smdk2410.h

mkconfig用法



## uboot环境变量相关命令

uboot默认的环境变量

|  |  |
| --- | --- |
| 环境变量 | 解释说明 |
| bootdelay | 执行bootcmd中的命令之前的等候秒数 |
| baudrate | 串口控制台的波特率 |
| netmask | 以太网的网络掩码 |
| ethaddr | 以太网的MAC地址 |
| bootfile | 默认的下载文件名 |
| bootargs | 传递给Linux内核的启动参数 |
| bootcmd | 自动启动时执行的命令 |
| serverip | TFTP服务端的IP地址 |
| ipaddr | 开发板本地的IP地址 |
| stdin | 标准输入设备，一般是串口 |
| stdout | 标准输出设备，一般是串口，也可以是LCD（VGA） |
| stderr | 标准出错设备，一般是串口，也可以是LCD（VGA） |

## 开发板无法ping通计算机的原因

有以下几种：

1. uboot网卡驱动有问题
2. uboot网络协议延时配置有问题
3. 网络参数配置有问题，如uboot几个网络相关的环境变量。

## 开发板下载文件到主存

方法一、通过NFS命令下载：

# nfs 0x30008000 192.168.0.6:/home/ubuntu/u-boot.bin

方法二、通过TFTP命令下载：

# tftp 0x30008000 u-boot.bin

## 编译和烧写uboot

[主目录](#_目录)

配置uboot：make mini2440\_config或者make smdk2440\_config,其中的关键字由顶层Makefile中找到。

TQ210： make TQ210\_config

Smart210: make smart210\_config

OK210: make forlinux\_linux\_config

OK6410: make forlinux\_nand\_ram256\_config

Tiny6410: make tiny6410\_config

TQ2440: make TQ2440\_config

Mini2440: make mini2440\_config

uboot下载与运行：

TQ210: tftp 0xc0008000 uImage

Smart210: tftp 0x20000000 uImage

OK210: tftp 0xc0008000 uImage

OK6410: tftp 0xc0008000 uImage

Tiny6410: tftp 0xc0008000 uImage

TQ2440: tftp 0x31000000 uImage

Mini2440: tftp 0x31000000 uImage

编译uboot：make ARCH=arm CROSS\_COMPILE=arm-linux-来编译uboot，编译后产生的u-boot.bin文件则存在于顶层目录中。

NOR/SD卡方式下载uboot.bin：通过NOR/SD方式启动开发板，然后dnw来将uboot.bin下载到nandflash中。

根据不同板子：dnw u-boot.bin 30000000, 最后通过NAND方式启动开发板进入烧写好的uboot中。

Uboot源代码下载地址：<ftp://ftp.denx.de/pub/u-boot/>

## Uboot命令详解

## 环境变量相关命令

[主目录](#_目录)

帮助命令：help

查看环境变量：printenv, printenv var1

新增/修改/删除环境变量：setenv var1 value1; setenv var1 value2; setenv var1

保存环境变量到FLASH中：saveenv

## 程序下载命令

[主目录](#_目录)

程序下载命令：tftp 0x31000000 uImage, 由于串口方式下载内核这类大点的文件时会很慢，所以使用tftp。

但使用tftp下载之前，需要先配置好网络：

## **Linux的tftp服务器的配置**

编辑Linux下的/etc/xinet.d/tftp配置文件

[主目录](#_目录)

# default: off

# description: The tftp server serves files using the trivial file transfer \

# protocol. The tftp protocol is often used to boot diskless \

# workstations, download configuration files to network-aware printers, \

# and to start the installation process for some operating systems.

**service** tftp

**{**

socket\_type **=** dgram

protocol **=** udp

**wait** **=** yes

user **=** root

server **=** **/**usr**/**sbin**/**in.tftpd

server\_args **=** -s **/**tftpboot

disable **=** no

per\_source **=** 11

cps **=** 100 2

flags **=** IPv4

**}**

启动Linux下的tftp服务程序

[主目录](#_目录)

**[**root**@**redhat6 redhat6**]**# vim **/**etc**/**xinetd.d**/**tftp

**[**root**@**redhat6 redhat6**]**# **/**etc**/**init.d**/**xinetd start

Starting xinetd**:**

**[**root**@**redhat6 redhat6**]**# **/**etc**/**init.d**/**xinetd restart

Stopping xinetd**:** **[** OK **]**

Starting xinetd**:** **[** OK **]**

**[**root**@**redhat6 redhat6**]**# netstat -a **|** **grep** "tftp"

udp 0 0 **\*:**tftp **\*:\***

**[**root**@**redhat6 redhat6**]**#

配置好开发板上的网络参数

[主目录](#_目录)

# setenv ipaddr 192.168.1.15 //设置开发板IP地址

# setenv serverip 192.168.1.10 //设置服务端tftp的IP地址

# saveev 之后重启开发板

# tftp 0x31000000 uImage

# bootm 0x31000000 启动指定位置内核文件

## 内存操作命令

[主目录](#_目录)

内存查看命令：md [.b, .w, .l] address

内存修改命令：mm address

## FLASH操作命令

[主目录](#_目录)

擦除nandflash命令：nand erase 起始地址长度

范例：nand erase 0x400000 0x500000

读nandflash命令：nand read 内存起始 FLASH起始地址长度

范例：nand read c0008000 400000 500000

写nandflash命令：nand write 内存起始 FLASH起始地址长度

范例：nand write c0008000 400000 500000

## 设置自动启动参数

[主目录](#_目录)

设置从nandflash自动启动

# setenv bootcmd nand read c0008000 400000 500000 \; bootm c0008000

设置自动下载内核文件到内存后启动

# setenv bootcmd tftp c0008000 uImage.bin \; bootm c0008000

开发板设置NFS启动参数

# setenv bootargs noinitrd console=ttySAC0,115200 init=/init root=/dev/nfs \

nfsroot=192.168.1.10:/home/redhat6/rootfs \

ip=192.168.1.15:192.168.1.10:192.168.1.1:255.255.255.0::eth0:off

uboot工作模式

大多数bootloader都包含两种不同的操作模式：启动模式和下载模式，这种区别仅对于开发人员才有意义。启动模式也成为自主模式，是指bootloader从目标机上的某个固态存储设备上将OS自动加载到RAM中运行，并且整个过程没有用户的介入。这种模式也是产品最终开发出来的模式，也就是用户使用的模式。下载模式就是bootloader将通过串口或网络等通信手段从宿主机下载文件，然后控制启动流程。

为什么要对uboot移植？

因为它依赖于具体的CPU体系、具体的板级设备配置(芯片级移植、板级移植)

Uboot移植:移植方法

开始移植之前，首先要**分析uboot已经支持的开发板**，选择出**硬件配置最接近的开发板**。选择的原则是，**首先选择MCU相同的开发板**，如果没有，则选择**MPU相同的开发板**。

移植范例:mini2440为例

以mini2440开发板为例，该板子采用s3c2440芯片，根据选择原则，**首先选择****MCU为s3c2440的开发板**，但uboot各版本均不支持。因此**根据第二原则，选择MPU相同，即ARM核为ARM920T的开发板**，uboot支持SMDK2410开发板，并且SMDK2440采用s3c2410芯片，s3c2410采用的MPU正好是ARM920T，因此选取SMDK2410开发板作为移植参考板。

MCU和MPU有什么区别？

MCU（micro controller unit）微控制器单元，

MPU （micro processor uint）微处理器单元，

其中，MCU集成了片上外围器件，而MPU不带外围器件（例如存储器阵列），是高度集成的通用结构的处理器，是去除了集成外设的MCU。

**下面的移植工作都是针对mini2440-Norflash来移植的**

## 移植步骤1:修改顶层Makefile

在顶层Makefile中为开发板添加新的配置选项，使用已有的配置项目为例:

smdk2410\_config : unconfig

@./mkconfig $(@:\_config=) arm arm920t smdk2410 NULL s3c24x0

参考上面2行，添加2行:

**mini2440\_config : unconfig**

**@./mkconfig $(@:\_config=) arm arm920t mini2440 NULL s3c24x0**

上面的配置参数说明:

arm:CPU架构

arm920t:CPU类型，对应arch/arm/cpu/arm920t目录

mini2440:开发板型号，对应board/samsung/mini2440目录

NULL:开发者

s3c24x0:片上系统(SOC)

实验1:移植uboot:移植uboot到mini2440开发板:具备ping命令:Norflash写入支持,移植可在mini2440 Norflash中运行的uboot。

以mini2440开发板为例，该板子采用s3c2440芯片，根据选择原则，首先选择MCU为s3c2440的开发板，但uboot各版本均不支持。因此根据第二原则，选择MPU相同，即ARM核为ARM920T的开发板，uboot支持SMDK2410开发板，并且SMDK2440采用s3c2410芯片，s3c2410采用的正好是ARM920T，因此选取SMDK2410开发板作为移植参考板。

## 移植步骤2:创建新开发板目录并添加文件

移植步骤2:创建新开发板目录并添加文件

**mkdir -p board/samsung/mini2440**

**cp -rf board/samsung/smdk2410/\* board/samsung/mini2440**

对smdk2410.c和Makefile做修改

# cd board/samsung/mini2440

# mv smdk2410.c mini2440.c

修改board/mini2440/Makefile相关内容

将代码块:

COBS:=smdk2410.o flash.o

改为:

COBS:=**mini2440.o** flash.o

## 移植步骤3:为开发板添加新配置文件

移植步骤3:为开发板添加新配置文件

先复制参考板的配置文件，再修改:

**cp include/configs/smdk2410.h include/configs/mini2440.h**

## 移植步骤4:体系结构相关文件修改

### 修改include/configs/<board>.h

板级设备的配置文件位于**include/configs/<board>.h**中。

**#define CONFIG\_ARM920T 1 //CPU类型**

**#define CONFIG\_S3C24X0 /\* in a SAMSUNG S3C24x0-type SoC \*/**

**#define CONFIG\_S3C2410 1 //MCU类型**

**#define CONFIG\_SMDK2410 1 //开发板型号**

**#define USE\_920T\_MMU 1 //使用MMU**

#define CONFIG\_SYS\_TEXT\_BASE 0x0

**//不使用IRQ/FIQ**

**#undef CONFIG\_USE\_IRQ /\* we don't need IRQ/FIQ stuff \*/**

#define CONFIG\_CMDLINE\_TAG /\* enable passing of ATAGs \*/

#define CONFIG\_SETUP\_MEMORY\_TAGS

#define CONFIG\_INITRD\_TAG

**//数据段大小:128字节**

**#define CFG\_GBL\_DATA\_SIZE 128**

/\*

\* Hardware drivers

\*/

**//使用CS8900网卡**

**#define CONFIG\_CS8900 /\* we have a CS8900 on-board \*/**

**//CS8900基地址**

**#define CONFIG\_CS8900\_BASE 0x19000300**

#define CONFIG\_CS8900\_BUS16

/\* the Linux driver does accesses as shorts \*/

**//使用串口1**

**#define CONFIG\_SERIAL1 1 /\* we use SERIAL 1 on SMDK2410 \*/**

#define CONFIG\_RTC\_S3C24X0

**//波特率设置:115200**

**#define CONFIG\_BAUDRATE 115200**

**//定义使用的命令，可添加额外命令，如PING**

**#define CONFIG\_CMD\_BSP**

**#define CONFIG\_CMD\_CACHE**

**#define CONFIG\_CMD\_DATE**

**#define CONFIG\_CMD\_DHCP**

**#define CONFIG\_CMD\_ELF**

**#define CONFIG\_CMD\_NAND**

**#define CONFIG\_CMD\_PING**

**#define CONFIG\_CMD\_REGINFO**

**#define CONFIG\_CMD\_USB**

/\* autoboot \*/

**//自动启动时间:5秒**

**#define CONFIG\_BOOTDELAY 5**

**#define CONFIG\_BOOT\_RETRY\_TIME -1**

**#define CONFIG\_RESET\_TO\_RETRY**

**#define CONFIG\_ZERO\_BOOTDELAY\_CHECK**

**#define CONFIG\_NETMASK 255.255.255.0**

**#define CONFIG\_IPADDR 10.0.0.110**

**#define CONFIG\_SERVERIP 10.0.0.1**

**//可用的波特率**

**#define CONFIG\_SYS\_BAUDRATE\_TABLE { 9600, 19200, 38400, 57600, 115200 }**

/\*-----------------------------------------------------------------------

\* Physical Memory Map

\*/

**//有一片SDRAM**

**#define CONFIG\_NR\_DRAM\_BANKS 1 /\* we have 1 bank of DRAM \*/**

#define PHYS\_SDRAM\_1 0x30000000 /\* SDRAM Bank #1 \*/

#define PHYS\_SDRAM\_1\_SIZE 0x04000000 /\* 64 MB \*/

**//flash No1的基地址**

**#define PHYS\_FLASH\_1 0x00000000 /\* Flash Bank #0 \*/**

**//flash的基地址**

**#define CONFIG\_SYS\_FLASH\_BASE PHYS\_FLASH\_1**

**//malloc池大小**

**#define CONFIG\_SYS\_MALLOC\_LEN (4 \* 1024 \* 1024)**

### 修改include/configs/s3c24x0.h:

***片上系统的配置文件修改***

在有s3c2410宏定义开关里加入对s3c2440的支持,

将代码块:

S3C24X0\_REG32 INTPND;

S3C24X0\_REG32 INTOFFSET;

#if defined(CONFIG\_S3C2410)

改为:

S3C24X0\_REG32 INTPND;

S3C24X0\_REG32 INTOFFSET;

/\*modified by seafly, for 2440 \*/

#elif defined(CONFIG\_S3C2410)||**defined(CONFIG\_S3C2440)**

将下面99行左右代码块:

#if defined(CONFIG\_S3C2410)

改为:

#elif defined(CONFIG\_S3C2410)||**defined(CONFIG\_S3C2440)**

将下面110行左右代码块:

#if defined(CONFIG\_S3C2410)

改为:

#elif defined(CONFIG\_S3C2410)||**defined(CONFIG\_S3C2440)**

将下面128行左右代码块:

S3C24X0\_REG32 CLKSLOW

S3C24X0\_REG32 CLKDIVN

#endif

改为:

S3C24X0\_REG32 CLKSLOW **//CLKSLOW:Slow clock control register**

S3C24X0\_REG32 CLKDIVN **//CLKDIVN:Clock divider control register**

**#if defined(CONFIG\_S3C2440)**

**S3C24X0\_REG32 CAMDIVN; //CAMDIVN:Camera clock divider control**

**//添加CAMDIVN寄存器定义**

**#endif**

#endif

将下面148行左右代码块:

#if defined(CONFIG\_S3C2410)

改为:

#if defined(CONFIG\_S3C2410)||**defined(CONFIG\_S3C2440)**

将下面404行左右代码块:

#if defined(CONFIG\_S3C2410)

改为:

#if defined(CONFIG\_S3C2410)||**defined(CONFIG\_S3C2440)**

### 修改include/configs/mini2440.h

***开发板的配置文件修改***

为开发板添加新的软硬件配置文件，先复制参考的smdk2410.h，然后修改:

# cp include/configs/smdk2410.h include/configs/mini2440.h

添加对s3c2440宏定义(37,38行左右):

#define CONFIG\_ARM920T 1

#define CONFIG\_S3C2410 1

#define CONFIG\_SMDK2410 1

改为:

#define CONFIG\_ARM920T 1

**#define CONFIG\_S3C2440 1**

**#define CONFIG\_MINI2410 1**

修改命令提示符(114行左右):

#define CFG\_PROMPT “SMDK2410#”

改为:

**#define CFG\_PROMPT “MINI2440#”**

### 修改arch/arm/cpu/arm920t/s3c24x0/interrupts.c

***片上系统中断相关修改***

**在有s3c2410宏定义开关里加入对s3c2440的支持**

//将下面代码块:

#if defined(CONFIG\_S3C2400)||\

defined(CONFIG\_S3C2410)||\

defined(CONFIG\_TRAB)

//改为:

#if defined(CONFIG\_S3C2400)||\

defined(CONFIG\_S3C2410)||\

defined(CONFIG\_S3C2440)||\

defined(CONFIG\_TRAB)

//将下面代码:

#elif defined(CONFIG\_S3C2410) //改为:

#elif defined(CONFIG\_S3C2410)||defined(CONFIG\_S3C2440)

在函数get\_tbclk中，添加对mini2440支持

//函数中的代码块:

#if defined(CONFIG\_SMDK)||defined(CONFIG\_TRAB)

tbclk **=** timer\_load\_var **\*** 100**;**

#elif defined(CONFIG\_SBC2410X)||\

defined(CONFIG\_SMDK2410)||\

defined(CONFIG\_VCMA9)

tbclk **=** CFG\_HZ**;**

//改为:

#if defined(CONFIG\_SMDK)||defined(CONFIG\_TRAB)

tbclk **=** timer\_load\_var **\*** 100**;**

#elif defined(CONFIG\_SBC2410X)||\

defined(CONFIG\_SMDK2410)||\

defined(CONFIG\_MINI2440)||\

defined(CONFIG\_VCMA9)

tbclk **=** CFG\_HZ**;**

### 修改arch/arm/cpu/arm920t/s3c24x0/serial.c:

***片上系统的串口相关修改***

//在有s3c2410宏定义开关里加入对s3c2440的支持,

//把代码块:

#if defined(CONFIG\_S3C2400)||\

defined(CONFIG\_S3C2410)||\

defined(CONFIG\_TRAB)

//改为:

#if defined(CONFIG\_S3C2400)||\

defined(CONFIG\_S3C2410)||\

defined(CONFIG\_S3C2440)||\

defined(CONFIG\_TRAB)

//将代码块:

#elif defined(CONFIG\_S3C2410)

//改为:

#elif defined(CONFIG\_S3C2410)||defined(CONFIG\_S3C2440)

### 修改arch/arm/include/asm /mach-types.h

***添加新板子的机器ID***

339/660

添加mini2440机器ID，要求添加的ID与内核提供的ID保持一致。在**内核源码文件include/asm-arm/match-types.h**的1859行左右提供有关mini2440的ID

#define MACH\_TYPE\_MINI2440 1999

因此在代码1859行左右:

#define MACH\_TYPE\_GENERA 1873

之后添加:

**#define MACH\_TYPE\_MINI2440 1999**

### 修改drivers/rtc/s3c24x0\_rtc.c:

在有s3c2410宏定义开关里加入对s3c2440的支持,

将代码块:

#elif defined(CONFIG\_S3C2410)

改为:

#elif defined(CONFIG\_S3C2410)||**defined(CONFIG\_S3C2440)**

### 体系相关arch/arm/cpu/arm920t/start.S

***处理器核心文件的修改***

**//该文件用来区分板级配置:同一款处理器(ARM920T),不同板级(S3C2440)**

**//所以需要修改板级相关的硬件相关代码**

**关键字:CONFIG\_S3C2410:添加CONFIG\_S3C2410宏定义:**

**关键字:添加S3C2440片上的时钟初始化**

#if defined(CONFIG\_S3C2400)||defined(CONFIG\_S3C2410)

修改为:

#if defined(CONFIG\_S3C2400)||\

defined(CONFIG\_S3C2410)||\

**defined(CONFIG\_S3C2440)**

**添加寄存器定义**

//在149行左右:下去几行#endif:在#endif之后新增如下这几行(红色):

#ifdef CONFIG\_S3C24X0

/\* turn off the watchdog \*/

//因为看门狗属于核心设置:2410和2440核心都是ARM920T

//同一款CPU,那么他们的地址总线是相同的,所以定义的寄存器地址也基本相同。

# if defined(CONFIG\_S3C2400)

# define pWTCON 0x15300000

# define INTMSK 0x14400008 /\* Interrupt-Controller base addresses \*/

# define CLKDIVN 0x14800014 /\* clock divisor register \*/

#else

# define pWTCON 0x53000000

# define INTMSK 0x4A000008 /\* Interrupt-Controller base addresses \*/

# define INTSUBMSK 0x4A00001C

# define CLKDIVN 0x4C000014 /\* clock divisor register \*/

# endif

/\*\*\*\*\*\*\*\*\*\*\*\*\*紧接着添加寄存器的定义\*\*\*\*\*\*\*\*\*\*\*\*\*/

**//裸机部分:定时器理论知识的巩固:管理时钟体系的这个章节**

**//S3C2440片上时钟体系:**

**// LOCKTIME:Address**

**// MDIV\_405 //MPLLCON**

**// PSDIV\_405**

**// MDIV\_200**

**// PSDIV\_200**

**// UPLL\_MDIV\_48 //UPLLCON**

**// UPLL\_PSDIV\_48**

**//LOCKTIME:0x4c000000:PLL lock time counter**

**#define CLK\_CTL\_BASE 0x4c000000**

**//PLL:MDIV:405MHz:0x7f:CLKDIVN:MPLLCON:**

**#define MDIV\_405 (0x7f<<12)**

**//PDIV|SDIV == 2<<4|1<<0 == 0b0010 0001 = 0x21:MPLLCON:PLLCON**

**#define PSDIV\_405 0x21**

**//PLL:MDIV:48MHz:0x38:CLKDIVN:MPLLCON:**

**#define UPLL\_MDIV\_48 (0x38<<12)**

**//PDIV|SDIV == 2<<4|2<<0 == 0b0010 0010 == 0x22:UPLLCON:PLLCON**

**#define UPLL\_PSDIV\_48 0x22**

**//PLL:MDIV:200MHz:0xa1:CLKDIVN:MPLLCON:**

**#define MDIV\_200 (0xa1<<12)**

**//PDIV|SDIV == 3<<4|1<<0 == 0b0010 0010 == 0x22:UPLLCON:PLLCON**

**#define PSDIV\_200 0x31**

**添加s3c2440中断禁止代码**

#if defined(CONFIG\_S3C2410)

ldr r1,=0x3f

ldr r0,=INTSUBMSK

str r1,[r0]

#endif

**//中断属于板级配置,如果参考的芯片手册,那么就是板级配置。**

/\*上面这段代码之后添加如下s3c2440中断禁止代码: \*/

**//[31:15]Not used,INT\_AC97开中断,其余的都屏蔽中断**

**#if defined(CONFIG\_S3C2440)**

**ldr r1,=0x00007fff**

**ldr r0,=INTSUBMSK**

**str r1,[r0]**

**#endif**

**修改时钟设置:将s3c2440主频设置为405MHz**

定位到下面这块代码(166-171行左右)

/\*FCLK;HCLK;PCLK = 1:2:4\*/

/\*default FCLK is 120 MHz!\*/

ldr r0,=CLKDIVN

mov r1,#3

str r1,[r0]

#endif /\*CONFIG\_S3C2400||CONFIG\_S3C2410\*/

删除上面这块代码，并接着添加如下代码:

**//下面代码用来配置板子的时钟体系的,参考芯片手册7章节:时钟和电源管理。**

**//设置分频系数0101 == 0x5 == 1:4:8**

**#if defined(CONFIG\_S3C2440)**

**/\* FCLK:HCLK:PCLK = 1:4:8 \*/**

**ldr r0, =CLKDIVN**

**mov r1, #5**

**str r1, [r0]**

**//紧接着设置CPU为异步总线模式:根据芯片手册NOTE知2440不支持同步**

**mrc p15,0,r1,c1,c0,0 /\*read ctrl register \*/**

**orr r1,r1,#0xc0000000 /\*Asynchronous\*/**

**mcr p15,0,r1,c1,c0,0 /\*write ctrl register \*/**

**//上面已经设置了CPU时钟为405.00MHz**

**/\* now CPU clock is 405.00MHz \*/**

**mov r1,#CLK\_CTL\_BASE**

**mov r2,#UPLL\_MDIV\_48**

**add r2,r2,#UPLL\_PSDIV\_48**

**str r2,[r1,#0x08] /\*write UPLL first,48MHz\*/**

**mov r2,#MDIV\_405 /\*MPLL\_405MHz\*/**

**add r2,r2,#PSDIV\_405 /\*MPLL\_405MHz\*/**

**str r2,[r1,#0x04] /\*MPLLCON\*/**

**#else**

**/\*分频系数FCLK:HCLK:PCLK=1:2:4\*/**

**ldr r0,=CLKDIVN**

**mov r1,#3**

**str r1,[r0]**

**//紧接着设置CPU为异步总线模式**

**mrc p15,0,r1,c1,c0,0 /\*read ctrl register \*/**

**orr r1,r1,#0xc0000000 /\*Asynchronous\*/**

**mcr p15,0,r1,c1,c0,0 /\*write ctrl register \*/**

**/\* now CPU clock is 202.8MHz \*/**

**mov r1,#CLK\_CTL\_BASE**

**mov r2,#MDIV\_200 /\*MPLL\_405MHz\*/**

**add r2,r2,#PSDIV\_200 /\*MPLL\_405MHz\*/**

**str r2,[r1,#0x04] /\*MPLLCON\*/**

**#endif**

**#endif /\*CONFIG\_S3C2400||CONFIG\_S3C2410\*/**

### 体系相关arch/arm/cpu/arm920t/s3c24x0/speed.c

***片上系统的处理速度(时钟)相关修改***

将下面代码块:

#if defined(CONFIG\_S3C2400)||\

defined(CONFIG\_S3C2410)||\

defined(CONFIG\_TRAB)

改为:

#if defined(CONFIG\_S3C2400)||\

defined(CONFIG\_S3C2410)||\

**defined(CONFIG\_S3C2440)||\**

defined(CONFIG\_TRAB)

将代码块:

#elif defined(CONFIG\_S3C2410) 改为:

#elif defined(CONFIG\_S3C2410)||**defined(CONFIG\_S3C2440)**

修改函数get\_PLLCLK()

由于s3c2410与s3c2440的MPLL、UPLL计算公式不同，所以需要修改相关代码:

将下面代码块:

m = ((r & 0xff000) >> 12)+8;

p = ((r & 0x003f0) >> 4)+2;

s = r & 0x3;

修改为:

m = ((r & 0xff000) >> 12)+8;

p = ((r & 0x003f0) >> 4)+2;

s = r & 0x3;

**#if defined(CONFIG\_S3C2440)**

**if (pllreg == MPLL)**

**return ((CONFIG\_SYS\_CLK\_FREQ\*m\*2)/(p<<s));**

**else if (pllreg == UPLL) ;**

**#endif**

修改函数get\_HCLK()

同样由于s3c2410与s3c2440对分频的设置也不同，

//删除下列代码:

**return** **((**clk\_power**->**CLKDIVN **&** 0x2**)?** get\_FCLK**()/**2**:**get\_FCLK**());**

//把下面代码块:

S3C24X0\_CLOCK\_POWER **\*** const clk\_power **=** S3C24X0\_GetBase\_CLOCK\_POWER**();**

//修改为:

S3C24X0\_CLOCK\_POWER **\*** const clk\_power **=** S3C24X0\_GetBase\_CLOCK\_POWER**();**

#if defined(CONFIG\_S3C2440)

**if** **(**clk\_power**->**CLKDIVN **&** 0x6**)**

**{**

**if** **((**clk\_power**->**CLKDIVN **&** 0x6**)==**2**)**

**return** **(**get\_FCLK**()/**2**);**

**if** **((**clk\_power**->**CLKDIVN **&** 0x6**)==**6**)**

**return** **((**clk\_power**->**CAMDIVN **&** 0x100**)?** \

get\_FCLK**()/**6**:**get\_FCLK**()/**3**);**

**if** **((**clk\_power**->**CLKDIVN **&** 0x6**)==**4**)**

**return** **((**clk\_power**->**CAMDIVN **&** 0x200**)?** \

get\_FCLK**()/**8**:**get\_FCLK**()/**4**);**

**return** **(**get\_FCLK**());**

**}**

**else** **return** **(**get\_FCLK**());**

#else

**return** **((**clk\_power**->**CAMDIVN **&** 0x2**)?** \

get\_FCLK**()/**2**:**get\_FCLK**());**

#endif

### 体系相关board/samsung/mini2440/mini2440.c

***开发板相关修改***

修改PLL的配置:

将代码块:39行左右

#elif FCLK\_SPEED==1 /\*Fout=202.8MHz\*/

#define M\_MDIV 0xa1

#define M\_PDIV 0x3

#define M\_SDIV 0x1

#endif

#define USB\_CLOCK 1

改为:

#elif FCLK\_SPEED==1 /\*Fout=202.8MHz\*/

**#if defined(CONFIG\_S3C2410) /\*Fout=202.8MHz\*/**

**#define M\_MDIV 0xa1**

**#define M\_PDIV 0x3**

**#define M\_SDIV 0x1**

**#endif**

**#if defined(CONFIG\_S3C2440) /\*Fout=405.00MHz\*/**

**#define M\_MDIV 0x7f**

**#define M\_PDIV 0x2**

**#define M\_SDIV 0x1**

**#endif**

#endif

#define USB\_CLOCK 1

修改UPLL配置(47行左右)

#if USB\_CLOCK==0

#define U\_M\_MDIV 0xa1

#define U\_M\_PDIV 0x3

#define U\_M\_SDIV 0x1

#elif USB\_CLOCK==1

#define U\_M\_MDIV 0x48

#define U\_M\_PDIV 0x3

#define U\_M\_SDIV 0x2

#endif

改为:

#if USB\_CLOCK==0

#define U\_M\_MDIV 0xa1

#define U\_M\_PDIV 0x3

#define U\_M\_SDIV 0x1

#elif USB\_CLOCK==1

**#if defined(CONFIG\_S3C2410)**

**#define U\_M\_MDIV 0x48**

**#define U\_M\_PDIV 0x3**

**#endif**

**#if defined(CONFIG\_S3C2440)**

**#define U\_M\_MDIV 0x48**

**#define U\_M\_PDIV 0x2**

**#endif**

#define U\_M\_SDIV 0x2

#endif

为了引导内核，修改函数board\_init中开发板类型代码部分:

代码(106行左右):

/\* arch number of SMDK2410-Board \*/

gd->bd->bi\_arch\_number=MACH\_TYPE\_SMDK2410;

改为:

/\* arch number of SMDK2410-Board \*/

gd->bd->bi\_arch\_number=MACH\_TYPE\_**MINI2440**;

### 体系相关board/samsung/mini2440/lowlevel\_init.S:

**开发板低级初始化文件的修改**

修改REFRESH的刷新周期(123~126行左右)

将代码:

#define Trp 0x0 /\*2 clk\*/

....

#define REFCNT 1113

/\*period=15.6us, HCLK=60MHz, (2048+1-15.6\*60)\*/

改为:

**#define Trp 0x2 /\*2 clk\*/**

....

**#define REFCNT 1012**

/\*period=15.6us, HCLK=60MHz, (2048+1-15.6\*60)\*/

## 移植步骤5:选择板级配置

**make mini2440\_config**

移植步骤4:选择板级配置

make mini2440\_config

## 移植步骤6:编译并测试uboot

执行**make CROSS\_COMPILE=arm-linux-**命令，编译成功则可得到uboot映像。

最后在windows上通过JLINK把这个uboot.bin烧写到2440 Norflash中。

烧写完毕后拔掉JLINK接口，从Norflash启动2440开发板并测试我们移植的uboot是否可用：通过**help**和**printenv**命令如果能输出相关信息，表明移植的uboot可在mini2440开发板上使用。

总结：本实验移植的uboot只是最基本在mini2440开发板上可用的uboot,但功能还不完善,如环境变量不能保存,网络不可用等;**在修改代码过程中要养成添加注释的习惯**,一个好的注释有助于别人阅读代码以及自己以后修改代码，本实验只要代码有修改的地方都添加有相关注释语句。

## 实验2:uboot网卡驱动移植

359/660

### 网卡驱动移植相关理论知识

本实验在上面的实验1的基础之上进行的,

通过资料得知mini2440开发板使用的网卡芯片是DM9000，在uboot源码中已支持该芯片的驱动(**drivers/net/dm9000.c**),但并不完善,网卡工作原理。

### 修改include/configs/mini2440.h

**我们来屏蔽原来关于网卡的配置(58~60行左右)**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define CONFIG\_DRIVER\_CS8900

#define CS8900\_BASE 0x19000300

#define CS8900\_BUS16 1

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

屏蔽上面这段代码如下:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**#if 0**

#define CONFIG\_DRIVER\_CS8900

#define CS8900\_BASE 0x19000300

#define CS8900\_BUS16 1

**#endif**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**我们来添加对DM9000网卡的配置**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**#define CONFIG\_DRIVER\_DM9000 1**

**#define CONFIG\_DM9000\_USE\_16BIT 1**

**#define CONFIG\_DM9000\_BASE 0x20000300**

**#define DM9000\_IO 0x20000000**

**#define DM9000\_DATA 0x20000004**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//添加如下对ping命令的支持(63~67行左右)

**#define CONFIG\_CMD\_PING**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### 修改网卡驱动文件drivers/net/dm9000x.c

在网卡驱动drivers/net/dm9000x.c中,有一段程序试图连接网卡的MII接口,而实际MII接口并未使用、会有10秒的等待时间，然后报错，可以将此段程序注释掉:

**#if 0**

i = 0;

while (!(dm9000\_phy\_read(1) & 0x20)) { /\* autonegation complete bit \*/

udelay(1000);

i++;

if (i == 10000) {

printf("could not establish link\n");

return 0;

}

}

/\* see what we've got \*/

lnk = dm9000\_phy\_read(17) >> 12;

printf("operating at ");

switch (lnk) {

case 1:

printf("10M half duplex ");

break;

case 2:

printf("10M full duplex ");

break;

case 4:

printf("100M half duplex ");

break;

case 8:

printf("100M full duplex ");

break;

default:

printf("unknown: %d ", lnk);

break;

}

printf("mode\n");

**#endif //把上面这段代码屏蔽掉**

return 0;

}

### DM9000网卡驱动移植测试

上面过程之后，此网卡驱动程序修改完成，接下来对修改后的代码进行编译:

**# make clean**

**# make mini2440\_config**

**# make CROSS\_COMPILE=arm-linux-**

注:如果编译过程中出现错误,根据实际情况修改,

上面编译过后,通过**JTAG**口烧写到开发板的Norflash中,

然后通过串口控制台来配置网络环境变量,然后通过**ping**命令来ping服务器,如果能够ping通说明网卡驱动移植成功.

总结:uboot下的DM9000网卡驱动移植,只要求大家先实现功能,熟悉移植步骤,至于网卡工作原理,在驱动章节将详细介绍。

## 实验3:uboot-nor flash驱动移植

368/660

### Norflash驱动移植相关理论知识

Nor flash是存储器中的一种,uboot中的nor flash驱动移植就是正确实现读、写、擦除flash操作。

mini2440 nor flash芯片为SST39vf160,通过本实验需要大家掌握该芯片工作原理,实现对芯片读、写、擦除等操作。

SST39vf1601芯片工作原理:

其核心就是读、写、擦除操作对应的软件命令操作序列,数据手册第9页Table6列出了SST39vf160芯片读、写、擦除所对应的软件命令操作序列。其中Word-Program表示写flash时应满足的命令操作序列;Sector-Erase表示按扇区擦除flash时应满足的命令操作序列;Block-Erase按块擦除flash时应满足的命令操作命令;Software ID Entry表示读取芯片ID时应满足的命令操作序列。

首先看一下datasheet第2页Sector/Block-Erase Operation章节,flash是16Mbit的也就是2MB大小,按扇区操作时每个扇区Sector大小是2Kword也就是4Kbyte,按块操作时每块block大小是32Kword也就是64Kbyte。

本实验选择按block的方式操作,故2MB的flash有32个block,每个block大小为64Kbyte。

### 搜索uboot源码norflash读-写-擦除等操作移植

**找到sst160芯片支持**

**# grep -nir “sst160”**看看其他开发板有没有使用该flash芯片,以方便我们作为移植的参考.分析搜索结果,可以看出uboot中有很多开发板选用SST39vf160作为Norflash芯片,本实验以board/dave/flash.c作为移植参考。

拷贝参考的flash.c到board/mini2440

**# cp board/dave/common/flash.c board/mini2440/flash.c**

**移植函数flash\_init**

**函数flash\_init是uboot norflash驱动入口函数**,在board/mini2440/flash.c中并没有初始化函数flash\_init,需要从board/dave/B2/flash.c中拷贝函数flash\_init到board/mini2440/flash.c中。

### 修改flash初始化函数: flash\_get\_size()

addr2[CFG\_FLASH\_ADDR0]=(CFG\_FLASH\_WORD\_SIZE)0x00aa00aa;

addr2[CFG\_FLASH\_ADDR0]=(CFG\_FLASH\_WORD\_SIZE)0x00900090;

addr2[CFG\_FLASH\_ADDR1]=(CFG\_FLASH\_WORD\_SIZE)0x00550055;

改为:

addr2[CFG\_FLASH\_ADDR0]=(CFG\_FLASH\_WORD\_SIZE)**0x00aa**;

addr2[CFG\_FLASH\_ADDR0]=(CFG\_FLASH\_WORD\_SIZE)**0x0090**;

addr2[CFG\_FLASH\_ADDR1]=(CFG\_FLASH\_WORD\_SIZE)**0x0055**;

为什么要做这样的修改呢？

修改的代码部分是去读取芯片ID操作,应满足上面software ID entry命令操作序列。

**改动2:**

函数中不是别deviceID为234BH的SST39vf1601的芯片,故需添加相关代码.

在**board/mini2440/flash.c**最开始添加如下宏定义:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**#define FLASH\_SST1601 0x004b**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**改动3:**

在函数flash\_get\_size的device id匹配中添加如下代码,使之初始化时能识别该芯片:

value = addr2[CONFIG\_SYS\_FLASH\_READ1]; /\* device ID \*/

switch (value) {

case (CONFIG\_SYS\_FLASH\_WORD\_SIZE)AMD\_ID\_LV400T:

info->flash\_id += FLASH\_AM400T;

info->sector\_count = 11;

info->size = 0x00080000;

break; /\* => 0.5 MB \*/

case (CONFIG\_SYS\_FLASH\_WORD\_SIZE)AMD\_ID\_LV400B:

info->flash\_id += FLASH\_AM400B;

info->sector\_count = 11;

info->size = 0x00080000;

break; /\* => 0.5 MB \*/

**//上面代码中value为获取的ID值,switch-case语句为device-id的匹配,在switch-case语句中添加如下代码:**

value = addr2[CONFIG\_SYS\_FLASH\_READ1]; /\* device ID \*/

switch (value) {

case (CONFIG\_SYS\_FLASH\_WORD\_SIZE)AMD\_ID\_LV400T:

info->flash\_id += FLASH\_AM400T;

info->sector\_count = 11;

info->size = 0x00080000;

break; /\* => 0.5 MB \*/

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**case (CONFIG\_SYS\_FLASH\_WORD\_SIZE)SST\_ID\_xF1601:**

**info->flash\_id += FLASH\_SST1601;**

**info->sector\_count = 32;**

**info->size = 0x00200000;**

**break; /\* => 2 MB \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

case (CONFIG\_SYS\_FLASH\_WORD\_SIZE)AMD\_ID\_LV400B:

info->flash\_id += FLASH\_AM400B;

info->sector\_count = 11;

info->size = 0x00080000;

break; /\* => 0.5 MB \*/

**改动4:**

函数flash\_get\_size倒数第4行代码左右:

/\*

\* Prevent writes to uninitialized FLASH.

\*/

if (info->flash\_id != FLASH\_UNKNOWN) {

addr2 = (CONFIG\_SYS\_FLASH\_WORD\_SIZE \*)info->start[0];

**\*addr2 = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00F000F0; /\* reset bank** \*/

}

return (info->size);

}

改为:

/\*

\* Prevent writes to uninitialized FLASH.

\*/

if (info->flash\_id != FLASH\_UNKNOWN) {

addr2 = (CONFIG\_SYS\_FLASH\_WORD\_SIZE \*)info->start[0];

**\*addr2 = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00F0; /\* reset bank \*/**

}

return (info->size);

}

**改动3:**

在函数get\_flash\_size中注释语句/\*setup sector start address table \*/下面的代码是建立flash的block的分区地址表,我们的block的大小是64KB(0x10000),从语句info->start[i] = base + (i\*0x00010000);可以看出该驱动确实是按照每块64KB来建立分区表的故不用修改.再次检查flash\_get\_offsets函数,发现里面的偏移也是64KB,故也不用修改.

### 修改函数:write\_buff()

把该函数中的所有**CONFIG\_B2**改为**CONFIG\_MINI2440**

### 修改函数:flash\_print\_info()

该函数主要就是打印输出一些flash信息,如芯片型号、大小、分区信息等,uboot命令finfo就是调用的该函数.我们做的主要就是在switch-case中添加stt1601的打印支持:

switch (info->flash\_id & FLASH\_TYPEMASK) {

case FLASH\_AM400B: printf ("AM29LV400B (4 Mbit, bottom boot sect)\n");

break;

case FLASH\_AM400T: printf ("AM29LV400T (4 Mbit, top boot sector)\n");

break;

case FLASH\_AM800B: printf ("AM29LV800B (8 Mbit, bottom boot sect)\n");

break;

case FLASH\_AM800T: printf ("AM29LV800T (8 Mbit, top boot sector)\n");

break;

添加如下:

switch (info->flash\_id & FLASH\_TYPEMASK) {

case FLASH\_AM400B: printf ("AM29LV400B (4 Mbit, bottom boot sect)\n");

break;

case FLASH\_AM400T: printf ("AM29LV400T (4 Mbit, top boot sector)\n");

break;

case FLASH\_AM800B: printf ("AM29LV800B (8 Mbit, bottom boot sect)\n");

break;

**case FLASH\_SST1601:**

**printf ("SST39LF/VF1601 (16 Mbit, uniform.sector size)\n");**

**break;**

case FLASH\_AM800T: printf ("AM29LV800T (8 Mbit, top boot sector)\n");

break;

### 添加宏定义:board/mini2440/flash.c

此时去编译uboot的时候会报一些错误，如CFG\_FLASH\_ADDR0没有定义，于是我们在board/mini2440/flash.c开始部分添加如下宏定义,参考include/configs/B2.h文件,

**#define CFG\_FLASH\_ADDR0 0x5555**

**#define CFG\_FLASH\_ADDR1 0x2aaa**

**#define CFG\_FLASH\_WORD\_SIZE unsigned short**

**#define CFG\_FLASH\_READ0 0x0000 /\*A0 = 0\*/**

**#define CFG\_FLASH\_READ1 0x0001 /\*A0 = 1\*/**

**#define CFG\_FLASH\_READ2 0x0002 /\*A0 = 2\*/**

**修改软硬件配置文件include/configs/mini2440.h,屏蔽原来有关flash的宏定义**

#define CONFIG\_AMD\_LV400 1

**#if 0**

#define CONFIG\_AMD\_LV800 1

**#endif**

**修改软硬件配置文件include/configs/mini2440.h,添加新的有关flash的宏定义**

#define PHYS\_FLASH\_SIZE 0x00200000 /\*2MB\*/

#define CFG\_MAS\_FLASH\_SECT (32)

#define CONFIG\_ENV\_ADDR (CFG\_FLASH\_BASE+0x030000)

//把环境变量首地址设置为0x030000,当然这个是可以修改的,而环境变量大小CONFIG\_ENV\_SIZE 0x10000 (64KB)

#define CFG\_MONITOR\_BASE TEXT\_BASE

#define CFG\_MONITOR\_LEN (256\*1024) /\*reserve 256KB for monitor\*/

### 修改flash的读函数flash\_erase()

### 修改flash的写函数write\_word()

for (i=0; i<4/sizeof(CONFIG\_SYS\_FLASH\_WORD\_SIZE); i++)

{

addr2[CONFIG\_SYS\_FLASH\_ADDR0] = **(CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00AA00AA;**

**addr2[CONFIG\_SYS\_FLASH\_ADDR1] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00550055;**

**addr2[CONFIG\_SYS\_FLASH\_ADDR0] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00A000A0;**

dest2[i] = data2[i];

/\* re-enable interrupts if necessary \*/

if (flag)

enable\_interrupts();

/\* data polling for D7 \*/

start = get\_timer (0);

**while ((dest2[i] & (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00800080) !=**

**(data2[i] & (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00800080)) {**

if (get\_timer(start) > CONFIG\_SYS\_FLASH\_WRITE\_TOUT) {

return (1);

}

}

}

return (0);

}

改为:

for (i=0; i<4/sizeof(CONFIG\_SYS\_FLASH\_WORD\_SIZE); i++)

{

**addr2[CONFIG\_SYS\_FLASH\_ADDR0] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00AA;**

**addr2[CONFIG\_SYS\_FLASH\_ADDR1] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x0055;**

**addr2[CONFIG\_SYS\_FLASH\_ADDR0] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00A0;**

dest2[i] = data2[i];

/\* re-enable interrupts if necessary \*/

if (flag)

enable\_interrupts();

/\* data polling for D7 \*/

start = get\_timer (0);

**while ((dest2[i] & (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x0080) !=**

**(data2[i] & (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x0080)) {**

if (get\_timer(start) > CONFIG\_SYS\_FLASH\_WRITE\_TOUT) {

return (1);

}

}

}

return (0);

}

**修改flash的擦除函数flash\_erase()**

**本实验选择按块方式擦除芯片**

**改动一:**

根据block-erase命令操作序列,将代码:

/\* Start erase on unprotected sectors \*/

for (sect = s\_first; sect<=s\_last; sect++) {

if (info->protect[sect] == 0) { /\* not protected \*/

addr2 = (CONFIG\_SYS\_FLASH\_WORD\_SIZE \*)(info->start[sect]);

if ((info->flash\_id & FLASH\_VENDMASK) == FLASH\_MAN\_SST) {

addr[CONFIG\_SYS\_FLASH\_ADDR0] = **(CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00AA00AA;**

**addr[CONFIG\_SYS\_FLASH\_ADDR1] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00550055;**

**addr[CONFIG\_SYS\_FLASH\_ADDR0] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00800080;**

**addr[CONFIG\_SYS\_FLASH\_ADDR0] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00AA00AA;**

**addr[CONFIG\_SYS\_FLASH\_ADDR1] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00550055;**

**addr2[0] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00500050; /\* block erase \*/**

for (i=0; i<50; i++)

udelay(1000); /\* wait 1 ms \*/

} else {

if (sect == s\_first) {

**addr[CONFIG\_SYS\_FLASH\_ADDR0] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00AA00AA;**

**addr[CONFIG\_SYS\_FLASH\_ADDR1] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00550055;**

**addr[CONFIG\_SYS\_FLASH\_ADDR0] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00800080;**

**addr[CONFIG\_SYS\_FLASH\_ADDR0] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00AA00AA;**

**addr[CONFIG\_SYS\_FLASH\_ADDR1] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00550055;**

**}**

**addr2[0] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00300030; /\* sector erase \*/**

}

l\_sect = sect;

}

}

改为:

/\* Start erase on unprotected sectors \*/

for (sect = s\_first; sect<=s\_last; sect++) {

if (info->protect[sect] == 0) { /\* not protected \*/

addr2 = (CONFIG\_SYS\_FLASH\_WORD\_SIZE \*)(info->start[sect]);

if ((info->flash\_id & FLASH\_VENDMASK) == FLASH\_MAN\_SST) {

addr[CONFIG\_SYS\_FLASH\_ADDR0] = **(CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00AA;**

**addr[CONFIG\_SYS\_FLASH\_ADDR1] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x0055;**

**addr[CONFIG\_SYS\_FLASH\_ADDR0] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x0080;**

**addr[CONFIG\_SYS\_FLASH\_ADDR0] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00AA;**

**addr[CONFIG\_SYS\_FLASH\_ADDR1] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x0055;**

**addr2[0] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x0050; /\* block erase \*/**

for (i=0; i<50; i++)

udelay(1000); /\* wait 1 ms \*/

} else {

if (sect == s\_first) {

**addr[CONFIG\_SYS\_FLASH\_ADDR0] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00AA;**

**addr[CONFIG\_SYS\_FLASH\_ADDR1] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x0055;**

**addr[CONFIG\_SYS\_FLASH\_ADDR0] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x0080;**

**addr[CONFIG\_SYS\_FLASH\_ADDR0] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00AA;**

**addr[CONFIG\_SYS\_FLASH\_ADDR1] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x0055;**

**}**

**addr2[0] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x0030; /\* sector erase \*/**

}

l\_sect = sect;

}

}

**改动二:**

start = get\_timer (0);

last = start;

addr = (CONFIG\_SYS\_FLASH\_WORD\_SIZE \*)(info->start[l\_sect]);

**while ((addr[0] & (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00800080) != (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00800080) {**

if ((now = get\_timer(start)) > CONFIG\_SYS\_FLASH\_ERASE\_TOUT) {

printf ("Timeout\n");

return 1;

}

/\* show that we're waiting \*/

if ((now - last) > 1000) { /\* every second \*/

putc ('.');

last = now;

}

}

DONE:

/\* reset to read mode \*/

addr = (CONFIG\_SYS\_FLASH\_WORD\_SIZE \*)info->start[0];

**addr[0] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00F000F0; /\* reset bank \*/**

printf (" done\n");

return 0;

}

改为:

start = get\_timer (0);

last = start;

addr = (CONFIG\_SYS\_FLASH\_WORD\_SIZE \*)(info->start[l\_sect]);

**while ((addr[0]&(CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x0080) != (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x0080) {**

if ((now = get\_timer(start)) > CONFIG\_SYS\_FLASH\_ERASE\_TOUT) {

printf ("Timeout\n");

return 1;

}

/\* show that we're waiting \*/

if ((now - last) > 1000) { /\* every second \*/

putc ('.');

last = now;

}

}

DONE:

/\* reset to read mode \*/

addr = (CONFIG\_SYS\_FLASH\_WORD\_SIZE \*)info->start[0];

**addr[0] = (CONFIG\_SYS\_FLASH\_WORD\_SIZE)0x00F0; /\* reset bank \*/**

printf (" done\n");

return 0;

}

### Norflash驱动移植测试

至此mini2440 nor flash驱动移植的主体工作已经完成,接下来编译移植后的uboot源代码.

# make clean

# make mini2440\_config

# make CROSS\_COMPILE=arm-linux-

注:如果编译过程中出现错误,根据实际错误去修改

然后通过JTAG口将编译出的uboot.bin烧写进Norflash中,

然后测试flash分区信息是否正确

开发板上电后查看串口控制台输出,可以看到flash大小由原来的512KB变为2MB,

然后使用uboot命令flinfo查看分区是否正确,通过输出结果看出,flash大小为2MB,一共有32个block,每个block大小为0x10000(64KB),与实际相符。

最后测试flash是否可写

使用setenv命令添加一个环境变量并保存:

# setenv name mini2440

# saveenv

重启开发板并用printenv命令查看环境变量name是否依然存在,存在则说明可写.

**Norflash驱动移植总结:**

\*\*熟悉nor flash芯片工作原理

\*\*为移植找一份参考代码,阅读参考代码

\*\*移植参考代码到自己开发板

\*\*测试移植是否成功

## U-boot 添加自定义命令

//cmd\_hello.c 文件

#include <common.h>

#include <watchdog.h>

#include <command.h>

#include <image.h>

#include <malloc.h>

#include <zlib.h>

#include <bzlib.h>

#include <environment.h>

#include <asm/byteorder.h>

int do\_hello **(**cmd\_tbl\_t **\***cmdtp**,** int flag**,** int argc**,** char **\***argv**[])**

**{**

int i **=** 0**;**

printf**(**"Hello CoCo! %d\n"**,** argc**);**

**for** **(**i**=**0**;** i**<**argc**;** i**++)** **{**

printf**(**"argv[%d]: %s\n"**,** i**,** argv**[**i**]);**

**}**

**return** 0**;**

**}**

U\_BOOT\_CMD**(<**span style**=**"white-space:pre"**>** **</**span**>**//命令名， 参数个数， 是否重复， 对应执行的函数， 简短帮助信息， 详细的帮助信息

hello**,** CFG\_MAXARGS**,** 1**,** do\_hello**,**

"hello - just for test\n"**,**

"hello, long help..............\n"

**);**

1. 把cmd\_hello.c文件放到 common目录下，

2. 修改该目录下的Makefile文件

在目标COBJS后面添加 cmd\_hello.o 即可

3. make

## u-boot 板级配置配置文件

## u-boot 最小系统相关文件（即一个能编译的 u-boot 需要哪些文件）

# 专题03-嵌入式Linux内核制作

## 核心理论

[主目录](#_目录)

## Linux内核概念

[主目录](#_目录)

Linux体系结构/Linux系统架构

Linux系统分为用户空间和内核空间，其中用户空间有用户程序、C库、配置文件等，其中内核空间有系统调用接口、内核代码、体系结构相关代码等。

Linux系统划分用户空间和内核空间原因？

Linux系统利用了CPU的这一特性，使用了其中的两级来分别运行Linux内核与应用程序，这样是操作系统本身得到充分的保护。例如：

X86，用户代码运行在ring3，内核代码运行在ring0。

Linux内核架构

系统调用接口、进程管理子系统、内存管理子系统、虚拟文件系统、网络协议栈、设备驱动、体系结构相关代码。

Linux内核源代码下载：<https://www.kernel.org/>

Linux内核代码管理：sourceinsight或vim+插件

Linux内核目录结构

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放在同一个子目录下，使得程序更具可读性。

Linux内核目录结构：arch目录

内核所支持的每种CPU体系，在该目录下都有对应的子目录。每个CPU子目录又进一步分解为boot, mm, kernel等子目录。

Linux内核目录结构：fs目录

存放各种文件系统的实现代码。每个子目录对应一种文件系统的实现，公用的源程序用于实现虚拟文件系统VFS。

## Linux的x86平台内核升级

[主目录](#_目录)

为什么要配置内核：选出需要的，去掉不要的。

内核配置(x86)

内核配置：make config, make menuconfig, make xconfig

General setup：system V IPC – system V进程间通信（IPC）支持，许多程序需要这个功能建议选。

内核配置参考文件

内核配置参考文件位于顶层：arch/arm/configs; ls –a .config

编译内核（x86）

编译内核（x86）：make bzImage, make bzImage V=1, 编译好的内核位于arch/x86/boot/目录下。

编译内核模块（x86）

编译内核模块（x86）:make modules

这一步所需的时间较长，为了加快编译速度，我们可以加上“-j”选项，后面跟的数字是jobsnum ，建议设置为 CPU 核心数 + 1，jobsnum值可以由命令“ cat /proc/cpuinfo |sed -n '/^processor/p' |wc -l”获得。

补充：（以下等价于#make -jx）

# make –jx //多线程编译内核模块,x为你设置的线程数

安装内核模块（x86）

安装内核模块（x86）:make modules\_install此步骤将编译好的内核模块从源代码目录拷贝至/lib/modules/目录下。

制作initramdisk（x86）:

cd /lib/modules && mkinitrd initrd-$version $version

PC机Linux内核升级（x86）

:cp arch/x86/boot/bzImage /boot/ && \

cp /lib/modules/inird-$version /boot/ && \

vi /etc/grub.conf 或 vi lilo.conf

:/etc/grub.conf内容如下:

修改内核文件权限：将/boot/下的新的内核文件改为可执行。

清理内核源代码中间文件：make clean; make distclean

加载模块/卸载模块

:insmod hello.ko, modprobe hello.ko; rmmod `lsmod | grep “hello”`

## 嵌入式Linux内核制作

[主目录](#_目录)

清除原有配置与中间文件：（x86）make distclean; （arm）make distclean

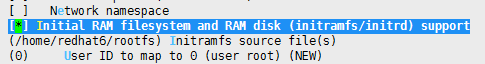
配置内核

:x86平台: make menuconfig ARCH=x86 (默认为x86平台)

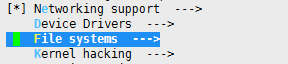
:arm平台: make menuconfig ARCH=arm

initramfs方式根文件系统的内核配置：

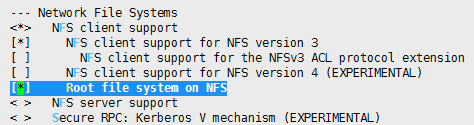




NFS方式根文件系统的内核配置：







注意此处需要一个为内核制作头部的工具：cp uboot/tools/mkimage /bin

编译内核

:x86平台: make bzImage

:arm平台: make uImage ARCH=arm CROSS\_COMPILE=arm-linux-

# 专题04-嵌入式文件系统

1. [主目录](#_目录)

## 核心理论

[主目录](#_目录)

虚拟文件系统(VFS):Virtual FileSystem

VFS隐藏各种文件系统的具体细节，为文件操作提供统一的操作接口。

系统调用接口（SCI）:System Call Interface

SCI层为用户空间提供了一套标准的系统调用函数来访问Linux内核，搭起了用户空间到内核空间的桥梁。

进程管理

进程管理的重点是创建进程（fork, exec）,停止进程（kill, exit）,并控制它们之间的通信（signal或POSIX机制）。

进程管理还包括控制活动进程如何共享CPU，即进程调度。

内存管理

内存管理的主要作用是控制多个进程安全地共享内存区域。

网络协议栈

内核协议栈为Linux提供了丰富的网络协议实现。

## 嵌入式根文件系统制作

[主目录](#_目录)

创建根文件目录

# mkdir rootfs &&cd rootfs && mkdir –p bin dev usr etc lib proc sbin sys mnt tmp var

# mkdir –p usr/bin usr/lib usr/sbin lib/modules

创建设备文件

# mknod –m 666 console c 5 1

# mknod –m 666 null c 1 3

加入配置文件

# tar –xzf etc.tar.gz

# mv etc/\* rootfs/etc/ -rf

往根文件系统中添加内核模块

# make modules ARCH=arm CROSS\_COMPILE=arm-linux-

# make modules\_install ARCH=arm INSTALL\_MOD\_PATH=/home/redhat6/rootfs

编译和安装busybox

# make menuconfig ARCH=arm

Busybox Settings >>Gerneral Configuration>>

Busybox Settings >>Build Options >>

[\*] Build BusyBox as a static binary (no shared libs)

() Cross Compiler prefix(arm-linux-)

Busybox Settings >>Debugging Options >>

Busybox Settings >>Installation Options >>

[\*] Don’t use /usr

() Busybox Installation prefix(/home/redhat6/rootfs)

选中 “Don‘t use /usr”, 选中该项可以避免busybox 被安装到宿主系统的  
/usr目录下，破坏宿主系统；该选项表明编译后的busybox的安装位置

Busybox Settings >>Busybox LibraryTuning>>

# make ARCH=arm CROSS\_COMPILE=arm-linux-

# make install

链接init文件:

**# ln -s ./bin/busybox ./init**

## 嵌入式文件系统

[主目录](#_目录)

文件系统类型：根据存储设备的硬件特性、产品的系统需求，不同的文件系统类型有不同的应用场合。在嵌入式Linux应用中，主要的存储设备为RAM和FLASH，常用的基于存储设备的文件系统类型包括：jffs2, yaffs2, ubifs, ramdisk等等。

基于nandflash的文件系统：yaffs2, ubifs

基于norflash的文件系统：jffs2

基于内存的文件系统：ramdisk(事先分配好大小), initramfs(根据实际使用量动态分配大小)

基于网络的文件系统：nfs(开发阶段经常使用)

## 使用嵌入式initramfs文件系统

[主目录](#_目录)

链接busybox文件到嵌入式根目录的init

# ln –s rootfs/bin/busybox rootfs/init

配置Linux内核以支持initramfs

# make menuconfig ARCH=arm

General setup >>

[\*] Initial RAM Filesystem and RAM disk (initramfs/initrd) support

(/home/redhat6/rootfs) Initramfs source file(s)

# make uImage ARCH=arm CROSS\_COMPILE=arm-linux-

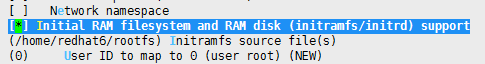
# cp linux/arch/arm/boot/uImage /tftpboot

# /etc/init.d/xinetd restart

注意：内核模块已经在上面的modules中制作添加好了。工具包已经在上面busybox中制作好了。

initramfs方式根文件系统的内核配置：





配置开发板启动参数并启动开发板

Nandflash方式启动开发板uboot命令行，然后设置initramfs启动参数：

# setenv bootargs noinitrd console=ttySAC0,115200

# tftp c0008000 uImage

# bootm c0008000

# setenv bootcmd tftp c0008000 \; bootm c0008000

## 使用嵌入式NFS文件系统

[主目录](#_目录)

链接busybox文件到嵌入式根目录的init

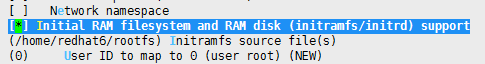
# ln –s rootfs/bin/busybox rootfs/init

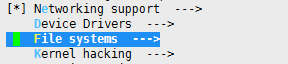
配置Linux内核以支持NFS

# make menuconfig ARCH=arm

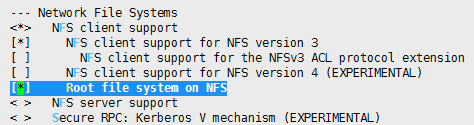
取消initramfs的支持











编译Linux内核uImage

# make uImage ARCH=arm CROSS\_COMPILE=arm-linux-

# cp arch/arm/boot/uImage /tftpboot

检查Linux的NFS服务器

# cat /etc/exports

home/redhat6/rootfs \*(rw,sync,no\_root\_squash)

# /etc/init.d/nfs restart

配置开发板启动参数并启动开发板（NFS）

第一步nandflash方式启动开发板进入uboot命令行

# setenv bootargs noinitrd \

console=ttySAC0,115200 \

init=/init \

root=/dev/nfs rw \

nfsroot=192.168.1.10:/home/redhat6/rootfs \

ip=192.168.1.15:192.168.1.15:192.168.1.10:192.168.1.1:255.255.255.0::eth0:off

# tftp c0008000 uImage

# bootm c0008000

# 专题05-内核模块开发

## 核心理论

内核模块能够帮助我们更加有效节省内存空间。

加载模块/卸载模块

:insmod hello.ko, modprobe hello.ko; rmmod `lsmod | grep “hello”`

## 模块模板代码hello.ko

[主目录](#_目录)

#include <linux/init.h>

#include <linux/module.h>

MODULE\_LICENSE**(**"GPL"**);**

static int hello\_init **(**void**)**

**{**

printk **(**KERN\_WARNING"hello,world\n"**);**

**return** 0**;**

**}**

static void hello\_exit **(**void**)**

**{**

printk **(**KERN\_WARNING"goodbye,world\n"**);**

**return** **;**

**}**

module\_init **(**hello\_init**);**

module\_exit **(**hello\_exit**);**

神总结：各位吃瓜群众请注意！编译自己的模块之前相应内核必须也要编译过！！！

**[**root**@**redhat6 linux-mini2440**]**# make uImage ARCH**=**arm CROSS\_COMPILE**=**arm-linux-

**......**

Kernel**:** arch**/**arm**/**boot**/**zImage is ready

UIMAGE arch**/**arm**/**boot**/**uImage

Image Name**:** Linux-2.6.32.2-FriendlyARM

Created**:** Sun Oct 30 17**:**06**:**02 2016

Image Type**:** ARM Linux Kernel Image **(**uncompressed**)**

Data Size**:** 2287124 Bytes **=** 2233**.**52 kB **=** 2**.**18 MB

Load Address**:** 30008000

Entry Point**:** 30008000

Image arch**/**arm**/**boot**/**uImage is ready

**[**root**@**redhat6 linux-mini2440**]**#

## 模块hello.ko对应的Makefile

[主目录](#_目录)

#此参数为要编译的独立模块main.ko文件

obj-m**:=**main.o

#此参数为main.ko的其他依赖文件

#main-objs:=hello.o 1.o 2.o 3.o

KDIR**:=**/home/redhat6/linux-mini2440

all**:**

make modules -C $(KDIR) M=$(PWD) ARCH=arm CROSS\_COMPILE=arm-linux-

clean**:**

rm -f \*.o \*.ko \*.mod\* \*.sym\* \*.ord\*

## module license 'unspecified' taints kernel

单一的一个文件hello.c是可以编译成功（hello.ko）并且运行的。

#但是增加了两个文件后（f1.c和f1.h）。编译成功但是insmod时出现：

#module license 'unspecified' taints kernel

#是Makefile写的有问题

#hello.c和f1.c应该编译成两个模块再链接成新的模块就ok了

#源文件列表:hello.c f1.c f1.h

obj-m +**=**main.o #这里我们链接成最终的main.ko模块

main-objs **:=**hello.o f1.o

KDIR**:=**/home/redhat6/linux-mini2440

default**:**

make -C $(KDIR) M**=**$(PWD) modules CROSS\_COMPILE=arm-linux- ARCH=arm

make clean

cp \*.ko /home/NFS

clean**:**

rm -rf \*.mod\* \*.order \*.symvers \*.o .dht11\* .tmp\* .\*.cmd

display**:**

#echo $(obj-m)

编译测试模块并拷贝至开发板

[root@redhat6 hello]# ls

hello.c Makefile

[root@redhat6 hello]# make

make modules -C /home/redhat6/linux-mini2440M**=**/home/redhat6/hello ARCH=arm CROSS\_COMPILE=arm-linux-

make[1]**:** Entering directory `/home/redhat6/linux-mini2440'

CC [M] /home/redhat6/hello/hello.o

Building modules, stage 2.

MODPOST 1 modules

CC /home/redhat6/hello/hello.mod.o

LD [M] /home/redhat6/hello/hello.ko

make[1]**:** Leaving directory `/home/redhat6/linux-mini2440'

[root@redhat6 hello]# ls

hello.c hello.mod.c hello.o modules.order

hello.ko hello.mod.o Makefile Module.symvers

[root@redhat6 hello]#cp hello.ko /home/redhat6/rootfs

[root@redhat6 hello]#

内核打印

在<linux/kernel.h>中定义了8种记录级别。按照优先级递减的顺序分别是：

KERN\_EMERG //0-用于紧急消息，常常是那些崩溃前的消息”

KERN\_ALERT //1-需要立刻行动的消息”

KERN\_CRIT //2-严重情况”

KERN\_ERR //3-错误情况”

KERN\_WARNING //4-有问题的警告”

KERN\_NOTICE //5-正常情况，但是仍然值得注意”

KERN\_INFO //6-信息型消息”

KERN\_DEBUG //7-用作调试消息”

内核模块可选项

MODULE\_LICENSE**(**“GPL”**);** //许可证声明

MODULE\_AUTHOR**(**“seafly”**);** //作者声明

MODULE\_DESCRIPTION**(**“hello world”**);** //模块描述

MODULE\_VERSION**(**“V1.0”**);** //模块版本

MODULE\_ALIAS**(**“hello\_world”**);** //模块别名

模块参数：module\_param (name, type, perm);

Name是模块参数名称，type是这个参数类型（bool,int,charp），perm是模块参数的访问权限（~~S\_IWUGO~~, S\_IRUGO, ...）

内核符号导出

//EXPORT\_SYMBOL(符号名);

//EXPORT\_SYMBOL\_GPL(符号名);

//其中EXPORT\_SYMBOL\_GPL只能用于包含GPL许可证的模块。

模块版本不匹配

确保编译内核模块时，所依赖的内核代码版本等同于当前正在运行的内核。

使用 modprobe --force-modversion强行插入。 查看当前内核版本：uname -r

内核模块与普通应用程序对比

对比应用程序，内核模块具有以下不同：应用程序是从头（main）到尾执行任务，执行结束后从内存中消失。内核模块则是先在内核中注册自己以便服务于将来的某个请求，然后它的初始化函数结束，此时模块仍然存在于内核中，直到卸载函数被调用，模块才从内核中消失。

## 模块测试-模块参数传入

[主目录](#_目录)

int a**;**

//如何传入参数：insmod hello.ko a=10

module\_param**(**a**,**int**,**S\_IRUGO**|**S\_IWUGO**);**

static int hello\_init **(**void**)**

**{**

printk **(**KERN\_WARNING"hello,world\n"**);**

printk **(**"a = %d\n"**,** a**);**

**return** 0**;**

**}**

编译时出现如下错误：

[root@redhat6 hello]# make

make modules -C /home/redhat6/linux-mini2440 M**=**/home/redhat6/hello ARCH=arm CROSS\_COMPILE=arm-linux-

make[1]**:** Entering directory `/home/redhat6/linux-mini2440'

CC [M] /home/redhat6/hello/hello.o

/home/redhat6/hello/hello.c**:**8: error: negative width in bit-field '<anonymous>'

make[2]**:** \*\*\* [/home/redhat6/hello/hello.o] Error 1

make[1]**:** \*\*\* [\_module\_/home/redhat6/hello] Error 2

make[1]**:** Leaving directory `/home/redhat6/linux-mini2440'

make**:** \*\*\* [all] Error 2

[root@redhat6 hello]#

修改错误如下：

int a**;**

module\_param**(**a**,**int**,**S\_IRUGO**);**

//如何传入参数：insmod hello.ko a=10

//最后发现是module\_param申明中有关权限问题，

//这个权限是仅不可写的。所以只要不可写就行。

static int hello\_init **(**void**)**

**{**

printk **(**KERN\_WARNING"hello,world\n"**);**

printk **(**"a = %d\n"**,** a**);**

**return** 0**;**

**}**

## 内核模块参数范例

[主目录](#_目录)

范例代码

#include <linux/init.h>

#include <linux/module.h>

MODULE\_LICENSE**(**"GPL"**);**

int a**;** //参数变量定义时千万别初始化

char **\***ptr**;** //参数变量定义时千万别初始化

module\_param**(**a**,**int**,**S\_IRUGO**);**

module\_param**(**ptr**,**charp**,**S\_IRUGO**);**

static int hello\_init **(**void**)**

**{**//通过安装模块时传入参数: insmod hello.ko a=60 ptr=”nice to meet you!”

printk **(**KERN\_WARNING"hello,world\n"**);**

printk **(**"a = %d\n"**,** a**);**

printk **(**"ptr = %s\n"**,** ptr**);**

**return** 0**;**

**}**

static void hello\_exit **(**void**)**

**{**

printk **(**KERN\_WARNING"goodbye,world\n"**);**

**return** **;**

**}**

module\_init **(**hello\_init**);**

module\_exit **(**hello\_exit**);**

编译模块并拷贝到开发板

[root@redhat6 hello]# ls

hello.c Makefile

[root@redhat6 hello]# make

make modules -C /home/redhat6/linux-mini2440 M**=**/home/redhat6/hello ARCH=arm CROSS\_COMPILE=arm-linux-

make[1]**:** Entering directory `/home/redhat6/linux-mini2440'

CC [M] /home/redhat6/hello/hello.o

Building modules, stage 2.

MODPOST 1 modules

CC /home/redhat6/hello/hello.mod.o

LD [M] /home/redhat6/hello/hello.ko

make[1]**:** Leaving directory `/home/redhat6/linux-mini2440'

[root@redhat6 hello]# cp hello.ko /home/redhat6/rootfs/

cp**:** overwrite `/home/redhat6/rootfs/hello.ko'? y

[root@redhat6 hello]#

开发板模块参数测试

# insmod hello.ko a=60 ptr=”nice to meet you!”

Hello world

a **=** 60

ptr **=** nice to meet you!

## 内核模块符号输出

[主目录](#_目录)

调用模块源文件:hello.ko

#include <linux/init.h>

#include <linux/module.h>

MODULE\_LICENSE**(**"GPL"**);**

extern int add **(**int**,**int**);**//调用外部模块的函数时候必须先声明

static int hello\_init **(**void**)**

**{**

printk **(**KERN\_WARNING"hello,world\n"**);**

printk **(**"a=6,b=26,add(a,b) = %d\n"**,** add**(**6**,**26**));**

**return** 0**;**

**}**

static void hello\_exit **(**void**)**

**{**

printk **(**KERN\_WARNING"goodbye,world\n"**);**

**return** **;**

**}**

module\_init **(**hello\_init**);**

module\_exit **(**hello\_exit**)**

被调用模块源文件:add.ko

#include <linux/init.h>

#include <linux/module.h>

//声明为外部可调用函数extern并独立实现该函数文件

extern int add **(**int a**,** int b**)**

**{**

**return** a**+**b**;**

**}**

static int hello\_init **(**void**)**

**{**

**return** 0**;**

**}**

static void hello\_exit **(**void**)**

**{**

**return** **;**

**}**

EXPORT\_SYMBOL**(**add**);**//将外部可调用函数通过符号导出

module\_init **(**hello\_init**);**

module\_exit **(**hello\_exit**);**

修改Makefile文件以增加add.ko模块

obj-m**:=**hello.o add.o

KDIR**:=**/home/redhat6/linux-mini2440

all**:**

make modules-C $(KDIR)M=$(PWD)ARCH=arm CROSS\_COMPILE=arm-linux-

clean**:**

rm -f \*.o \*.ko \*.mod\* \*.sym\* \*.ord\*

编译模块源程序并将模块拷贝至开发板进行测试

[root@redhat6 hello]# ls

add.c hello.c Makefile

[root@redhat6 hello]# make

make modules -C /home/redhat6/linux-mini2440 M**=**/home/redhat6/hello ARCH=arm CROSS\_COMPILE=arm-linux-

make[1]**:** Entering directory `/home/redhat6/linux-mini2440'

CC [M] /home/redhat6/hello/hello.o

CC [M] /home/redhat6/hello/add.o

Building modules, stage 2.

MODPOST 2 modules

CC /home/redhat6/hello/add.mod.o

LD [M] /home/redhat6/hello/add.ko

CC /home/redhat6/hello/hello.mod.o

LD [M] /home/redhat6/hello/hello.ko

make[1]**:** Leaving directory `/home/redhat6/linux-mini2440'

[root@redhat6 hello]# cp hello.ko add.ko /home/redhat6/rootfs/

cp**:** overwrite `/home/redhat6/rootfs/hello.ko'? y

cp**:** overwrite `/home/redhat6/rootfs/add.ko'? y

[root@redhat6 hello]#

开发板模块测试-符号导出测试

# insmod add.ko

# insmod hello.ko

Hello,world

a=6,b=26,add(a,b) = 32符号导出测试成功!

## 模块测试-关于模块的依赖文件问题

[主目录](#_目录)

测试模块hello.ko源文件

一个驱动模块依赖多个C文件的Makefile写法

测试模块hello.ko的依赖文件add.o源文件

测试模块-关于有依赖文件的模块文件Makefile文件修改

测试模块-编译模块并拷贝到开发板

开发板模块依赖测试

# 测试失败TEST\_ERROR！！！暂时搁置...

## 多个模块依赖编译成一个.ko文件Makefile

范例1:

obj-m:=**main.o**

**main**-objs:=f1.o f2.o f3.o

范例2:

obj-m +=**main.o**

**main**-objs +=f1.o f2.o f3.o

# 专题06-Linux内核子系统

[主目录](#_目录)

## Linux内存管理子系统

Linux子系统模型（7个子系统）：

SCI(System Call Interface)

PM(process Management)

MM(memory Management)

VFS(Virtual File System)

NS(Network Stack)

DD(Device Drivers)

Arch(体系结构相关代码)

嵌入式硬件平台中最重要的2个硬件：CPU以及内存。它们非常重要却也非常有限。

内存管理子系统职能：（金钱管理职能）

虚拟地址与物理地址映射的管理（虚拟金钱与现金映射的管理）

物理内存分配的管理。（现金分配的管理）

地址映射管理：分为

虚拟地址空间分布（数字金额分配）

虚拟地址转化为物理地址。（数字金额转化为现金）

虚拟地址空间大小由硬件决定，CPU为32位则可以访问空间为2的32次方（4GB）

虚拟地址空间分布：0G-3G（用户空间）（用户），3G-4G（内核空间）（管理员）

内核空间分为4个区(3G~4G):直接映射区-3G~(3G+896MB)、vmalloc区、永久映射区、固定映射区。

为什么会划分这些呢？就是因为它们的映射方式不同。

直接映射区：物理地址=虚拟地址-3GB

Vmalloc区：既可以访问到高低端区域也能访问到高端地址。高低端区域都能访问。

永久映射区：固定用来访问高端内存的。

固定映射区：用来访问一些固定的特殊的寄存器。

Linux内核虚拟地址转化为物理地址

896MB以下为低端内存，896MB以上为高端内存。

Linux物理内存分配

Linux是实行虚拟地址的管理方式。例如100个座位看电影，电影票和座位的关系，当我们拿到票的时候如果你不去电影院就不一定有座位（物理地址）。所以Linux中使用malloc分配的时候，只有当你去访问虚拟地址的时候才会去给你分配实实在在的内存。

用户空间（malloc）时会事先分配给我们地址（电影票），当我们真正去访问这个虚拟地址时，会把vaddr转化为物理地址，后来Linux内核会发现原来没有vaddr所对应的物理地址，这个时候内核会发生请页异常（或缺页异常），然后从我们空闲页框（每格4KB通常叫一个页框）看看有没有空闲的物理内存，并分配物理内存给vaddr。

Slab管理器：内核程序（kmalloc）时返回的vaddr实际上就已经和一段物理内存对应好了。首先slab会先从空闲页框分配一些物理内存放到池中，当使用kmalloc时就给内核的vaddr分配物理地址。

## Linux进程管理子系统

[主目录](#_目录)

Linux进程要素：程序与进程, 进程4要素, Linux进程状态, 进程描述结构。

Linux进程调度：调度策略, 调度时机, 调度步骤。

### 程序与进程

程序是存放在磁盘上的一系列代码和数据的可执行映像，是一个静止的实体;是一个执行中的程序,它是动态的实体。

### 进程四要素

有一段程序供其执行,这段程序不一定是某个进程所专有,可以与其他进程共用;有进程专用的内核空间堆栈;在内核中有一个task\_struct数据结构,即通常所说的“进程控制块（PCB）”,有了这个数据结构,进程才能成为内核调度的一个基本单位接受内核的调度;有独立的用户空间

用户进程（有独立用户空间）、

用户线程（只有共享用户空间）、内核线程（俩空间都没有）

如何区分进程、用户线程、内核线程之间的区别？

通过C语法来解释：有独立用户空间? 进程:有共享用户空间? 用户线程:内核线程;

Linux进程状态: 就绪态、执行态、阻塞态（三角关系）。

Linux进程状态细分(相当重要的其中6个状态)

TASK\_RUNNING(就绪、执行)

进程正在被CPU执行,或者已经准备就绪。当一个进程刚被创建时就处于TASK\_RUNNING状态。

TASK\_INTERRUPTIBLE(阻塞(可被唤醒的阻塞态))

处于等待中的进程,待等待条件为真时被唤醒,也可以被信号或中断唤醒。

TASK\_UNINTERRUPTIBLE

处于等待中的进程,待资源有效时唤醒,但是不可以由其他进程通过信号(signal)或中断唤醒。

TASK\_KILLABLE(睡眠状态)

Linux2.6.25新引入的进程睡眠状态,原理类似于TASK\_UNINTERRUPTIBLE,但是可以被致命信号(SIGKILL)唤醒。

TASK\_TRACED(调试态): 处于被调试状态的进程

TASK\_DEAD: 进程退出时（调用do\_exit）,所处的状态。

### Linux进程描述

在Linux内核代码中,线程、进程都使用PCB：task\_struct(sched.h)来表示,它包含了大量描述进程/线程的信息,其中比较重要的有：

// pid\_t pid; //进程/线程号

// long state;//进程状态

// int prio; //进程优先级

什么是进程调度？

从就绪的进程中选出最适合的一个来执行。

学习进程调度需要掌握哪些知识点？

1-调度策略 2-调度时机 3-调度步骤

### 调度策略

SCHED\_NORMAL(SCHED\_OTHER):普通的分时进程。

SCHED\_FIFO:先入先出的实时进程。

SCHED\_RR:时间片轮转的实时进程。

SCHED\_BATCH:批处理进程。

SCHED\_IDLE:只在系统空闲时才能够被调度执行的进程。

### 调度时机

什么时候发生调度？即schedule()函数什么时候被调用？

调度时机-主动式

在内核中直接调用schedule()。当进程需要等待资源而暂时停止运行时，会把自己的状态置于挂起（睡眠），并主动请求调度，让出CPU。（主动放弃CPU）范例：

// current->state = TASK\_INTERRUPTIBLE;（task\_struct）

// schedule();

调度时机-被动式

被动式调度又名:抢占式调度。分为用户态抢占（Linux2.4、Linux2.6）和内核态抢占（Linux2.6）

调度时机-用户态抢占

用户抢占发生在:

\*从系统调用返回用户空间

\*从中断处理程序返回用户空间

内核即将返回用户空间的时候,如果need\_resched标志被设置,会导致schedule()被调用,即发生用户抢占。

\*当某个进程耗尽它的时间片时，会设置need\_resched标志。

\*当一个优先级更高的进程进入可执行状态的时候,也会设置need\_resched标志。

调度时机-内核态抢占

用户态抢占缺陷(linux-2.4):进程/线程一旦运行到内核态,就可以一直执行,直到它主动放弃或时间片耗尽为止。这样会导致一些非常紧急的进程/线程将长时间得不到运行,降低整个系统的实时性。

改进方式(linux-2.6):允许系统在内核态也支持抢占,更高优先级的进程/线程可以抢占正在内核态运行的低优先级进程/线程。

内核抢占发生在:

\*中断处理程序完成,返回内核空间之前。

\*当内核代码再一次具有可抢占性的时候,如解锁及使能软中断等。

在支持内核抢占的系统中,某些特例是不允许抢占的:

\*内核正在运行中断处理。

\*内核正在进行中断上下文的bottom half(中断的底半部)处理。硬件中断返回前会执行软中断,此时仍然处于中断上下文中。

\*进程正持有spinlock自旋锁、writelock/readlock读写锁等,当持有这些锁时,不应该被抢占,否则由于抢占将可能导致其他进程长期得不到锁,而让系统处于死锁状态。

\*内核正在执行调度程序schedule。抢占的原因就是因为进行新的调度,没有理由将调度程序抢占掉再运行调度程序。

内核抢占计数

为保证Linux内核在以上情况下不会被抢占，抢占式内核使用了一个变量preempt\_count,称为内核抢占计数。这一变量被设置在进程的thread\_info结构中。每当内核要进入以上几种状态时，变量preempt\_count就加1，指示内核不允许抢占。每当内核从以上几种状态退出时,变量preempt\_count就减1,同时进行可抢占的判断与调度。

调度步骤：schedule()函数工作流程如下:

**清理当前运行中的进程;**

**选择下一个要运行的进程;**

**设置新进程的运行环境;**

**进程上下文切换。**

# 专题07-Linux内核链表

[主目录](#_目录)

## 核心理论

[主目录](#_目录)

链表简介

链表是一种常用的数据结构，它通过指针将一系列数据节点连接成一条数据链。相对于数组，链表具有更好的动态性，建立链表时无需预先知道数据总量，可以随机分配空间，可以随机分配空间，可以高效地在链表中的任意位置实时插入或删除数据。链表的开销主要是访问的顺序性和组织链的空间损失。

Linux内核链表函数清单

创建链表:INIT\_LIST\_HEAD()

插入节点:list\_add()

尾插节点:list\_add\_tail()

删除节点:list\_del()

取出节点:list\_entry()

遍历链表:list\_for\_each()

Linux内核链表为双向循环链表~

## Linux内核链表实现

struct list\_head **{**

struct list\_head **\***next**,\***prev**;**

**};**

#define LIST\_HEAD\_INIT(name) {&(name), &(name)}

#define LIST\_HEAD(name) struct list\_head name=LIST\_HEAD\_INIT(name)

static inline void INIT\_LIST\_HEAD**(**struct list\_head **\***list**)**

**{**

list**->**prev **=** list**;**

list**->**next **=** list**;**

**}**

static inline void list\_add \

**(**struct list\_head **\***new**,** struct list\_head **\***head**)**

**{**

\_\_list\_add **(**new**,** head**,** head**->**next**);**

**}**

static inline void \_\_list\_add \

**(**struct list\_head **\***new**,**struct list\_head **\***prev**,**struct list\_head **\***next**)**

**{**

next**->**prev **=** new**;**

new**->**next **=** next**;**

new**->**prev **=** prev**;**

prev**->**next **=** new**;**

**}**

static inline void list\_add\_tail\

**(**struct list\_head **\***new**,** struct list\_head **\***head**)**

**{**

\_\_list\_add **(**new**,** head**->**prev**,** head**);**

**}**

static inline void \_\_list\_del\

**(**struct list\_head **\***prev**,** struct list\_head **\***next**)**

**{**

next**->**prev **=** prev**;**

prev**->**next **=** next**;**

**}**

static inline void list\_del**(**struct list\_head **\***entry**)**

**{**

\_\_list\_del **(**entry**->**prev**,** entry**->**next**);**

entry**->**prev **=** 0xbeefdead**;**

entry**->**next **=** 0xdeadbeef**;**

**}**

#define list\_entry(ptr,type,member) container\_of(ptr,type,member)

#define \_\_list\_for\_each(pos,head) \

for(pos=(head)->next;prefetch(pos->next),pos!=(head);pos=pos->next)

offset **=** **sizeof(**type**);**

void **\***entry **=** **&**pos – offset**;**

## 测试模块-Linux内核链表使用范例

[主目录](#_目录)

#include <linux/init.h>

#include <linux/module.h>

#include <linux/list.h>

MODULE\_LICENSE**(**"GPL"**);**

**typedef** struct tag\_score

**{**

int num**;**

int english**;**

int math**;**

struct list\_head list**;**

**}** SCORE**;**

struct list\_head score\_head**;** //用来存放链表头的变量

struct list\_head **\***pos**;**

struct tag\_score stu1**,**stu2**,**stu3**;**

struct tag\_score **\***tmp**;**

static int hello\_init **(**void**)**

**{**

INIT\_LIST\_HEAD **(&**score\_head**);**//初始化链表头变量

stu1**.**num **=** 1**;**

stu1**.**english **=** 71**;**

stu1**.**math **=** 61**;**

stu2**.**num **=** 2**;**

stu2**.**english **=** 72**;**

stu2**.**math **=** 62**;**

stu3**.**num **=** 3**;**

stu3**.**english **=** 73**;**

stu3**.**math **=** 63**;**

list\_add\_tail **(&(**stu1**.**list**),** **&**score\_head**);**

list\_add\_tail **(&(**stu2**.**list**),** **&**score\_head**);**

list\_add\_tail **(&(**stu3**.**list**),** **&**score\_head**);**

printk**(**"list create and build for students' score!\n"**);**

**return** 0**;**

**}**

static void hello\_exit **(**void**)**

**{**

list\_for\_each**(**pos**,** **&**score\_head**)**

**{** //tmp(tag\_score \*), pos(list\_head \*), list(list\_head)

tmp **=** list\_entry **(**pos**,** struct tag\_score**,** list**);**

printk**(**"num: %d, english: %d, math: %d\n"**,\**

tmp**->**num**,**tmp**->**english**,**tmp**->**math**);**

**}**

list\_del **(&(**stu1**.**list**));**

list\_del **(&(**stu2**.**list**));**

list\_del **(&(**stu3**.**list**));**

**return** **;**

**}**

module\_init **(**hello\_init**);**

module\_exit **(**hello\_exit**);**

## 测试模块hello.ko对应Makefile

[主目录](#_目录)

obj-m**:=**hello.o

#hello-objs:=add.o

KDIR**:=**/home/redhat6/linux-mini2440

all**:**

make modules-C $(KDIR)M=$(PWD)ARCH=arm CROSS\_COMPILE=arm-linux-

clean**:**

rm -f \*.o \*.ko \*.mod\* \*.sym\* \*.ord\*

## 编译开发板测试模块

[主目录](#_目录)

[root@redhat6 hello]# ls

hello.c Makefile

[root@redhat6 hello]# make

make modules-C /home/redhat6/linux-mini2440M**=**/home/redhat6/hello ARCH=arm CROSS\_COMPILE=arm-linux-

make[1]**:** Entering directory `/home/redhat6/linux-mini2440'

CC [M] /home/redhat6/hello/hello.o

Building modules, stage 2.

MODPOST 1 modules

CC /home/redhat6/hello/hello.mod.o

LD [M] /home/redhat6/hello/hello.ko

make[1]**:** Leaving directory `/home/redhat6/linux-mini2440'

[root@redhat6 hello]# cp hello.ko /home/redhat6/rootfs/

cp**:** overwrite `/home/redhat6/rootfs/hello.ko'? y

[root@redhat6 hello]#

## 开发板进行测试模块测试

[主目录](#_目录)

# lsmod

# insmod hello.ko

list create and build for students' score!

# lsmod

hello 1020 0 - Live 0xbf006000

# rmmod hello

num**:** 1, english: 71, math: 61

num**:** 2, english: 72, math: 62

num**:** 3, english: 73, math: 63

rmmod**:** module 'hello' not found

#

内核链表的移植工作作为个人作业来完成！

# 专题08-Linux系统调用

[主目录](#_目录)

## 核心理论

[主目录](#_目录)

Linux系统调用：作用、工作流程分析、实现新的系统调用。

什么是系统调用？

如open(), close(), read(), write()等等。Linux被分为用户空间和内核空间，在内核空间我们实现了某个函数，这个函数要在用户空间被引用到；像read()函数的实现体不在C库也不在其他应用程序，而是来自于内核空间。

系统调用工作流程

应用程序通过系统调用从用户空间进入内核空间需要swi(软中断ARM指令)来切换到内核空间，在内核中系统调用中的每个函数都被编号，把编号赋给r7，内核代码首先从r7取出编号，利用编号作为偏移找到相应系统调用函数。

[主目录](#_目录)

## 实现一个新的系统调用

[主目录](#_目录)

### 内核空间的系统调用函数实现

我们将新的系统调用函数实现写入一个内核文件printk.c中：

extern void my\_syscall**(**void**)**

**{**

printk **(**“this is a new syscall**!**\n”**);**

**}**

通过<linux/unistd.h>我们找到<asm/unistd.h>

然后在unistd.h中添加新的系统调用函数(arch/arm/include/asm/unistd.h)

#define \_\_NR\_pk (\_\_NR\_SYSCALL\_BASE+365)

在系统调用列表最后添加:注意偏移量必须与call.S中的一致。

在相关处理器的calls.S中追加如下函数：（arch/arm/kernel/calls.S）

添加如下代码：CALL(my\_syscall)

然后重新编译内核并烧写到开发板

LD arch/arm/boot/compressed/vmlinux

OBJCOPY arch/arm/boot/zImage

Kernel**:** arch/arm/boot/zImage is ready

UIMAGE arch/arm/boot/uImage

Image Name**:** Linux-2.6.32.2-FriendlyARM

Created**:** Mon Oct 31 19:43:02 2016

Image Type**:** ARM Linux Kernel Image (uncompressed)

Data Size**:** 2287096 Bytes = 2233.49 kB = 2.18 MB

Load Address**:** 30008000

Entry Point**:** 30008000

Image arch/arm/boot/uImage is ready

[root@redhat6 linux-mini2440]# cp arch/arm/boot/uImage /tftpboot/

cp**:** overwrite `/tftpboot/uImage'? y

[root@redhat6 linux-mini2440]#

### 用户空间的系统调用函数实现

编写应用层系统调用函数

extern void pk**(**void**)** //系统调用用户空间接口函数

**{**

\_\_asm\_\_ volatile **(**

"ldr r7,=365\n"

"swi\n"

**:**

**:**

**:**"memory"

**);**

**}**

int main **(**void**)**

**{**

pk**();**//测试程序使用接口函数来调用新的系统调用

**return** 0**;**

**}**

静态交叉编译开发板测试程序

[root@redhat6 test]# arm-linux-gcc -static syscall.c -Os -o syscall

[root@redhat6 test]# ls -sh

total 576K

572K syscall 4.0K syscall.c

[root@redhat6 test]# cp syscall /home/redhat6/rootfs/

cp**:** overwrite `/home/redhat6/rootfs/syscall'? y

[root@redhat6 test]#

应用程序在开发板上测试

# ls

bin hello.ko initrd media sbin syscall usr

dev home lib mnt sqlite3 tmp var

etc init linuxrc proc sys uImage

# ./syscall

info**:** this is a new syscall

#

# 专题09-Linux驱动开发前奏

[主目录](#_目录)

## 核心理论

[主目录](#_目录)

驱动分类-常规分类法

驱动分类-常规分类法:字符设备,块设备,网络设备。

驱动分类-总线分类法:USB设备,PCI设备,平台总线设备。

字符设备:字符设备是一种按字节来访问的设备,字符驱动则负责驱动字符设备,这样的驱动通常支持open,close,read,write系统调用。例如：串口,LED,按键。

块设备:在大部分UNIX系统中,块设备定义为以块(通常为512字节)为最小传输单位的设备,块设备不能按字节处理数据。而Linux则允许块设备传送任意数目的字节。因此,块设备和字符设备的区别仅仅是驱动的与内核的接口不同。常见的块设备包括硬盘,flash,SD卡等。

网络接口设备:网络接口可以是一个硬件设备,如网卡;但也可以是一个纯粹的软件设备,比如回环接口。一个网络接口负责发送和接收数据报文。

[root@redhat6 test]# ifconfig

eth0 Link encap:Ethernet HWaddr 00:0C:29:FB:39:C1

inet addr:192.168.1.10 Bcast:192.168.1.255 Mask:255.255.255.0

lo Link encap:Local Loopback

inet addr:127.0.0.1 Mask:255.0.0.0

[root@redhat6 test]#

驱动学习方法:驱动模型,硬件操作。

驱动学习切忌:驱动学习初期(前半年)请不要过多的去阅读内核代码！

[主目录](#_目录)

## 硬件访问技术

[主目录](#_目录)

硬件访问技术:硬件访问流程,地址映射,寄存器读写。

硬件访问实质

驱动程序控制设备,主要是通过访问设备内的寄存器来达到控制的目的,因此我们讨论如何访问硬件,就成了如何访问这些寄存器了。

硬件访问流程:地址映射,读写寄存器。

地址映射:动态映射,静态映射。

## 动态映射:ioremap()

动态映射:所谓动态映射,是指在驱动程序中采用ioremap函数将物理地址映射为虚拟地址。

原型:void \* ioremap (physaddr, size);

返回值:映射后的虚拟地址。

静态映射:所谓静态映射,是指Linux系统根据用户事先指定的映射关系,在内核启动时,自动将物理地址映射为虚拟地址。

\*如何事先指定映射关系？

\*内核启动时,在什么地方完成自动映射？

## 静态映射:将映射植入内核代码

[主目录](#_目录)

将映射代码写入内核文件<cpu关联文件>中

如何静态映射？

在静态映射中,用户是通过map\_desc结构来指明物理地址与虚拟地址的映射关系。

第一是填充如下结构（以下结构）

struct map\_desc **{**

unsigned long virtual**;**//映射后的虚拟地址

unsigned long pfn**;**//物理地址所在的页帧号

unsigned long length**;**//映射长度

unsigned int type**;**//映射的设备类型

**};**

Pfn:利用\_\_phys\_to\_pfn(paddr)可以计算出物理地址所在的物理页帧号。

第二是填充好结构后还得告诉Linux内核(将填充好的结构添加到对应数组中的尾部)

(例如arch/arm/mach-s3c2440/mach-smdk2440.c)

(例如arch/arm/mach-s5p64x0/cpu.c)

static struct map\_desc smdk2440\_iodesc**[]** \_\_initdata **=** **{**

/\* ISA IO Space map (memory space selected by A24) \*/

**{**

**.**virtual **=** **(**u32**)**S3C24XX\_VA\_ISA\_WORD**,**

**.**pfn **=** \_\_phys\_to\_pfn**(**S3C2410\_CS2**),**

**.**length **=** 0x10000**,**

**.**type **=** MT\_DEVICE**,**

**},** **{**

**.**virtual **=** **(**u32**)**S3C24XX\_VA\_ISA\_WORD **+** 0x10000**,**

**.**pfn **=** \_\_phys\_to\_pfn**(**S3C2410\_CS2 **+** **(**1**<<**24**)),**

**.**length **=** SZ\_4M**,**

**.**type **=** MT\_DEVICE**,**

**},** **{**

**.**virtual **=** **(**u32**)**S3C24XX\_VA\_ISA\_BYTE**,**

**.**pfn **=** \_\_phys\_to\_pfn**(**S3C2410\_CS2**),**

**.**length **=** 0x10000**,**

**.**type **=** MT\_DEVICE**,**

**},** **{**

**.**virtual **=** **(**u32**)**S3C24XX\_VA\_ISA\_BYTE **+** 0x10000**,**

**.**pfn **=** \_\_phys\_to\_pfn**(**S3C2410\_CS2 **+** **(**1**<<**24**)),**

**.**length **=** SZ\_4M**,**

**.**type **=** MT\_DEVICE**,**

**}**

**};**

第三就交给内核在启动时去初始化地址映射了

static void \_\_init mini2440\_map\_io**(**void**)**

**{**

s3c24xx\_init\_io**(**mini2440\_iodesc**,** ARRAY\_SIZE**(**mini2440\_iodesc**));**

s3c24xx\_init\_clocks**(**12000000**);**

s3c24xx\_init\_uarts**(**mini2440\_uartcfgs**,** ARRAY\_SIZE**(**mini2440\_uartcfgs**));**

**}**

static void \_\_init smdk2440\_map\_io**(**void**)**

**{**

s3c24xx\_init\_io**(**smdk2440\_iodesc**,** ARRAY\_SIZE**(**smdk2440\_iodesc**));**

s3c24xx\_init\_clocks**(**16934400**);**

s3c24xx\_init\_uarts**(**smdk2440\_uartcfgs**,** ARRAY\_SIZE**(**smdk2440\_uartcfgs**));**

**}**

## 寄存器读写函数集

在完成地址映射（通常为静态映射）后，就可以读写寄存器了，Linux内核提供了一系列函数，来读写寄存器。

unsigned ioread8**(**void **\***addr**);**

unsigned ioread16**(**void **\***addr**);**

unsigned ioread32**(**void **\***addr**);**

unsigned readb**(**address**);**

unsigned readw**(**address**);**

unsigned readl**(**address**);**

unsigned iowrite8**(**u8 value**,** void **\***addr**);**

unsigned iowrite16**(**u16 value**,** void **\***addr**);**

unsigned iowrite32**(**u32 value**,** void **\***addr**);**

unsigned writeb**(**unsigned value**,** address**);**

unsigned writew**(**unsigned value**,** address**);**

unsigned writel**(**unsigned value**,** address**);**

# 专题10-字符设备驱动模型

## 核心理论

[主目录](#_目录)

使用字符设备驱动程序

编译/安装字符设备驱动程序,创建设备文件,访问设备。

[主目录](#_目录)

编译/安装字符设备驱动程序

在Linux系统中，驱动程序通常采用内核模块的程序结构来进行编码。因此，编译/安装一个驱动程序，其实质就是编译/安装一个内核模块。

动手做实验-Makefile的编写

obj-m**:=**memdev.o

KDIR**:=**/home/redhat6/linux-mini2440

PWD**:=**$(shell pwd)

all**:**

make modules-C $(KDIR) M**=**$(PWD) ARCH=arm CROSS\_COMPILE=arm-linux-

.PHONY**:**clean

clean**:**

rm -rf \*.ko \*.o \*.mod\* \*.sym\* \*.order\*

将memdev.ko拷贝到开发板安装驱动。

[主目录](#_目录)

字符设备文件

通过字符设备文件，应用程序可以使用相应的字符设备驱动程序来控制字符设备。创建字符设备文件的方法一般有两种: 创建字符设备文件

使用mknod /dev/文件名 c 主设备号 次设备号

使用函数在驱动程序中创建（后续课程介绍）

自动创建设备文件

2.4内核:

devfs\_register (dir,\*name,flags,major,minor,mode,\*ops,\*info);

dir:目录名，为空时默认在/dev/目录下创建；

name:文件名；

flags:创建标志；

major:主设备号；

minor:次设备号；

mode:创建模式；

ops:操作函数集；

info:通常为空。

2.6内核:

从2.6.13开始,devfs不复存在,udev成为devfs的替代。

相比devfs, udev(mdev)存在于应用层。利用udev(mdev)来实现设备文件的自动创建很简单，只要在驱动初始化的代码里调用class\_create()为该设备创建一个classs,再为每个设备调用device\_create()创建对应的设备。

范例:自动创建设备文件：

static struct class\_device **\***myclass\_dev**;**

static struct class **\***myclass**;**

major **=** register\_chrdev **(**0**,** “mydev”**,** **&**ops**);**

myclass **=** class\_create **(**THIS\_MODULE**,** “mydev”**);**

myclass\_dev **=** device\_create **(**myclass**,** **NULL,** MKDEV**(**major**,** 0**),** **NULL,**”xy”**);**

范例:自动创建删除文件：

unregister\_chrdev **(**major**,** "mydev"**);**

class\_device\_unregister **(**myclass\_dev**);**

device\_destroy **(&**myclass**,** major**);**

class\_destroy **(**myclass**);**

当驱动被加载时，udev(mdev)就会自动在/dev/目录下创建my\_device设备文件。

### app.c

#include<stdio.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<fcntl.h>

#include<unistd.h>

#include<sys/mman.h>

int main**()**

**{**

int fd**;**

char **\***start **=** **NULL;**

//char buf[100];

char **\***buf**;**

fd **=** open**(**"/dev/memdev0"**,**O\_RDWR**);**

buf **=** **(**char **\*)**malloc**(**100**);**

memset**(**buf**,** 0**,** 100**);**

start **=** mmap**(NULL,** 100**,** PROT\_READ**|**PROT\_WRITE**,**MAP\_SHARED**,**fd**,**0**);**

strcpy**(**buf**,**start**);**

sleep **(**1**);**

printf**(**"buf 1 = %s\n"**,**buf**);**

strcpy**(**start**,**"Buf Is Not Null!"**);**

memset**(**buf**,** 0**,** 100**);**

strcpy**(**buf**,**start**);**

sleep **(**1**);**

printf**(**"buf 2 = %s\n"**,**buf**);**

munmap**(**start**,**100**);** /\*解除映射\*/

free**(**buf**);**

close**(**fd**);**

**return** 0**;**

**}**

### memdev.c

#include <linux/module.h>

#include <linux/fs.h>

#include <linux/init.h>

#include <linux/cdev.h>

#include <asm/uaccess.h>

int dev1\_registers**[**5**];**

int dev2\_registers**[**5**];**

struct cdev cdev**;**

dev\_t devno**;**

/\*文件打开函数\*/

int mem\_open**(**struct inode **\***inode**,** struct file **\***filp**)**

**{**

/\*获取次设备号\*/

int num **=** MINOR**(**inode**->**i\_rdev**);**

**if** **(**num**==**0**)**

filp**->**private\_data **=** dev1\_registers**;**

**else** **if(**num **==** 1**)**

filp**->**private\_data **=** dev2\_registers**;**

**else**

**return** **-**ENODEV**;** //无效的次设备号

**return** 0**;**

**}**

/\*文件释放函数\*/

int mem\_release**(**struct inode **\***inode**,** struct file **\***filp**)**

**{**

**return** 0**;**

**}**

/\*读函数\*/

static ssize\_t mem\_read**(**struct file **\***filp**,** char \_\_user **\***buf**,** size\_t size**,** loff\_t **\***ppos**)**

**{**

unsigned long p **=** **\***ppos**;**

unsigned int count **=** size**;**

int ret **=** 0**;**

int **\***register\_addr **=** filp**->**private\_data**;** /\*获取设备的寄存器基地址\*/

/\*判断读位置是否有效\*/

**if** **(**p **>=** 5**\*sizeof(**int**))**

**return** 0**;**

**if** **(**count **>** 5**\*sizeof(**int**)** **-** p**)**

count **=** 5**\*sizeof(**int**)** **-** p**;**

/\*读数据到用户空间\*/

**if** **(**copy\_to\_user**(**buf**,** register\_addr**+**p**,** count**))**

**{**

ret **=** **-**EFAULT**;**

**}**

**else**

**{**

**\***ppos **+=** count**;**

ret **=** count**;**

**}**

**return** ret**;**

**}**

/\*写函数\*/

static ssize\_t mem\_write**(**struct file **\***filp**,** const char \_\_user **\***buf**,** size\_t size**,** loff\_t **\***ppos**)**

**{**

unsigned long p **=** **\***ppos**;**

unsigned int count **=** size**;**

int ret **=** 0**;**

int **\***register\_addr **=** filp**->**private\_data**;** /\*获取设备的寄存器地址\*/

/\*分析和获取有效的写长度\*/

**if** **(**p **>=** 5**\*sizeof(**int**))**

**return** 0**;**

**if** **(**count **>** 5**\*sizeof(**int**)** **-** p**)**

count **=** 5**\*sizeof(**int**)** **-** p**;**

/\*从用户空间写入数据\*/

**if** **(**copy\_from\_user**(**register\_addr **+** p**,** buf**,** count**))**

ret **=** **-**EFAULT**;**

**else**

**{**

**\***ppos **+=** count**;**

ret **=** count**;**

**}**

**return** ret**;**

**}**

/\* seek文件定位函数 \*/

static loff\_t mem\_llseek**(**struct file **\***filp**,** loff\_t offset**,** int whence**)**

**{**

loff\_t newpos**;**

**switch(**whence**)** **{**

**case** SEEK\_SET**:**

newpos **=** offset**;**

**break;**

**case** SEEK\_CUR**:**

newpos **=** filp**->**f\_pos **+** offset**;**

**break;**

**case** SEEK\_END**:**

newpos **=** 5**\*sizeof(**int**)-**1 **+** offset**;**

**break;**

**default:**

**return** **-**EINVAL**;**

**}**

**if** **((**newpos**<**0**)** **||** **(**newpos**>**5**\*sizeof(**int**)))**

**return** **-**EINVAL**;**

filp**->**f\_pos **=** newpos**;**

**return** newpos**;**

**}**

/\*文件操作结构体\*/

static const struct file\_operations mem\_fops **=**

**{**

**.**llseek **=** mem\_llseek**,**

**.**read **=** mem\_read**,**

**.**write **=** mem\_write**,**

**.**open **=** mem\_open**,**

**.**release **=** mem\_release**,**

**};**

/\*设备驱动模块加载函数\*/

static int memdev\_init**(**void**)**

**{**

/\*初始化cdev结构\*/

cdev\_init**(&**cdev**,** **&**mem\_fops**);**

/\* 注册字符设备 \*/

alloc\_chrdev\_region**(&**devno**,** 0**,** 2**,** "memdev"**);**//动态获取设备号

cdev\_add**(&**cdev**,** devno**,** 2**);**

// 在/sys/class目录下生成memdev\_class目录

myclass **=** class\_create**(**THIS\_MODULE**,**"memdev\_class"**);**

device\_create**(**myclass**,NULL,**MKDEV**(**mem\_major**,**mem\_minor**+**i**),NULL,**"memdev%d"**,**i**);**

**}**

/\*模块卸载函数\*/

static void memdev\_exit**(**void**)**

**{**

cdev\_del**(&**cdev**);** /\*注销设备\*/

unregister\_chrdev\_region**(**devno**,** 2**);** /\*释放设备号\*/

class\_destroy**(&**myclass**);**

device\_destroy **(&**myclass**,** dev\_t**);**

**}**

module\_init**(**memdev\_init**);**

module\_exit**(**memdev\_exit**);**

MODULE\_LICENSE**(**"GPL"**);** /\* 模块许可证 \*/

MODULE\_AUTHOR**(**"seafly"**);** /\* 作者声明 \*/

MODULE\_DESCRIPTION**(**"memdev module"**);** /\* 模块描述 \*/

MODULE\_VERSION**(**"V1.0"**);** /\* 模块版本声明 \*/

通过应用程序读写字符设备文件来测试字符设备文件驱动是否编写成功

应用程序在开发板上运行提示:not found字样的解决方法

\*交叉编译采用静态编译(1)

\*通过arm-linux-readelf –d app.elf来查询应用程序依赖的动态链接库。并根据提示来添加相应库。(2)

查询应用程序依赖的动态链接库(前提是动态交叉编译的elf文件)

[root@redhat6 test]# arm-linux-readelf -d memio.elf

Dynamic section at offset 0x6f0 contains 24 entries:

Tag Type Name/Value

0x00000001 (NEEDED) Shared library: [libc.so.6]

0x0000000c (INIT) 0x8374

0x0000000d (FINI) 0x862c

查询应用程序的机器属性

[root@redhat6 test]# arm-linux-readelf -a memio.elf

ELF Header:

Magic: 7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00

Class: ELF32

Data: 2's complement, little endian

Version: 1 (current)

OS/ABI: UNIX - System V

ABI Version: 0

Type: EXEC (Executable file)

Machine: ARM

Version: 0x1

Entry point address: 0x841c

[主目录](#_目录)

## 测试程序实验室

[主目录](#_目录)

### memdev.c

#include <linux/module.h>

#include <linux/fs.h>

#include <linux/init.h>

#include <linux/cdev.h>

#include <asm/uaccess.h>

int dev1\_registers**[**5**];**

int dev2\_registers**[**5**];**

struct cdev cdev**;**

dev\_t devno**;**

/\*文件打开函数\*/

int mem\_open**(**struct inode **\***inode**,** struct file **\***filp**)**

**{**

/\*获取次设备号\*/

int num **=** MINOR**(**inode**->**i\_rdev**);**

**if** **(**num**==**0**)**

filp**->**private\_data **=** dev1\_registers**;**

**else** **if(**num **==** 1**)**

filp**->**private\_data **=** dev2\_registers**;**

**else**

**return** **-**ENODEV**;** //无效的次设备号

**return** 0**;**

**}**

/\*文件释放函数\*/

int mem\_release**(**struct inode **\***inode**,** struct file **\***filp**)**

**{**

**return** 0**;**

**}**

/\*读函数\*/

static ssize\_t mem\_read**(**struct file **\***filp**,** char \_\_user **\***buf**,** size\_t size**,** loff\_t **\***ppos**)**

**{**

unsigned long p **=** **\***ppos**;**

unsigned int count **=** size**;**

int ret **=** 0**;**

int **\***register\_addr **=** filp**->**private\_data**;** /\*获取设备的寄存器基地址\*/

/\*判断读位置是否有效\*/

**if** **(**p **>=** 5**\*sizeof(**int**))**

**return** 0**;**

**if** **(**count **>** 5**\*sizeof(**int**)** **-** p**)**

count **=** 5**\*sizeof(**int**)** **-** p**;**

/\*读数据到用户空间\*/

**if** **(**copy\_to\_user**(**buf**,** register\_addr**+**p**,** count**))**

**{**

ret **=** **-**EFAULT**;**

**}**

**else**

**{**

**\***ppos **+=** count**;**

ret **=** count**;**

**}**

**return** ret**;**

**}**

/\*写函数\*/

static ssize\_t mem\_write**(**struct file **\***filp**,** const char \_\_user **\***buf**,** size\_t size**,** loff\_t **\***ppos**)**

**{**

unsigned long p **=** **\***ppos**;**

unsigned int count **=** size**;**

int ret **=** 0**;**

int **\***register\_addr **=** filp**->**private\_data**;** /\*获取设备的寄存器地址\*/

/\*分析和获取有效的写长度\*/

**if** **(**p **>=** 5**\*sizeof(**int**))**

**return** 0**;**

**if** **(**count **>** 5**\*sizeof(**int**)** **-** p**)**

count **=** 5**\*sizeof(**int**)** **-** p**;**

/\*从用户空间写入数据\*/

**if** **(**copy\_from\_user**(**register\_addr **+** p**,** buf**,** count**))**

ret **=** **-**EFAULT**;**

**else**

**{**

**\***ppos **+=** count**;**

ret **=** count**;**

**}**

**return** ret**;**

**}**

/\* seek文件定位函数 \*/

static loff\_t mem\_llseek**(**struct file **\***filp**,** loff\_t offset**,** int whence**)**

**{**

loff\_t newpos**;**

**switch(**whence**)** **{**

**case** SEEK\_SET**:**

newpos **=** offset**;**

**break;**

**case** SEEK\_CUR**:**

newpos **=** filp**->**f\_pos **+** offset**;**

**break;**

**case** SEEK\_END**:**

newpos **=** 5**\*sizeof(**int**)-**1 **+** offset**;**

**break;**

**default:**

**return** **-**EINVAL**;**

**}**

**if** **((**newpos**<**0**)** **||** **(**newpos**>**5**\*sizeof(**int**)))**

**return** **-**EINVAL**;**

filp**->**f\_pos **=** newpos**;**

**return** newpos**;**

**}**

/\*文件操作结构体\*/

static const struct file\_operations mem\_fops **=**

**{**

**.**llseek **=** mem\_llseek**,**

**.**read **=** mem\_read**,**

**.**write **=** mem\_write**,**

**.**open **=** mem\_open**,**

**.**release **=** mem\_release**,**

**};**

/\*设备驱动模块加载函数\*/

static int memdev\_init**(**void**)**

**{**

/\*初始化cdev结构\*/

cdev\_init**(&**cdev**,** **&**mem\_fops**);**

/\* 注册字符设备 \*/

alloc\_chrdev\_region**(&**devno**,** 0**,** 2**,** "memdev"**);**

cdev\_add**(&**cdev**,** devno**,** 2**);**

**}**

/\*模块卸载函数\*/

static void memdev\_exit**(**void**)**

**{**

cdev\_del**(&**cdev**);** /\*注销设备\*/

unregister\_chrdev\_region**(**devno**,** 2**);** /\*释放设备号\*/

**}**

MODULE\_LICENSE**(**"GPL"**);**

module\_init**(**memdev\_init**);**

module\_exit**(**memdev\_exit**);**

### memio.c

#include <string.h>

#include <sys/types.h>

#include <fcntl.h>

#include <unistd.h>

int main **(**void**)**

**{**

int fd **=** 0**;**

char buf**[**100**]** **=** "abcdef123456"**;**

fd **=** open**(**"/dev/memdev"**,** O\_WRONLY**);**

write **(**fd**,** buf**,** strlen**(**buf**));** printf **(**"write:buf:%s\n"**,** buf**);**

close **(**fd**);**

//注意设备文件操作（读写）要分开进行(重新打开文件)，否则测试实验不会成功。

fd **=** open **(**"/dev/memdev"**,** O\_RDONLY**);**

memset **(**buf**,** 0**,** **sizeof(**buf**));** printf **(**"memset:buf:%s\n"**,** buf**);**

read**(**fd**,** buf**,** **sizeof(**buf**));** printf **(**"read:buf: %s\n"**,** buf**);**

close **(**fd**);**

**return** 0**;**

**}**

### Makefile

bj-m**:=**memdev.o

KDIR**:=**/home/redhat6/linux-mini2440

all**:**

make modules -C $(KDIR) M**=**$(PWD) ARCH=arm CROSS\_COMPILE=arm-linux-

arm-linux-gcc -static -Os memio.c -o memio.elf

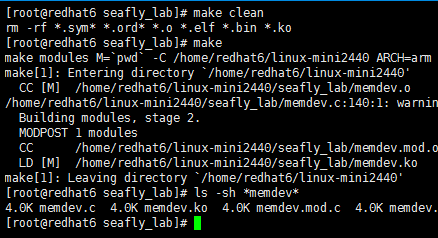
.PHONY**:**clean

clean**:**

rm -rf \*.ko \*.o \*.mod\* \*.sym\* \*.order\* \*.elf

[主目录](#_目录)

编译并将范例驱动和测试程序拷贝至开发板测试



### 手动创建设备文件

#insmod memdev.ko

#cat /proc/devices #获取主设备号

#mknod /dev/memdev major minor

应用程序---文件名---设备文件(桥梁)---主设备号---次设备号---设备驱动程序

## 字符驱动编程模型

[主目录](#_目录)

[主目录](#_目录)

字符驱动编程模型:设备描述结构cdev, 字符设备驱动模型, 范例驱动分析。

[主目录](#_目录)

谢老师经验分享

在Linux系统中，设备的类型非常繁多，如:字符设备, 块设备, 网络接口设备, USB设备, I2C设备, PCI设备, 平台设备, 混杂设备......,而设备类型不同，也意味着其对应的驱动程序模型不同，这样就导致了我们需要去掌握众多的驱动程序模型。那么能不能从这些众多的驱动模型中提炼出一些具有共性的规则，则是我们能否学好Linux驱动的关键。

[主目录](#_目录)

设备驱动模型:驱动初始化, 实现设备操作, 驱动注册。

[主目录](#_目录)

驱动初始化:分配设备描述结构, 初始化设备描述结构, 注册设备描述结构, 硬件初始化。

[主目录](#_目录)

### 设备描述结构

设备描述结构-字符设备描述结构(<kernel>(linux-mini2440)/include/cdev.h)

在任何一种驱动模型中，设备都会用内核中的一种结构来描述。我们的字符设备在内核中使用struct cdev来描述:描述结构的用法可以参考内核其他模块的使用。

### 设备号管理

查看Linux设备号

[root@redhat6 test]# ls -l /dev/ | more

total 0

crw-rw----. 1 root video 10, 175 Nov 1 2016 agpgart //,分隔的分别为主次设备号

Linux内核中使用dev\_t来定义设备号,dev\_t实质为32位的uint型,其中高12位为主设备号,低20位为次设备号。

那如果知道主设备号,次设备号,怎么组合成dev\_t类型？

答:dev\_t dev = MKDEV(主设备号, 次设备号);

如何从dev\_t中分解出主设备号？如何从dev\_t中分解出次设备号？

答:主设备号 = MAJOR(dev\_t dev); 次设备号 = MINOR(dev\_t dev);

分配字符设备号

静态申请:开发者自己选择一个数字作为主设备号，然后通过函数register\_chrdev\_region向内核申请使用。缺点:如果申请使用的设备号已经被内核中的其他驱动使用了,则申请失败。

动态分配:使用alloc\_chrdev\_region由内核自动分配一个可用的主设备号。优点：因为内核知道哪些号已经被使用了，所以不会导致分配到已经被使用的号。

注销字符设备号

无论使用何种方法分配设备号，都应该在驱动退出时，使用unregister\_chrdev\_region函数释放这些设备号。否则最终会造成内存泄漏。

## 文件操作函数集

const struct file\_operations \*ops; //文件操作集

struct file\_operations cdev\_fops = {

.llseek = NULL,

.read = cdev\_read,

.write = cdev\_write,

.ioctl = cdev\_ioctl,

.open = cdev\_open,

.release = dev\_release

};

## 字符设备驱动设计流程

[主目录](#_目录)

## 创建字符设备

[主目录](#_目录)

静态分配:struct cdev mdev;

动态分配:struct cdev \*pdev = cdev\_alloc();

## 初始化字符设备

[主目录](#_目录)

使用函数:cdev\_init (struct cdev \*cdev, const struct file\_operations \*fops);

## 注册字符设备

[主目录](#_目录)

使用函数:cdev\_add (struct cdev \*pdev, dev\_t dev, unsigned count);

## 硬件初始化

[主目录](#_目录)

### 动态映射:ioremap()

动态映射:所谓动态映射,是指在驱动程序中采用ioremap函数将物理地址映射为虚拟地址。

原型:void \* ioremap (physaddr, size);

返回值:映射后的虚拟地址。

静态映射:所谓静态映射,是指Linux系统根据用户事先指定的映射关系,在内核启动时,自动将物理地址映射为虚拟地址。

\*如何事先指定映射关系？

\*内核启动时,在什么地方完成自动映射？

### 静态映射:将映射植入内核代码

[主目录](#_目录)

将映射代码写入内核文件<cpu关联文件>中

如何静态映射？

在静态映射中,用户是通过map\_desc结构来指明物理地址与虚拟地址的映射关系。

第一是填充结构（以下结构）

struct map\_desc {

unsigned long virtual;//映射后的虚拟地址

unsigned long pfn;//物理地址所在的页帧号

unsigned long length;//映射长度

unsigned int type;//映射的设备类型

};

Pfn:利用\_\_phys\_to\_pfn(paddr)可以计算出物理地址所在的物理页帧号。

第二是填充好结构后还得告诉Linux内核(将填充好的结构添加到对应数组中的尾部)

(例如arch/arm/mach-s3c2440/mach-smdk2440.c)

(例如arch/arm/mach-s5p64x0/cpu.c)

static struct map\_desc smdk2440\_iodesc[] \_\_initdata = {

/\* ISA IO Space map (memory space selected by A24) \*/

{

.virtual = (u32)S3C24XX\_VA\_ISA\_WORD,

.pfn = \_\_phys\_to\_pfn(S3C2410\_CS2),

.length = 0x10000,

.type = MT\_DEVICE,

}, {

.virtual = (u32)S3C24XX\_VA\_ISA\_WORD + 0x10000,

.pfn = \_\_phys\_to\_pfn(S3C2410\_CS2 + (1<<24)),

.length = SZ\_4M,

.type = MT\_DEVICE,

}, {

.virtual = (u32)S3C24XX\_VA\_ISA\_BYTE,

.pfn = \_\_phys\_to\_pfn(S3C2410\_CS2),

.length = 0x10000,

.type = MT\_DEVICE,

}, {

.virtual = (u32)S3C24XX\_VA\_ISA\_BYTE + 0x10000,

.pfn = \_\_phys\_to\_pfn(S3C2410\_CS2 + (1<<24)),

.length = SZ\_4M,

.type = MT\_DEVICE,

}

};

第三就交给内核在启动时去初始化地址映射了

static void \_\_init mini2440\_map\_io(void)

{

s3c24xx\_init\_io(mini2440\_iodesc, ARRAY\_SIZE(mini2440\_iodesc));

s3c24xx\_init\_clocks(12000000);

s3c24xx\_init\_uarts(mini2440\_uartcfgs, ARRAY\_SIZE(mini2440\_uartcfgs));

}

static void \_\_init smdk2440\_map\_io(void)

{

s3c24xx\_init\_io(smdk2440\_iodesc, ARRAY\_SIZE(smdk2440\_iodesc));

s3c24xx\_init\_clocks(16934400);

s3c24xx\_init\_uarts(smdk2440\_uartcfgs, ARRAY\_SIZE(smdk2440\_uartcfgs));

}

### 寄存器读写函数集

在完成地址映射（通常为静态映射）后，就可以读写寄存器了，Linux内核提供了一系列函数，来读写寄存器。

unsigned ioread8(void \*addr);

unsigned ioread16(void \*addr);

unsigned ioread32(void \*addr);

unsigned readb(address);

unsigned readw(address);

unsigned readl(address);

unsigned iowrite8(u8 value, void \*addr);

unsigned iowrite16(u16 value, void \*addr);

unsigned iowrite32(u32 value, void \*addr);

unsigned writeb(unsigned value, address);

unsigned writew(unsigned value, address);

unsigned writel(unsigned value, address);

## 实现操作集

[主目录](#_目录)

打开设备:int (\*open)(struct inode \*, struct file \*);

关闭设备:int (\*release)(struct inode \*, struct file \*);

重定位读写指针:loff\_t (\*llseek)(struct file \*, loff\_t, int);

控制设备: long (\*ioctl)(struct file \*fp, uint32 cmd, uint32 arg);

还有其他函数操作集...

read设备

Read设备方法通常完成2件事情:

\*从设备中读取数据（属于硬件访问类操作）

\*将读取到的数据返回给应用程序。

(红色为内核空间参数，蓝色为用户空间参数)

函数:ssize\_t (\*read)(struct file \*filp, char \_\_usr \*buf, size\_t count, loff\_t \*offp);

参数分析:

Filp:与字符设备文件关联的file结构指针，由内核创建。

Buff:从设备读取到的数据需要保存到的位置。由read系统调用提供该参数。

Count:请求传输的数据量，由read系统调用提供该参数。

Offp:文件的读写位置，由内核从file结构中取出后，传递进来。

### 用户空间和内核空间数据传递

其中buff参数是来源于用户空间的指针，这类指针都不能被内核代码直接引用，必须使用专门的函数:

int copy\_from\_user(void \*to, const void \_\_user \*from, int size);

int copy\_to\_user(void \_\_user \*to, const void \*from, int size);

write设备

Read设备方法通常完成2件事情:

\*从应用程序提供的地址中取出数据。

\*将读取到的数据写入设备。（属于硬件访问类操作）

函数:ssize\_t (\*write)(struct file \*filp, char \_\_usr \*buf, size\_t count, loff\_t \*offp);

参数分析:

Filp:与字符设备文件关联的file结构指针，由内核创建。

Buff:从设备读取到的数据需要保存到的位置。由read系统调用提供该参数。

Count:请求传输的数据量，由read系统调用提供该参数。

Offp:文件的读写位置，由内核从file结构中取出后，传递进来。

## 注销字符设备

[主目录](#_目录)

当我们从内核中卸载驱动程序的时候，需要使用cdev\_del函数来完成字符设备的注销。

## 字符设备驱动memdev.ko源码分析

（此驱动核心包含文件操作（文件读写指针操作））

[主目录](#_目录)

驱动程序模块主程序

#include <linux/module.h>

#include <linux/fs.h>

#include <linux/init.h>

#include <linux/cdev.h>

#include <asm/uaccess.h>//头文件的包含根据编译出错提示添加

MODULE\_LICENSE("GPL");

/\*字符设备驱动模块加载函数\*/

static int memdev\_init(void)

{

[cdev\_init](#_字符设备描述cdev的初始化)(&[cdev](#_Cdev.sourcecode), &[mem\_fops](#_mem_fops.sourcecode)); /\*初始化字符设备\*/

[alloc\_chrdev\_region](#_分配一个(主)设备号)(&[devno](#_devno.sourcecode), 0, 2, "memdev");//分配字符设备号

[cdev\_add](#_字符设备描述cdev的注册)(&cdev, devno, 2); /\* 注册字符设备 \*/

}

/\*字符设备驱动模块卸载函数\*/

static void memdev\_exit(void)

{

[cdev\_del](#_字符设备驱动注销)(&cdev); /\*注销字符设备\*/

[unregister\_chrdev\_region](#_注销设备号)(devno, 2); /\*释放字符设备号\*/

}

module\_init(memdev\_init);

module\_exit(memdev\_exit);

### [devno.sourcecode](#_字符设备驱动memdev.ko源码分析)

dev\_t devno;

### [Cdev.sourcecode](#_字符设备驱动memdev.ko源码分析)

[分配字符设备描述](#_字符设备描述cdev的分配)dev

struct cdev cdev;

### [mem\_fops.sourcecode](#_字符设备驱动memdev.ko源码分析)

/\*文件操作结构体\*/

static const struct file\_operations mem\_fops =

{

.llseek = [mem\_llseek](#_mem_llseek.sourcecode),

.read = [mem\_read](#_mem_read.sourcecode),

.write = [mem\_write](#_mem_write.sourcecode),

.open = [mem\_open](#_mem_open.sourcecode),

.release = [mem\_release](#_mem_release.sourcecode),

};

### [mem\_release.sourcecode](#_mem_fops.sourcecode)

/\*文件释放函数\*/

int mem\_release(struct inode \*inode, struct file \*filp)

{

return 0;

}

### [mem\_open.sourcecode](#_mem_fops.sourcecode)

/\*文件打开函数\*/

int mem\_open(struct inode \*inode, struct file \*filp)

{

/\*获取次设备号\*/

int num = MINOR(inode->i\_rdev);

//为file结构数据域赋值地址

if (num==0)

filp->private\_data = [dev1\_registers](#_registers.sourcecode);//分配sizeof(int)\*5内存空间

else if(num == 1)

filp->private\_data = [dev2\_registers](#_registers.sourcecode);//分配sizeof(int)\*5内存空间

else

return -ENODEV; //无效的次设备号

return 0;

}

### [registers.sourcecode](#_mem_open.sourcecode)

定义物理内存的虚拟地址

int dev1\_registers[5]; //sizeof(int) \* 5或者sizeof(char)\*4\*5个字节

int dev2\_registers[5]; //sizeof(int) \* 5或者sizeof(char)\*4\*5个字节

### [mem\_write.sourcecode](#_mem_fops.sourcecode)

神总结:文件IO指针操作, 比作0~100数字操作, 0为指向文件0位置,100为指向文件尾部,文件指针就在0~100之间根据用户操作自由移动。

static ssize\_t mem\_write(struct file \*filp, const char \_\_user \*buf, size\_t size, loff\_t \*ppos)

{

unsigned long p = \*ppos;

unsigned int count = size;

int ret = 0;

/\*获取设备的寄存器起始地址\*/

int \*register\_addr = filp->private\_data;

/\*分析和获取有效的写长度\*/

if (\*ppos>= 5\*sizeof(int)) //如果 (指针位置 >= 总内存)

return 0;

if (size> 5\*sizeof(int) - \*ppos) //如果 (传入size > (100-\*ppos))

count = 5\*sizeof(int) - \*ppos; //count = 100 - \*ppos

/\*从用户空间写入数据\*/

if (copy\_from\_user(register\_addr + p, buf, count))

ret = -EFAULT;

else

{

\*ppos += count;//\*ppos = \*ppos+(100-\*ppos)

ret = count;

}

return ret;

}

### [mem\_read.sourcecode](#_mem_fops.sourcecode)

神总结:文件IO指针操作, 比作0~100数字操作, 0为指向文件0位置,100为指向文件尾部,文件指针就在0~100之间根据用户操作自由移动。

/\*读函数\*/

static ssize\_t mem\_read (struct file \*filp, char \_\_user \*buf, size\_t size, loff\_t \*ppos)

{ //struct file在项目中先cscope\_find g file, 然后在quick中搜索字串struct file {

unsigned long p = \*ppos;

unsigned int count = size;

int ret = 0;

/\*获取设备的寄存器起始地址\*/

int \*register\_addr = filp->private\_data;

/\*判断读位置是否有效\*/

if (\*ppos>= 5\*sizeof(int))//如果 (指针位置>=总内存)

return 0;

if (size> 5\*sizeof(int) - \*ppos)//如果 (传入size >指针位置)

count = 5\*sizeof(int) - \*ppos;//count = 总内存 – 指针位置

/\*读数据到用户空间\*/

if ([copy\_to\_user](#_用户空间和内核空间数据传递)(buf, register\_addr+(\*ppos), count)!=0)

{copy\_to\_user(用户buf, 起始指针位置, 偏移量)

ret = -EFAULT;

}

else //copy\_to\_user()成功

{

\*ppos += count;//\*ppos = \*ppos+count;

ret = count;

}

return ret;

}

### [mem\_llseek.sourcecode](#_mem_fops.sourcecode)

/\* seek文件定位函数 \*/

static loff\_t mem\_llseek(struct file \*filp, loff\_t offset, int whence)

{

loff\_t newpos;

switch(whence) {

case SEEK\_SET:

newpos = offset;

break;

case SEEK\_CUR:

newpos = filp->f\_pos + offset;

break;

case SEEK\_END:

newpos = 5\*sizeof(int)-1 + offset;

break;

default:

return -EINVAL;

}

if ((newpos<0) || (newpos>5\*sizeof(int)))

return -EINVAL;

filp->f\_pos = newpos;

return newpos;

}

## 专题11-LED驱动程序设计

[主目录](#_目录)

## LED的Linux驱动程序分析

（此驱动核心包含硬件操作（硬件初始化））

[主目录](file:///C:UsersAdministratorDesktop课程笔记内核驱动-上学期.docx" \l "_目录)

#include <linux/module.h> //模块头文件

#include <linux/init.h> //模块头文件

#include <linux/cdev.h> //字符设备相关头文件

#include <linux/fs.h> //文件系统头文件

#include <linux/io.h> //文件IO头文件

#include <mach/gpio-bank-k.h> //GPIO头文件

#include "led.h"

static int led\_init()

{

[cdev\_init](file:///C:UsersAdministratorDesktop课程笔记内核驱动-上学期.docx" \l "_字符设备描述cdev的初始化)(&[cdev](file:///C:UsersAdministratorDesktop课程笔记内核驱动-上学期.docx" \l "_cdev.sourcecode_1),&[led\_fops](file:///C:UsersAdministratorDesktop课程笔记内核驱动-上学期.docx" \l "_led_fops.sourcecode)); //分配并初始化字符设备

[alloc\_chrdev\_region](file:///C:UsersAdministratorDesktop课程笔记内核驱动-上学期.docx" \l "_分配一个(主)设备号)(&[devno](file:///C:UsersAdministratorDesktop课程笔记内核驱动-上学期.docx" \l "_devno.sourcecode_1), 0 , 1 , "myled"); //分配字符设备设备号

[cdev\_add](file:///C:UsersAdministratorDesktop课程笔记内核驱动-上学期.docx" \l "_字符设备描述cdev的注册)(&[cdev](file:///C:UsersAdministratorDesktop课程笔记内核驱动-上学期.docx" \l "_cdev.sourcecode_1), [devno](file:///C:UsersAdministratorDesktop课程笔记内核驱动-上学期.docx" \l "_devno.sourcecode_1), 1); //注册该字符设备

return 0;

}

static void led\_exit()

{

[cdev\_del](file:///C:UsersAdministratorDesktop课程笔记内核驱动-上学期.docx" \l "_字符设备驱动注销)(&[cdev](file:///C:UsersAdministratorDesktop课程笔记内核驱动-上学期.docx" \l "_cdev.sourcecode_1)); //注销该字符设备

[unregister\_chrdev\_region](file:///C:UsersAdministratorDesktop课程笔记内核驱动-上学期.docx" \l "_注销设备号)([devno](file:///C:UsersAdministratorDesktop课程笔记内核驱动-上学期.docx" \l "_devno.sourcecode_1),1); //注销释放设备号

}

module\_init(led\_init);

module\_exit(led\_exit);

### [cdev.sourcecode](file:///C:\\Users\\Administrator\\Desktop\\课程笔记\\内核驱动-上学期.docx" \l "_LED的Linux驱动程序分析)

[分配字符设备描述结构](file:///C:UsersAdministratorDesktop课程笔记内核驱动-上学期.docx" \l "_字符设备描述cdev的分配)

struct cdev cdev;

### [devno.sourcecode](file:///C:\\Users\\Administrator\\Desktop\\课程笔记\\内核驱动-上学期.docx" \l "_LED的Linux驱动程序分析)

dev\_t devno;

### [led\_fops.sourcecode](file:///C:\\Users\\Administrator\\Desktop\\课程笔记\\内核驱动-上学期.docx" \l "_LED的Linux驱动程序分析)

static struct file\_operations led\_fops =

{

.open = [led\_open](file:///C:UsersAdministratorDesktop课程笔记内核驱动-上学期.docx" \l "_led_open.sourcecode),

.unlocked\_ioctl = [led\_ioctl](file:///C:UsersAdministratorDesktop课程笔记内核驱动-上学期.docx" \l "_led_ioctl.sourcecode),

};

### [led\_config.sourcecode](file:///C:\\Users\\Administrator\\Desktop\\课程笔记\\内核驱动-上学期.docx" \l "_led_open.sourcecode)

unsigned int \*led\_config;

### [led\_data.sourcecode](file:///C:\\Users\\Administrator\\Desktop\\课程笔记\\内核驱动-上学期.docx" \l "_led_open.sourcecode)

unsigned int \*led\_data;

### [led\_open.sourcecode](file:///C:\\Users\\Administrator\\Desktop\\课程笔记\\内核驱动-上学期.docx" \l "_led_fops.sourcecode)

int led\_open(struct inode \*node, struct file \*filp)

{

//动态地址映射(物理地址映射到虚拟地址上,其中led\_config为映射的虚拟地址)

[led\_config](file:///C:UsersAdministratorDesktop课程笔记内核驱动-上学期.docx" \l "_led_config.sourcecode) = [ioremap](file:///C:UsersAdministratorDesktop课程笔记内核驱动-上学期.docx" \l "_动态映射:ioremap())([LEDCON](file:///C:UsersAdministratorDesktop课程笔记内核驱动-上学期.docx" \l "_Physaddr.sourcecode),4);

[writel](file:///C:UsersAdministratorDesktop课程笔记内核驱动-上学期.docx" \l "_寄存器读写函数集)(0x11110000,led\_config);

[led\_data](file:///C:UsersAdministratorDesktop课程笔记内核驱动-上学期.docx" \l "_led_data.sourcecode) = [ioremap](file:///C:UsersAdministratorDesktop课程笔记内核驱动-上学期.docx" \l "_动态映射:ioremap())([LEDDAT](file:///C:UsersAdministratorDesktop课程笔记内核驱动-上学期.docx" \l "_Physaddr.sourcecode),4);

return 0;

}

### [Physaddr.sourcecode](file:///C:\\Users\\Administrator\\Desktop\\课程笔记\\内核驱动-上学期.docx" \l "_led_open.sourcecode)

#define LEDCON 0x7f008800

#define LEDDAT 0x7f008808

### [led\_ioctl.sourcecode](file:///C:\\Users\\Administrator\\Desktop\\课程笔记\\内核驱动-上学期.docx" \l "_led_fops.sourcecode)

long led\_ioctl(struct file \*filp, unsigned int cmd, unsigned long arg)

{

switch (cmd)

{

case LED\_ON:

writel(0x00,led\_data);

return 0;

case LED\_OFF:

writel(0xff,led\_data);

return 0;

default:

return -EINVAL;

}

}

## 驱动访问大揭秘

[主目录](#_目录)

[主目录](#_目录)

驱动访问模型

应用程序(read,write)------内核??????------驱动程序(xx\_read, xx\_write)

[主目录](#_目录)

我们从read入手开始研究内核是如何为应用程序和驱动建立关联的（其他系统调用都是类似的原理）。

编译memrd.c

下面我们来分析上面的memdev.c的应用程序memrd.c反汇编代码：

[root@redhat6 test]# ls

Makefile memdev.c memio.c memrd.c memwr.c

[root@redhat6 test]# make

make –C $(KDIR) modules M=$(PWD) ARCH=arm CROSS\_COMPILE=arm-linux-

make[1]: Entering directory $(KDIR)

......

LD [M] /home/redhat6/test/memdev.ko

make[1]: Leaving directory $(KDIR)

arm-linux-gcc –g –Os –static memrd.c –o memrd.elf

......

[root@redhat6 test]# make dump

Arm-linux-objdump –D –S memrd.elf > memrd.dump

......

[主目录](#_目录)

[主目录](#_目录)

打开启用调试选项的反汇编文件memrd.dump

从read()找到sys\_read()函数

然后我们的目光集中在read系统调用函数反汇编出来的汇编代码上，由反汇编代码可知read()函数的三个参数分别被保存在r0,r1,r2寄存器中（当参数小于4个时候参数直接保存在寄存器中），然后看到read最后一行反汇编代码调用了\_\_libc\_read函数，所以我们继续跟踪到\_\_libc\_read函数的反汇编代码处，把目光集中在其中两行反汇编代码上（“mov r7, #3\n”“svc 0x00000000\n”），其中svc是一个很特殊的指令（系统调用指令），当我们在应用程序当中使用这个指令了之后，我们的PC指针会从用户空间进入到内核空间而且入口是固定的（也就是首先它会通过固定的入口进入到内核）（1-入口），之后内核会去取r7里面的值（num）（2-取编号），然后根据这个num去查表并找到对应的sys\_read()。此过程可从内核代码(arch/arm/kernel/entry-common.S)中找到，先找到vector\_swi关键字，vector\_swi的入口即是进入内核空间的入口，然后通过关键字system call number，即可找到get system call number，所以能看出（2-取编号），再通过关键字sys\_call\_table找到一个表所在的文件，通过关键字ENTRY(sys\_call\_table)找到关键字#include “call.S”，然后我们去打开这个call.S汇编文件，然后看到里面有很多函数列成的表，最后我们根据编号来找到对应的函数调用。

通过sys\_read()找到我们实现的mem\_read()函数

因为sys\_read()的实现体(首先我们是无法直接根据名字跟踪找到sys\_read的实现体的)放在read\_write.c中（fs/read\_write.c），然后我们通过关键字SYSCALL\_DEFINE3找到sys\_read的实现代码。然后在实现代码中找到vfs\_read()函数并跟踪到这个函数的实现处，然后找到最重要的一行代码ret = file->f\_op->read(file, buf, count, pos);其中的f\_op中的read函数就调用了我们驱动中实现的mem\_read()函数了。

[主目录](#_目录)

# 第四季-内核驱动奥秘多(内核驱动)-下学期

# 专题01-课程规划与学习方法

# 专题01-按键驱动程序设计

[主目录](#_目录)

## 混杂设备核心理论

[主目录](#_目录)

混杂设备驱动模型：混杂设备描述, 混杂驱动注册, 范例驱动分析。

[主目录](#_目录)

混杂设备概念

在Linux系统中，存在一类字符设备，他们拥有相同的主设备号（10），但次设备号不同，我们称这类设备为混杂设备（miscdevice）。所有的混杂设备形成一个链表，对设备访问时内核根据次设备号查找到相应的混杂设备。

Linux里面驱动很多，但是它们都有一个共性的东西，比如说Linux这么多设备中，我首先要找出Linux内核给我们提供了什么样的结构来描述这个设备的，其次我们要知道怎么来注册这个设备的，最后是我们怎么来注销这个设备的。

[主目录](#_目录)

### 混杂设备描述

混杂设备的描述结构<linux/miscdevice.h>

include**/**linux**/**miscdevice**.**h

struct miscdevice **{**

int minor**;** //Device minor,混杂设备主设备号为10

const char **\***name**;** //device name

const struct file\_operations **\***fops**;** //device file operations

struct list\_head list**;**

struct device **\***parent**;**

struct device **\***this\_device**;**

const char **\***nodename**;**

mode\_t mode**;**

**};**

混杂设备的注册与注销

extern int misc\_register **(**struct miscdevice **\*** misc**);**

extern int misc\_deregister **(**struct miscdevice **\*** misc**);**

学习新的设备驱动的学习方法

不管是字符设备、混杂设备等等这些设备，首先我们要先找出在Linux内核中使用什么结构来描述这些设备的，我们要使用结构中哪些成员，哪些成员由内核完成。

然后寻找怎么注册这个设备的。最后就是怎么注销这个设备的。

[主目录](#_目录)

## 混杂设备驱动模型

### key.c

#include <linux/init.h>

#include <linux/module.h>

#include <linux/miscdevice.h>

MODULE\_LICENSE**(**"GPL"**);**

struct file\_operations mdev\_fops**;**

struct miscdevice mdev **=** **{**

**.**minor **=** 200**,**

**.**name **=** "mdevice"**,**

**.**fops **=** **&**mdev\_fops

**};**

static int key\_init **(**void**)**

**{**

misc\_register**(&**mdev**);**

**return** 0**;**

**}**

static void key\_init **(**void**)**

**{**

misc\_deregister **(&**mdev**);**

**return** **;**

**}**

module\_init **(**key\_init**);**

module\_exit **(**key\_exit**);**

### Makefile

obj-m**:=**key.o

KDIR**:=**/home/redhat6/linux-mini2440

all**:**

make modules M=$(PWD) $(KDIR) ARCH=arm CROSS\_COMPILE=arm-linux-

.PHONY**:**clean

clean**:**

rm -rf \*.o \*.elf \*.ko \*.order\* \*.sym\* \*.mod\* \*.bin

## Linux中断处理深层解析

为什么会在按键驱动过程中去插入一个中断处理的知识点呢？

在我们前面学习裸机按键驱动的时候，我们是以中断的方式让按键工作的，所以说我们要让按键能够按照中断方式工作，所以说我们必须要让处理器中断这种机制能够工作起来；而Linux下面的按键驱动程序，我们一样要了解在我们的Linux系统中是怎么来处理中断的，整个Linux处理中断的流程是怎么样的，中断处理程序该怎么去设计等等，那么这是我们在学习Linux按键驱动之前必须要掌握的知识点。

回顾裸机按键中断处理流程

在start.S中，对中断会有一个统一的入口(irq)，当我们硬件发生中断的时候，不管是哪种类型的中断，它都会跳转到中断向量表这个地方来，然后从irq入口，也就是统一的入口，统一入口的任务就是环境的保存、执行处理函数、环境的恢复；跳转到中断函数处理中，该函数首先会读取产生中断的设备，也就是读取中断源，然后根据中断源编号调用相应的中断处理程序（switch-case），而这些中断处理程序是事先实现并注册进来的程序，最后清除中断。

Linux系统中断处理流程

在 entry-armv.S中有个标号是\_\_irq\_svc，那么这个标处实际上就是Linux中断的统一入口，进入到这个统一入口后，它同样去做环境保存啊这些工作，首先是要拿到产生中断源的编号，然后根据这个中断号去做接下来的事情，利用获得的中断号找出irq\_desc结构，比如说irq\_desc[0]就代表0号中断，irq\_desc[1]就代表1号中断，以此类推，硬件产生的每个中断编号都和irq\_desc相对应，在irq\_desc结构当中就会有action这个选项，而在action中就是存放的用户事先填写进去的中断处理函数，这些函数在当前irq\_desc下串联成一个任务链表，当该中断号发生时，这个irq\_desc里的每个中断处理函数都会去执行一遍，那没有产生中断的函数也要执行吗？当然要，只不过处理函数里面首先要添加中断检查，检查该函数所属硬件是否产生中断，如果没有就直接退出，这样就不会影响到其他中断处理函数的执行了。

为什么我要给同学们讲这个中断处理过程呢？

第一，让同学们对中断有个更深刻的认识，其次我们通过我的中断处理过程的分析，那么要得出一个结论，这个结论就是我们的驱动程序到底该做什么？那么从上面的整个流程来看，我们的驱动程序其实总的有两件事要做，首先第一点是要去实现中断处理程序，第二点，我们自己实现好的这个中断处理程序要能够被Linux系统能够调用到，也就说当相应中断产生的时候，要能调用到你这个驱动程序的中断处理程序，那么还要把这个中断处理程序注册到Linux系统当中来，说得更具体点就是注册到irq\_desc这个结构当中来，最后当卸载该模块的时候要注销中断。

Linux中注册中断

int request\_irq(); //0表示成功，或者返回一个错误码。

函数参数解析：

flags:和中断相关的选项，比如说IRQF\_DISABLED（SA\_INTERRUPT），如果设置该位，表示这是一个快速中断处理程序，否则就是一个慢速中断处理程序，快速中断主要区别在于快速中断保证中断处理的原子性（不被打断）；IRQF\_SHARED（SA\_SHIRQ），该位表明该中断号是多个设备共享的，比如irq\_desc[0]表示对应0号中断结构，但是同学们有没有发现它的里面怎么挂了多个中断处理函数？这个就是Linux中比较有特色的地方共享中断，我们多个设备可以共享一个中断号，比如说我的串口可以用0号中断，网卡可以用0号中断，然后你们看到没有，实际上irq\_desc[0]里面的处理函数形成一个链表，内核会把该链表中的每一个处理函数都去调用一遍，那么都去调用一遍会不会有问题呢？肯定会有问题，你的网卡根本就没有产生中断，那它的处理函数也会被调用一遍，所以说我们在要求编写我们的中断处理程序的时候首先要检查你的这个设备是不是产生了中断，比如说调用到你这个网卡驱动程序，没有关系，你调用就调用呗，调用进来之后我的网卡中断处理程序自己一检测发现我的网卡根本就没产生中断，那我就立刻退出咯，就不会有任何问题了。

devname:中断的对应的设备的名字。

void \*dev\_id:共享中断时使用。

中断处理程序

我们的驱动程序必须要去实现这个中断处理程序，那么中断处理程序一般是由C代码来写的，它和一般的C程序有什么区别呢？中断处理程序是运行在中断上下文当中的，那么它的行为要受到一定的限制，比如说你不能使用引起阻塞的函数，为什么呢？如果说你一旦阻塞了，那你想一想，特别是对于我们的快速中断，一旦在这儿阻塞了，停下来了，那么我整个系统的中断都没有办法进一步的处理，这个时候对我的系统的性能是一个很大的影响；第二个我们不能使用引起调度的函数。好，这是我们的两个原则。

我们来看看中断处理程序的一般流程，首先我们中断处理程序一进来之后，大家养成一个良好的习惯，就是检查你这个设备是否产生了这个中断，这是第一步；第二步，清除中断标志，因为你不去把这个产生中断的标志清除掉，那么下一步的中断是没有办法进一步产生的；然后我们就去完成相应的硬件操作，比如说我们要从设备拷贝数据啊，或者要往设备里面写数据啊，那么你就根据你这设备的逻辑完成剩下的操作；最后，当我们不再使用中断的时候，比如说我的驱动程序要注销的时候，那么我们一般就要使用free\_irq()这个函数把我们注册好的这个中断处理函数给注销掉，其中传入参数irq（中断号），那么这里又有一个问题，如果我只提供中断号，对于共享中断我怎么办？比如说我提供了一个中断号0（irq\_desc[0]），但是0下面挂了多个中断处理函数，那你到底是注销哪个呢？所以说我们在注册的时候，特别对于共享中断，我们通常会让它的dev\_id不一样，那么它就会根据dev\_id注销掉我们想要注销的中断处理函数了。

**注册中断处理函数:request\_irq()** //请求中断函数

**int request\_irq(unsigned int irq,**

**irq\_handler\_t handler,**

**unsigned long flags, const char \*devname, void \*dev\_id)**

其中传入参数:

irq:对应中断号

handler:中断处理函数

Flags: IRQF\_DISABLED: 如果设置该位,表示是一个快速中断处理程序.

IRQF\_SHARED: 该位表明该中断号是多个设备共享的。

devname:设备名

dev\_id:共享中断时使用。

中断处理程序的特别之处是在**中断上下文**中运行的，它的行为受到如下限制：

**不能使用可能引起阻塞的函数**，**不能使用可能引起调度的函数**。

**handler:中断处理函数**

\*\*检查设备是否产生了中断

\*\*清除中断产生标志

\*\*相应的硬件操作

**注销中断:free\_irq()**

当设备不再需要使用中断时（通常在驱动卸载时），应当把它们注销掉：

**void free\_irq (uint irq, void \*dev\_id);**

**Linux中断处理函数流程框架搭建**

## Linux中断处理框架代码搭建

#include <linux/init.h>

#include <linux/module.h>

#include <linux/miscdevice.h>

#include <linux/irqreturn.h>

MODULE\_LICENSE**(**"GPL"**);**

struct file\_operations mdev\_fops**;**

struct miscdevice mdev **=** **{**

**.**minor **=** 200**,**

**.**name **=** "mdevice"**,**

**.**fops **=** **&**mdev\_fops

**};**

static int dev\_init **(**void**)**

**{**

misc\_register**(&**mdev**);**

request\_irq **(**irq**,** dev\_init**,** "mdevice"**,** 0**,** IRQ\_PROBE\_SHARED**);**

**return** 0**;**

**}**

static void dev\_exit **(**void**)**

**{**

misc\_deregister **(&**mdev**);**

free\_irq **(**irq**,** 0**);**

**return** **;**

**}**

module\_init **(**dev\_init**);**

module\_exit **(**dev\_exit**);**

## Linux中断按键驱动硬件操作实现

分析裸机阶段的按键驱动程序

Linux驱动中，硬件初始化的地方：**open函数**或**module\_init函数**中

我们能不能直接把裸机阶段的按键驱动移植过来呢？答案是能的。

首先我们的按键是与GPIO相连的，因此我们的GPIO功能你首先要去做选择，这是其一，其二，我们的按键是以中断的方式来工作的，所以说我们要从这两个点来复习一下。

今天我们就写一个按键的驱动就行了，其它类似。

在我们Linux驱动程序当中，我们应该在哪个地方去完成这个硬件的初始化呢？实际上在我们Linux驱动开发当中，我们一般会选择两个地方来做硬件的初始化，一个地方呢是我们的open函数，另外一个地方呢是在我们的模块初始化当中来完成硬件的初始化。选哪一个其实没有一个严格的规定，具体情况看公司怎么规定，都可以。

接下来呢我们就首先去Linux驱动代码中去实现按键硬件初始化的代码，我们这里选择在模块初始化中初始化硬件，如下代码：

static int dev\_init **(**void**)**

**{**

misc\_register**(&**mdev**);**

request\_irq **(**irq**,** dev\_init**,** "mdevice"**,** 0**,** IRQ\_PROBE\_SHARED**);**

//Device hardware initialization

dev\_hw\_init **();**

**return** 0**;**

**}**

接下来呢我们就要来实现这个函数，这个函数的实现呢，当然要参考裸机代码，这里我们按键的初始化工作其实只有一项，就是对GPIO引脚的控制寄存器做相应初始化，我们把GPIO物理地址控制寄存器地址定义过来，而在我们Linux内核中不能直接去使用物理地址，我们必须要先把它转化成虚拟地址，通过ioremap()函数转换，所以接下来我们就来操作这个虚拟地址，首先我们要读取这个寄存器里面的原值，为什么呢？因为我们不能破坏里面原有的值，所以我们先读出来，在去设置该值得相应位，最后才写进这个寄存器。

下面是我们实现硬件初始化函数范例：如下代码：

static void dev\_hw\_init **(**void**)**

**{**

//From your board:K1:EINT8:GPG0:

#define GPGCON 0x56000060

unsigned int **\***gpio\_config**;**

unsigned int data**;**

gpio\_config **=** **(**int **\*)**ioremap**(**GPGCON**,** 4**);**

data **=** readw**(**gpio\_config**);**

data **&=** **~(**0x3**);**

data **|=** 0x2**;**

writew **(**data**,** gpio\_config**);**

**return** **;**

**}**

那么这样呢，我们的按键硬件方面初始化的函数就实现完了，我们接下来来看硬件的第二部分，我们第一部分就对按键引脚初始化，第二部分呢就是我们要对按键的中断处理做工作，下面我们就是来修改这个注册中断函数，我们前面在大框架过程中我们是在这个注册这个环节留下一些尾巴的，就是注册标志以及中断号，因为当时我们还不清楚怎么设置这些参数，之前只是为了搭建框架而填充的，我们的按键有两个状态，一个是按下，一个是弹起，所以我们选择的标志是IRQF\_TRIGGER\_FALLING标志，除了这些标志我们可以通过vim的Ctrl-n或sourceinsight中的自动匹配来查询还有其它标志列表。如下：

static int dev\_init **(**void**)**

**{**

misc\_register**(&**mdev**);**

request\_irq **(**irq**,** dev\_init**,** IRQ\_TRIGGER\_FALLING**,** "mdevice"**,** 0**);**

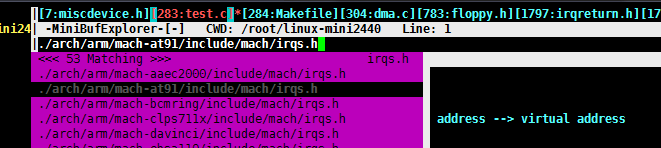
//Device hardware initialization

dev\_hw\_init **();**

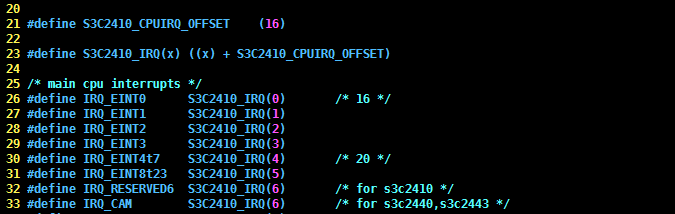
**return** 0**;**

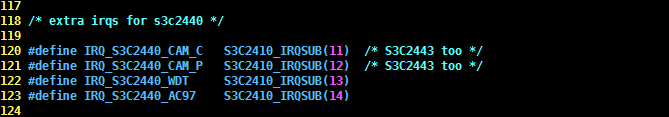
**}**

那么我们来完成下一个尾巴，我们下一个尾巴是个大尾巴，就是我们的中断号，我们裸机中获取中断号可以通过寄存器相应位来获取，然而在Linux驱动开发中我们怎么获取这个中断号呢？也就是我们的中断类型怎么和我们的中断号对应呢？我们先教大家来怎么查这个东西，我们通过LookupFile搜索irqs.h这个头文件，或者通过soueceinsight来搜索相应板子的这个irq.h然后找出相应板子的头文件即可，如下图：



下面我们打开相应开发板的头文件，会看到一大串定义的中断宏，如下图所示：



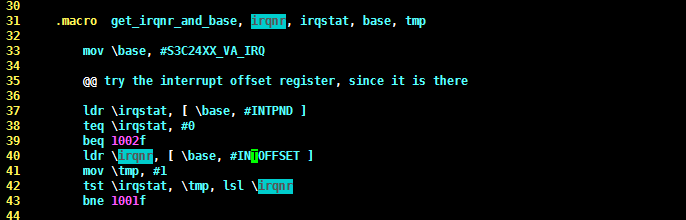


但是如果你想把中断号这个知识给搞透了，那么你还得继续听谢老师讲讲，讲讲这个中断号，它的值是等于多少的？它和我们芯片手册上的中断类型到底是一个什么样的关系？我们先来看它的值，IRQF\_EINIT0这个宏展开之后，那么得到IRQF\_EINIT0的值是16，那么我们这里通过芯片手册查看到EINT0的中断号是0，那么为什么这里的值是16呢？前面的16个中断号又是谁的呢？在2440上面是16，在6410是32，板子不同有所差异。那么这个OFFSET的中断号是留给谁的呢？答案是留给软中断的，也就是留给软件模拟的中断的。于是我们的这个OFFSET来源就搞清楚了。那么有的同学又问了，你这些中断号是用来干嘛的？根据我们前面裸机所学的知识，我们知道中断号是用来寻找中断表的。

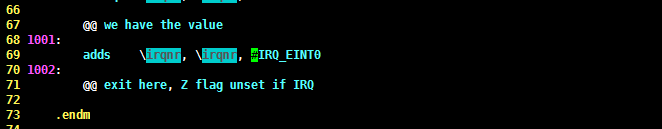
在我们Linux系统当中，当我们的硬件产生中断的时候，处理器会把这个中断号拿到，然后根据这个中断号，利用这个中断号去找到这个描述结构（irq\_desc），然后从这个结构当中的处理函数链表中取到相应中断处理函数。那么我们现在得到的中断号是16+实际的序号，有人会问，那么硬件产生的中断号也是16+这个序号吗？下面我们来打开一个文件来分析，打开entry-macro.S，我们中断号获取其实是由get\_irqnr\_and\_base这个宏来做的，如下图：



我们先找到这个irqnr(irq\_number)，实际上这里面保存的就是硬件的中断号，我们高亮这个变量，看看这个中断号是怎么拿到的，高亮这个变量，然后我们可以看到哪些地方操作了irqnr这个变量，所以我们通过高亮之后的代码里很快发现，irqnr里面的值实际上是从INTOFFSET这个寄存器得到的，如下图：



当我们的硬件产生中断之后，那么这个中断编号首先是从我们的INTOFFSET这个寄存器里面去读，打开我们的芯片手册，我们来看看这个INTOFFSET寄存器。所以我们这个irqnr读出来这个寄存器里面中断的序号，但是我们这里仅仅是需要也不行啊，这个序号和我们最后要用到的中断号还要差16呀，那么你和这个结构找不到呀，于是我们来看看接下来是怎么做的，如下图：



通过上图这个步骤之后，我们后面Linux要用的irq\_number于是就产生出来了。

所以我们这里来个总结：在Linux系统中，我们硬件实际上产生的是一个序号或者叫做偏移号，这是硬件实际产生出来的，产生出来之后在寄存器里保存起来的，但是这个号呢和我们后面irq\_desc使用方法有些出入，所以说我们去给这个序号+16, 然后就形成了为Linux所用的中断号，那么这就是Linux系统中的中断号，所以我们得出，Linux系统中的中断号是等于序号+基数，为什么要有这个基数，前面我们已经说过了，这个基数是留给软中断使用的。

我们接下来实现按键中断处理函数：

函数里我们首先要做的就是检测是否发生了按键中断，其次清除已经发生的按键中断，最后进行中断处理，这里我们就已简单打印来代替；其中的中断检测部分我们可以不做，因为我们的按键在该中断号下只有按键这么一个中断处理函数挂上，所以这一步我们可以暂时不用检查，还有我们后面的中断清除操作是针对外围硬件内部的中断清除，而我们按键的中断清除的寄存器GPIO属于处理器级别的，那么处理器界别的寄存器我们不需要去清除，处理器会自动去清除它，比如说我们用的网卡芯片，我们网卡芯片内部本身就有中断相关的状态寄存器，所以我们对于网卡这种外围设备来说是必须要做中断清除这个步骤的。

### 测试的key.c范例

#include <linux/init.h>

#include <linux/module.h>

#include <linux/miscdevice.h>

#include <linux/io.h> //ioremap, etc.

#include <linux/fs.h>

#include <linux/interrup.h> //request\_irq, etc.

#include <linux/irqreturn.h>

MODULE\_LICENSE**(**"GPL"**);**

struct file\_operations key\_fops**;**

struct miscdevice mdev **=** **{**

**.**minor **=** 200**,**

**.**name **=** "mdevice"**,**

**.**fops **=** **&**key\_fops

**};**

static void key\_hw\_init **(**void**)**

**{**

//From your board:K1:EINT8:GPG0:

#define GPGCON 0x56000060

unsigned int **\***gpio\_config**;**

unsigned int data**;**

gpio\_config **=** **(**int **\*)**ioremap**(**GPGCON**,** 4**);**

data **=** readw**(**gpio\_config**);**

data **&=** **~(**0x3**);**

data **|=** 0x2**;**

writew **(**data**,** gpio\_config**);**

**return** **;**

**}**

irqreturn\_t key\_int**(**int irq**,** void **\***dev\_id**)**

**{**

//01.check whether irq or not for the device

**;**

//02.clear irq issued

**;**

//03.print value of key

printk**(**"key down\n"**);**

**return** 0**;**

**}**

static int key\_init **(**void**)**

**{**

misc\_register**(&**mdev**);**

request\_irq **(**IRQ\_EINT0**,** key\_int**,** IRQF\_TRIGGER\_FALLING**,** "mdevice"**,** 0**);**

//Device hardware initialization

dev\_hw\_init **();**

**return** 0**;**

**}**

static void key\_exit **(**void**)**

**{**

misc\_deregister **(&**mdev**);**

free\_irq **(**irq**,** 0**);**

**return** **;**

**}**

module\_init **(**key\_init**);**

module\_exit **(**key\_exit**);**

### 相应的Makefile

obj-m**:=**key.o

KDIR**:=**/home/redhat6/linux-mini2440

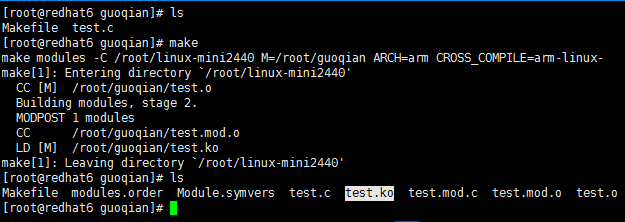
all**:**

make modules M=$(PWD) $(KDIR) ARCH=arm CROSS\_COMPILE=arm-linux-

.PHONY**:**clean

clean**:**

rm -rf \*.o \*.elf \*.ko \*.order\* \*.sym\* \*.mod\* \*.bin



## Linux中断分层技术

Linux是如何来处理中断嵌套的？

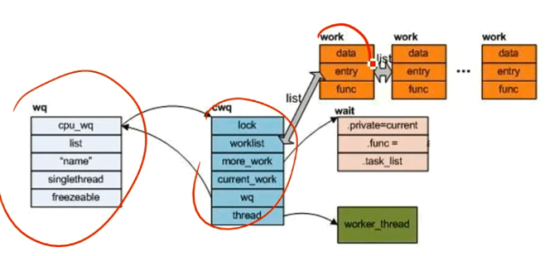
什么叫做中断嵌套呢？就是当一种类型中断发生的时候，又产生了其它的中断，那么其它的中断可以是同类型的，也可以是不同类型的中断，这就是所谓的中断嵌套，不同的OS对中断嵌套有不同的处理办法；我们这里以Linux为主，分为两种比较大的类型，一种就是慢速中断，什么是慢速中断呢？就是指的是在进行中断处理的时候，这个中断的总开关是不关闭的，也就说允许其它类型的中断产生，这就是慢速中断；假如说一个串口中断产生并开始进行处理，处理总时间为7秒钟，当执行到第3秒的时候又来了一个其它类型的中断，比如说网卡中断，那么这个时候Linux系统怎么处理呢？那么Linux呢它会暂停上面的串口处理，转而去执行网卡中断处理程序，加入网卡中断处理程序执行了3秒之后执行完毕，那么它又会回到串口接着执行剩下的中断处理程序，直到把它执行完毕；上面是第一种情况，那么第二种情况是同样是处理串口在执行前3秒时候又来了个串口中断，这个时候Linux会不会去执行新来的串口中断程序呢？答案是不会的，因为这两种类型中断是同类型的，Linux在处理同类型中断的时候，即使你再产生中断，它也不会暂停下来去处理新中断，它就会把它忽略掉，这就意味第二个中断的数据就丢失掉了；当然上面情况都是针对慢速中断情况。下面我们看看快速中断处理情况，快速中断就是在执行中断处理时候，这个中断的总开关是关闭的，也就是说是不允许其它中断打断的，比如说串口执行到第3秒时，这时候来了个网卡，你的网卡根本就产生不了中断，因为中断总开关是被关闭的，所以说你这个网卡中断是被忽略掉的，也就是意味数据丢掉，同类型新串口中断一样被忽略掉；综上得知，快速中断或慢速中断中的同类型中断都会出现这种“中断丢失”的现象，那么中断丢失是不是我们希望看到的情况呢？当然不是，因为我们的新中断需要处理的数据得不到处理，比如说我们的网卡收到一个数据包，那么它产生不了中断，那么这个数据包自然而然也没有办法送到系统当中去了，这显然不是我们希望看到的情况，那么怎么来解决这个问题呢？这里从而我们引出了“中断分层”的技术。

中断分层技术

假如说一个中断处理总共需要10秒钟的时间，现在当我们这个中断处理程序运行到第7秒钟的时候，这时候产生新类型中断，但是这个新类型中断呢很不幸丢失掉了，那么人们就在想怎么解决这个问题，于是想了一个很简单的办法就是我们能不能把这个中断处理的时间尽量缩短，比如说我们让它在第6秒时候就完成中断处理，这样就极大减少了中断丢失的可能，因为你后面4秒钟都不属于这个中断处理程序了，那你再来其它类型的中断也能得到处理了，所以说就减少了中断丢失可能；其实中断处理程序里面会做两种类型的工作，一种类型的工作是和硬件相关的，比如说网卡接收数据这个中断处理程序，那你要从相应寄存器中去读取收到的这个数据，那么这部分工作肯定就是和硬件密切相关的，第二部分工作就是和硬件无关的工作，比如说做一些检测或数据的相应处理啊，这些工作和硬件本身就没有太大关系，那么它就不一定要放在中断处理程序当中来做了；于是人们想了个办法，就是把和硬件密切相关的工作放到中断处理程序当中完成，把和硬件无关的操作就丢到系统给的队列里让内核空闲时候去完成，于是出现了“上半部”和“下半部”的概念，上半部就是和硬件密切相关的，这部分就必须在中断处理程序当中做，下半部呢就是硬件无关的处理，就不放在中断处理中执行了；那么问题来了，下半部要抛出去，怎么实现抛出去的工作呢？

中断分层方式

中断分层方式分为“软中断”、“tasklet”、“工作队列”，我们先在用得最广泛的是工作队列，你可以把工作队列的场景想象成有一个核CPU窗口，窗口外面排满打饭的学生，这些学生每个学生就是每个工作，我们的内核为每一个窗口都安排了一个为学生打饭的内核线程，这个内核线程就会选择相对比较空闲时候就会为排队的每一个学生打饭，每个打完饭的学生就会从队列中离开，也就是说这个专门打饭的内核线程就会在内核相对空闲的时候自动去检查学生队列有没有要打饭的学生，如果有，就帮他打饭并让他离开队列。



我们内核中是如何来描述这个工作队列的呢？我们内核用struct workqueue\_struct来描述一个工作队列的，用struct work\_struct来描述一个工作的，这项工作其实就是去执行一个函数，所以说工作这个结构中func这个成员是比较重要的。所以我们的步骤就出来了，首先是创建工作队列（create\_workqueue），其次创建工作（INIT\_WORK），最后是提交工作（queue\_work），提交工作就是把这个工作挂到工作队列中去。

但是大多数情况下，驱动并不需要自己去建立工作队列，只需要创建工作，然后将工作提交到内核已经定义好的工作队列keventd\_wq，通过schedule\_work函数提交工作到默认队列即可。

还有个问题，为什么硬件中断的时候不直接把硬件相关和软件相关的操作都放到队列中呢？这是不可能的，因为硬件的工作依赖于中断机制，没有中断做前提，硬件无法工作。

### queue.c

#include <linux/init.h>

#include <linux/module.h>

struct workqueue\_struct **\***my\_wq**;**

struct work\_struct **\***work1**;**

struct work\_struct **\***work2**;**

MODULE\_LICENSE**(**"GPL"**);**

void work1\_func**(**struct work\_struct **\***work**)**

**{**

printk**(**"this is work1->\n"**);**

**}**

void work2\_func**(**struct work\_struct **\***work**)**

**{**

printk**(**"this is work2->\n"**);**

**}**

int init\_que**(**void**)**

**{**

//1. 创建工作队列

my\_wq **=** create\_workqueue**(**"my\_que"**);**

//2. 创建工作1

work1 **=** kmalloc**(sizeof(**struct work\_struct**),**GFP\_KERNEL**);**

INIT\_WORK**(**work1**,** work1\_func**);**

//3. 挂载（提交）工作

queue\_work**(**my\_wq**,**work1**);**

//2. 创建工作2

work2 **=** kmalloc**(sizeof(**struct work\_struct**),**GFP\_KERNEL**);**

INIT\_WORK**(**work2**,** work2\_func**);**

//3. 挂载（提交）工作

queue\_work**(**my\_wq**,**work2**);**

**return** 0**;**

**}**

void clean\_que**()**

**{**

kfree **(**work1**);**

kfree **(**work2**);**

**}**

module\_init**(**init\_que**);**

module\_exit**(**clean\_que**);**

### Makefile

obj-m**:=**key.o

KDIR**:=**/home/redhat6/linux-mini2440

all**:**

make modules M=$(PWD) $(KDIR) ARCH=arm CROSS\_COMPILE=arm-linux-

.PHONY**:**clean

clean**:**

rm -rf \*.o \*.elf \*.ko \*.order\* \*.sym\* \*.mod\* \*.bin

## Linux中断分层技术修改key.c

**提交下半部（硬件无关工作）**

#include <linux/init.h>

#include <linux/module.h>

#include <linux/miscdevice.h>

#include <linux/io.h> //ioremap, etc.

#include <linux/fs.h>

#include <linux/interrup.h> //request\_irq, etc.

#include <linux/irqreturn.h>

#include <linux/workqueue.h> //workqueue

MODULE\_LICENSE**(**"GPL"**);**

struct file\_operations key\_fops**;**

struct miscdevice mdev **=** **{**

**.**minor **=** 200**,**

**.**name **=** "mdevice"**,**

**.**fops **=** **&**key\_fops

**};**

static struct work\_struct work**;**

//static struct workqueue\_struct \*workqueue;

void key\_work **(**struct work\_struct **\***work**)**

**{**

printk**(**"key down\n"**);**

**return** **;**

**}**

static void key\_hw\_init **(**void**)**

**{**

//From your board:K1:EINT8:GPG0:

#define GPGCON 0x56000060

unsigned int **\***gpio\_config**;**

unsigned int data**;**

gpio\_config **=** **(**int **\*)**ioremap**(**GPGCON**,** 4**);**

data **=** readw**(**gpio\_config**);**

data **&=** **~(**0x3**);**

data **|=** 0x2**;**

writew **(**data**,** gpio\_config**);**

**return** **;**

**}**

irqreturn\_t key\_int**(**int irq**,** void **\***dev\_id**)**

**{**

//01.check whether irq or not for the device

printk**(**"check whether irq or not for the device\n"**);**

//02.clear irq issued

printk**(**"clear irq issued\n"**);**

//hand on work

INIT\_WORK**(&**work**,** key\_work**);**

schedule\_work**(&**work**);**

**return** 0**;**

**}**

static int key\_init **(**void**)**

**{**

misc\_register**(&**mdev**);**

request\_irq **(**IRQ\_EINT0**,** key\_int**,** IRQF\_TRIGGER\_FALLING**,** "mdevice"**,** 0**);**

//Device hardware initialization

dev\_hw\_init **();**

**return** 0**;**

**}**

static void key\_exit **(**void**)**

**{**

misc\_deregister **(&**mdev**);**

free\_irq **(**irq**,** 0**);**

**return** **;**

**}**

module\_init **(**key\_init**);**

module\_exit **(**key\_exit**);**

## 按键定时器祛痘

前面一课把我们的按键驱动程序放到开发板上安装测试时，我们会发现我们按一下按键，它会打印多条信息，这并不是我们想要的，我们只是需要按一下打印一条信息。这个其实就是由于这个按键的抖动造成的，那么什么是按键的抖动呢？我们按键所用的开关通常都是机械弹性开关，当机械触点断开闭合时，由于机械触点的弹性作用，开关不会马上的接通或断开。因而在闭合断开的瞬间总是伴随有一连串的抖动。

按键祛痘的方法有两种，一种是单片机中常用的电路祛痘，另一种就是软件的延时祛痘，软件延时方法可以是轮询和定时器。但是延时会大大影响处理器性能，所以我们在Linux中使用定时器来搞定。Linux内核使用struct timer\_list来描述一个定时器，其中比较重要成员是expires和function函数指针，expires是指定延时时间。

那么我们怎么来用这个定时器呢？首先定义定时器变量，其次是初始化，初始化中需要我们使用init\_timer()函数初始化和自己去设置超时函数，然后就是add\_timer来注册定时器，最后启动通过mod\_timer这个定时器。那么我们的定时器是不是循环计时呢？比如说我们定时器5秒过时后，会不会又从0开始计时呢？答案是不会的，我们的定时器只能执行一次，然后就失效了，如果你还想它延时，就再执行一次mod\_timer，前面的初始化也可以不用做了。注销定时器使用del\_timer来注销定时器。

一般初始化定时器的工作就放在模块的初始化当中来完成，首先我们初始化定时器第一步，就是初始化这个定时器init\_timer()，然后设置超时函数，其中最重要的是我们的启动定时器放在什么位置，这个我们必须要搞清楚启动定时器和超时函数之间的关系，也就是说根据超时函数和启动定时器的关系，我们把定时器要执行的函数就放到超时函数里实现了，那么这个定时器的启动就放到中断后半部的函数处理中来执行。所以中断后半部函数主要通过mod\_timer来间接执行按键打印工作。

定时器工作原理其实就是，在多次抖动中选择第一个产生的触发来执行计时，计时时间到了之后再来检查是否还有触发，有就执行超时函数，否则不执行。

#include <linux/init.h>

#include <linux/module.h>

#include <linux/miscdevice.h>

#include <linux/io.h> //ioremap, etc.

#include <linux/fs.h>

#include <linux/interrup.h> //request\_irq, etc.

#include <linux/irqreturn.h>

#include <linux/workqueue.h> //workqueue

#include <linux/timer.h> //timer

MODULE\_LICENSE**(**"GPL"**);**

//From your board:K1:EINT8:GPG0:

#define GPGCON 0x56000060

#define GPGDAT 0x56000064

unsigned int **\***gpio\_config**;**

unsigned int data**;**

struct file\_operations key\_fops**;**

struct miscdevice mdev **=** **{**

**.**minor **=** 200**,**

**.**name **=** "mdevice"**,**

**.**fops **=** **&**key\_fops

**};**

static struct work\_struct work**;**

static struct timer\_list key\_timer**;**

//static struct workqueue\_struct \*workqueue;

//void key\_work (struct work\_struct \*work)

//{

// printk("key down\n");

// return ;

//}

void key\_timer\_func**(**unsigned long data**)**

**{**

unsigned int key\_value**;**

gpio\_data **=** ioremap**(**GPGDAT**,** 4**);**

key\_value **=** readw **(**gpio\_data**);**

**if** **(**key\_value **==** 0**)**

printk**(**"Key down!\n"**);**

**return** **;**

**}**

void key\_work\_func **(**struct work\_struct **\***work**)**

**{**

mod\_timer **(&**key\_timer**,** jiffies**+**HZ**/**10**);** //1000/10 = 100ms

**return** **;**

**}**

static void key\_hw\_init **(**void**)**

**{**

gpio\_config **=** **(**int **\*)**ioremap**(**GPGCON**,** 4**);**

data **=** readw**(**gpio\_config**);**

data **&=** **~(**0x3**);**

data **|=** 0x2**;**

writew **(**data**,** gpio\_config**);**

**return** **;**

**}**

irqreturn\_t key\_int**(**int irq**,** void **\***dev\_id**)**

**{**

//01.check whether irq or not for the device

printk**(**"check whether irq or not for the device\n"**);**

//02.clear irq issued

printk**(**"clear irq issued\n"**);**

//hand on work

schedule\_work**(&**work**);**

**return** 0**;**

**}**

static int key\_init **(**void**)**

**{**

misc\_register**(&**mdev**);**

request\_irq **(**IRQ\_EINT0**,** key\_int**,** IRQF\_TRIGGER\_FALLING**,** "mdevice"**,** 0**);**

//Device hardware initialization

dev\_hw\_init **();**

INIT\_WORK**(&**work**,**key\_work\_func**);**

init\_timer**(&**key\_timer**);**

key\_timer**.**function **=** key\_timer\_func**;**

add\_timer **(&**key\_timer**);**

**return** 0**;**

**}**

static void key\_exit **(**void**)**

**{**

misc\_deregister **(&**mdev**);**

free\_irq **(**IRQ\_EINT0**,** 0**);**

**return** **;**

**}**

module\_init **(**key\_init**);**

module\_exit **(**key\_exit**);**

## 驱动支持多按键优化

我们首先要知道的理论是同一个设备可以注册多个中断，所以于是就有了我们对下面这个注册中断修改：

static int key\_init **(**void**)**

**{**

misc\_register**(&**mdev**);**

request\_irq **(**IRQ\_EINT0**,** key\_int**,** IRQF\_TRIGGER\_FALLING**,** "mdevice"**,** 0**);**

request\_irq **(**IRQ\_EINT11**,** key\_int**,** IRQF\_TRIGGER\_FALLING**,** "mdevice"**,** 0**);**

//Device hardware initialization

dev\_hw\_init **();**

INIT\_WORK**(&**work**,**key\_work\_func**);**

init\_timer**(&**key\_timer**);**

key\_timer**.**function **=** key\_timer\_func**;**

add\_timer **(&**key\_timer**);**

**return** 0**;**

**}**

然后我们继续检查，发现其它地方大多数不需要修改，中断上半部的硬件初始化部分需要修改一下，因为里面的寄存器相应位要添加相应的中断设置位，如下图所示：

static void key\_hw\_init **(**void**)**

**{**

gpio\_config **=** **(**int **\*)**ioremap**(**GPGCON**,** 4**);**

data **=** readw**(**gpio\_config**);**

data **&=** **~(**0x3**);**

data **|=** 0x2**;** //K1:GPG0:[1:0]

data **|=** **(**0x2**<<**6**);** //K2:GPG3:[7:6]

writew **(**data**,** gpio\_config**);**

**return** **;**

**}**

然后我们继续检查，发现下半部处理的时候需要分别检测KEY1和KEY2是否按下，根据原理图得知，按键按下为低电平，所以最后修改如下图：

void key\_timer\_func**(**unsigned long data**)**

**{**

unsigned int key\_value**;**

gpio\_data **=** ioremap**(**GPGDAT**,** 4**);**

key\_value **=** readw **(**gpio\_data**)&**0x01**;** //0001

**if** **(**key\_value **==** 0**)**

printk**(**"Key1 down!\n"**);**

key\_value **=** readw **(**gpio\_data**)&**0x04**;** //0100

**if** **(**key\_value **==** 0**)**

printk**(**"Key2 down!\n"**);**

**return** **;**

**}**

### 多按键驱动完整代码key.c

#include <linux/init.h>

#include <linux/module.h>

#include <linux/miscdevice.h>

#include <linux/io.h> //ioremap, etc.

#include <linux/fs.h>

#include <linux/interrup.h> //request\_irq, etc.

#include <linux/irqreturn.h>

#include <linux/workqueue.h> //workqueue

#include <linux/timer.h> //timer

MODULE\_LICENSE**(**"GPL"**);**

//From your board:K1:EINT8:GPG0:

#define GPGCON 0x56000060

#define GPGDAT 0x56000064

unsigned int **\***gpio\_config**;**

unsigned int data**;**

struct file\_operations key\_fops**;**

struct miscdevice mdev **=** **{**

**.**minor **=** 200**,**

**.**name **=** "mdevice"**,**

**.**fops **=** **&**key\_fops

**};**

static struct work\_struct work**;**

static struct timer\_list key\_timer**;**

//static struct workqueue\_struct \*workqueue;

//void key\_work (struct work\_struct \*work)

//{

// printk("key down\n");

// return ;

//}

void key\_timer\_func**(**unsigned long data**)**

**{**

unsigned int key\_value**;**

gpio\_data **=** ioremap**(**GPGDAT**,** 4**);**

key\_value **=** readw **(**gpio\_data**)&**0x01**;** //0001

**if** **(**key\_value **==** 0**)**

printk**(**"Key1 down!\n"**);**

key\_value **=** readw **(**gpio\_data**)&**0x04**;** //0100

**if** **(**key\_value **==** 0**)**

printk**(**"Key2 down!\n"**);**

**return** **;**

**}**

void key\_work\_func **(**struct work\_struct **\***work**)**

**{**

mod\_timer **(&**key\_timer**,** jiffies**+**HZ**/**10**);** //1000/10 = 100ms

**return** **;**

**}**

static void key\_hw\_init **(**void**)**

**{**

gpio\_config **=** **(**int **\*)**ioremap**(**GPGCON**,** 4**);**

data **=** readw**(**gpio\_config**);**

data **&=** **~(**0x3**);**

data **|=** 0x2**;** //K1:GPG0:[1:0]

data **|=** **(**0x2**<<**6**);** //K2:GPG3:[7:6]

writew **(**data**,** gpio\_config**);**

**return** **;**

**}**

irqreturn\_t key\_int**(**int irq**,** void **\***dev\_id**)**

**{**

//01.check whether irq or not for the device

printk**(**"check whether irq or not for the device\n"**);**

//02.clear irq issued

printk**(**"clear irq issued\n"**);**

//hand on work

schedule\_work**(&**work**);**

**return** 0**;**

**}**

static int key\_init **(**void**)**

**{**

misc\_register**(&**mdev**);**

request\_irq **(**IRQ\_EINT0**,** key\_int**,** IRQF\_TRIGGER\_FALLING**,** "mdevice"**,** 0**);**

request\_irq **(**IRQ\_EINT11**,** key\_int**,** IRQF\_TRIGGER\_FALLING**,** "mdevice"**,** 0**);**

//Device hardware initialization

dev\_hw\_init **();**

INIT\_WORK**(&**work**,**key\_work\_func**);**

init\_timer**(&**key\_timer**);**

key\_timer**.**function **=** key\_timer\_func**;**

add\_timer **(&**key\_timer**);**

**return** 0**;**

**}**

static void key\_exit **(**void**)**

**{**

misc\_deregister **(&**mdev**);**

free\_irq **(**IRQ\_EINT0**,** 0**);**

**return** **;**

**}**

module\_init **(**key\_init**);**

module\_exit **(**key\_exit**);**

### 按键应用层测试代码key\_app.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <fcntl.h>

#include <unistd.h>

int main **(**void**)**

**{**

int fd**,** key\_number**;**

fd **=** open **(**"/dev/my\_device200"**,** O\_RDONLY**);**

**if** **(**fd **<** 0**)**

**{**

printf **(**"dev/my\_device200 open error!\n"**);**

exit **(**1**);**

**}**

read **(**fd**,** **&**key\_number**,** 4**);**

**if** **(**key\_number **==** 1**)**

printf **(**"key number: %d\n"**,** key\_number**);**

**else** **if** **(**key\_number **==** 2**)**

printf **(**"key number: %d\n"**,** key\_number**);**

**else**

printf **(**"key number: %d\n"**,** key\_number**);**

**return** 0**;**

**}**

### 相应的Makefile

obj-m **:=** key.o

KDIR **:=** /home/S5-driver/lesson7/linux-tq2440/

all**:**

make -C $(KDIR) M=$(PWD) modules CROSS\_COMPILE=arm-linux- ARCH=arm

clean**:**

rm -f \*.ko \*.o \*.mod.o \*.mod.c \*.symvers \*.bak \*.order

## 阻塞型驱动设计

其中比较重要的就是读写阻塞和等待队列的概念（候车室睡觉）。

阻塞存在必要性

当一个设备无法立刻满足用户的读写请求时应当如何处理？

例如：调用read时，设备没有数据提供，但以后可能会有；或者一个进程试图向设备写入数据，但是设备暂时没有准备好接收数据。当上述情况发生的时候，驱动程序应当阻塞进程，使它进入等待状态，直到请求可以得到满足。就好像坐公交车，到了车站但没有公交车，所以我们只有等待。

阻塞型驱动的访问模型

A进程向mem读数据，但mem暂时没数据，于是mem驱动将A放入候车室并让其睡眠，过段时间之后mem有数据了，于是mem驱动就从候车室中唤醒A并让A读取数据。

驱动程序有阻塞应用程序的责任，特别是进程读写设备数据的时候，驱动程序就必须把进程放进等待队列里并让其睡眠，当条件为真时，驱动再去唤醒队列里的进程，并让其出来执行读写操作。

内核等待队列

在实现阻塞驱动的过程中，也需要有一个候车室来安排被阻塞的进程休息，当唤醒它们的条件成熟时，则可以从候车室中将这些进程唤醒。而这个候车室就是等待队列。

**注意：我们所有驱动程序的读写功能都要实现阻塞型的驱动模型！,也就是说我们前面的所有驱动设计模型都是不对的。**

**step01 定义等待队列**

**wait\_queue\_head\_t my\_queue;**

**step02 初始化等待队列**

**init\_waitqueue\_head(&my\_queue);**

**step02 定义+初始化等待队列**

**DECLARE\_WAIT\_QUEUE\_HEAD(my\_queue);**

step03 进入等待队列，睡眠

**wait\_event (queue, condition);**

当condition为真时，立即返回；否则让进程进入**TASK\_UNINTERRUPTIBLE**模式的睡眠，并挂在queue参数所指定的等待队列上。

step03

**wait\_event\_interruptible(queue,condition);**

当condition为真时，立即返回；否则让进程进入**TASK\_INTERRUPTIBLE**模式的睡眠，并挂在queue参数所指定的等待队列上。

step03

**int wait\_event\_killable(queue, condition);**

当condition为真时，立即返回；否则让进程进入TASK\_KILLABLE的睡眠，并挂在queue参数所指定的等待队列上。

**step04 从等待队列中唤醒进程**

**wake\_up(wait\_queue\_t \*q);**

从等待队列q中唤醒状态为TASK\_UNINTERRUPTIBLE, ASK\_INTERRUPTIBLE, ASK\_INTERRUPTIBLE的所有进程。

**wake\_up\_interruptible(wait\_queue\_t \*q);**

从等待队列q中唤醒状态为TASK\_INTERRUPTIBLE的进程。

首先我们要初始化等待队列，这步操作一般都是在模块初始化中完成，如下图所示：

static int key\_init **(**void**)**

**{**

my\_class **=** class\_create **(**THIS\_MODULE**,** "my\_device"**);**

device\_create **(**my\_class**,** **NULL,** MKDEV**(**10**,**200**),** **NULL,** "my\_device%d"**,** 200**);**

misc\_register**(&**mdev**);**

request\_irq **(**IRQ\_EINT0**,** key\_int**,** IRQF\_TRIGGER\_FALLING**,** "mdevice"**,** 0**);**

request\_irq **(**IRQ\_EINT11**,** key\_int**,** IRQF\_TRIGGER\_FALLING**,** "mdevice"**,** 0**);**

//Device hardware initialization

dev\_hw\_init **();**

INIT\_WORK**(&**work**,**key\_work\_func**);**

init\_timer**(&**key\_timer**);**

key\_timer**.**function **=** key\_timer\_func**;**

add\_timer **(&**key\_timer**);**

init\_waitqueue\_head **(&**wait\_queue**);**

**return** 0**;**

**}**

其次我们在驱动的读写操作中假设没有数据读写，来设立队列的等待，并在读写完成后再把条件值改回等待需要的条件，如下图所示：

ssize\_t key\_read **(**struct file **\***filp**,** char \_\_user **\***buf**,** size\_t size**,** loff\_t **\***pos**)**

**{**

//if condition be true, continue, or not be waited

//following: if key\_number !=0, go to execute following code

wait\_event **(**wait\_queue**,** dev\_number **!=**0**);**

copy\_to\_user**(**buf**,** **&**key\_number**,** 4**);**

key\_number **=** 0**;**

**return** 0**;**

**}**

最后我们要在合适的地方唤醒，这里我们根据应用程序要读的是dev\_number值，所以我们只要找到驱动中设置dev\_number值的位置就可以了，也就是说，启动中设置dev\_number值之后就可以唤醒应用程序了，如下图所示：

void key\_timer\_func**(**unsigned long data**)**

**{**

unsigned int key\_value**;**

gpio\_data **=** ioremap**(**GPGDAT**,** 4**);**

key\_value **=** readw **(**gpio\_data**)&**0x01**;** //0001

**if** **(**key\_value **==** 0**)**

**{**

key\_number **=** 1**;**

printk**(**"Key1 down!\n"**);**

**}**

key\_value **=** readw **(**gpio\_data**)&**0x04**;** //0100

**if** **(**key\_value **==** 0**)**

**{**

key\_number **=** 2**;**

printk**(**"Key2 down!\n"**);**

**}**

wake\_up **(&**wait\_queue**);** //wake up queue if key\_number be value

**return** **;**

**}**

下面我们根据出错提示来补充相应头文件和定义全局变量，如下图所示：

#include <linux/init.h>

#include <linux/module.h>

#include <linux/miscdevice.h>

#include <linux/io.h> //ioremap, etc.

#include <linux/fs.h>

#include <linux/interrup.h> //request\_irq, etc.

#include <linux/irqreturn.h>

#include <linux/workqueue.h> //workqueue

#include <linux/timer.h> //timer

MODULE\_LICENSE**(**"GPL"**);**

//From your board:K1:EINT8:GPG0:

#define GPGCON 0x56000060

#define GPGDAT 0x56000064

unsigned int **\***gpio\_config**;**

unsigned int data**;**

static struct work\_struct work**;**

static struct timer\_list key\_timer**;**

struct class **\***my\_class**;**

unsigned int key\_number**;**

wait\_queue\_head\_t wait\_queue**;**

struct file\_operations key\_fops**;**

## 阻塞型按键驱动完整代码key.c

#include <linux/init.h>

#include <linux/module.h>

#include <linux/miscdevice.h>

#include <linux/io.h> //ioremap, etc.

#include <linux/fs.h>

#include <linux/interrup.h> //request\_irq, etc.

#include <linux/irqreturn.h>

#include <linux/workqueue.h> //workqueue

#include <linux/timer.h> //timer

MODULE\_LICENSE**(**"GPL"**);**

//From your board:K1:EINT8:GPG0:

#define GPGCON 0x56000060

#define GPGDAT 0x56000064

unsigned int **\***gpio\_config**;**

unsigned int data**;**

static struct work\_struct work**;**

static struct timer\_list key\_timer**;**

struct class **\***my\_class**;**

unsigned int key\_number**;**

wait\_queue\_head\_t wait\_queue**;**

struct file\_operations key\_fops**;**

ssize\_t key\_read **(**struct file **\***filp**,** char \_\_user **\***buf**,** size\_t size**,** loff\_t **\***pos**)**

**{**

//if condition be true, continue, or not be waited

//following: if key\_number !=0, go to execute following code

wait\_event **(**wait\_queue**,** dev\_number **!=**0**);**

copy\_to\_user**(**buf**,** **&**key\_number**,** 4**);**

key\_number **=** 0**;**

**return** 0**;**

**}**

struct miscdevice mdev **=** **{**

**.**minor **=** 200**,**

**.**name **=** "mdevice"**,**

**.**fops **=** **&**key\_fops

**};**

static struct work\_struct work**;**

static struct timer\_list key\_timer**;**

//static struct workqueue\_struct \*workqueue;

//void key\_work (struct work\_struct \*work)

//{

// printk("key down\n");

// return ;

//}

void key\_timer\_func**(**unsigned long data**)**

**{**

unsigned int key\_value**;**

gpio\_data **=** ioremap**(**GPGDAT**,** 4**);**

key\_value **=** readw **(**gpio\_data**)&**0x01**;** //0001

**if** **(**key\_value **==** 0**)**

**{**

key\_number **=** 1**;**

printk**(**"Key1 down!\n"**);**

**}**

key\_value **=** readw **(**gpio\_data**)&**0x04**;** //0100

**if** **(**key\_value **==** 0**)**

**{**

key\_number **=** 2**;**

printk**(**"Key2 down!\n"**);**

**}**

wake\_up **(&**wait\_queue**);** //wake up queue if key\_number be value

**return** **;**

**}**

void key\_work\_func **(**struct work\_struct **\***work**)**

**{**

mod\_timer **(&**key\_timer**,** jiffies**+**HZ**/**10**);** //1000/10 = 100ms

**return** **;**

**}**

static void key\_hw\_init **(**void**)**

**{**

gpio\_config **=** **(**int **\*)**ioremap**(**GPGCON**,** 4**);**

data **=** readw**(**gpio\_config**);**

data **&=** **~(**0x3**);**

data **|=** 0x2**;** //K1:GPG0:[1:0]

data **|=** **(**0x2**<<**6**);** //K2:GPG3:[7:6]

writew **(**data**,** gpio\_config**);**

**return** **;**

**}**

irqreturn\_t key\_int**(**int irq**,** void **\***dev\_id**)**

**{**

//01.check whether irq or not for the device

printk**(**"check whether irq or not for the device\n"**);**

//02.clear irq issued

printk**(**"clear irq issued\n"**);**

//hand on work

schedule\_work**(&**work**);**

**return** 0**;**

**}**

static int key\_init **(**void**)**

**{**

my\_class **=** class\_create **(**THIS\_MODULE**,** "my\_device"**);**

device\_create **(**my\_class**,** **NULL,** MKDEV**(**10**,**200**),** **NULL,** "my\_device%d"**,** 200**);**

misc\_register**(&**mdev**);**

request\_irq **(**IRQ\_EINT0**,** key\_int**,** IRQF\_TRIGGER\_FALLING**,** "mdevice"**,** 0**);**

request\_irq **(**IRQ\_EINT11**,** key\_int**,** IRQF\_TRIGGER\_FALLING**,** "mdevice"**,** 0**);**

//Device hardware initialization

dev\_hw\_init **();**

INIT\_WORK**(&**work**,**key\_work\_func**);**

init\_timer**(&**key\_timer**);**

key\_timer**.**function **=** key\_timer\_func**;**

add\_timer **(&**key\_timer**);**

init\_waitqueue\_head **(&**wait\_queue**);**

**return** 0**;**

**}**

static void key\_exit **(**void**)**

**{**

misc\_deregister **(&**mdev**);**

free\_irq **(**IRQ\_EINT0**,** 0**);**

**return** **;**

**}**

module\_init **(**key\_init**);**

module\_exit **(**key\_exit**);**

下面的应用程序就是上面驱动程序的测试程序，代码如下图所示：

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <fcntl.h>

#include <unistd.h>

int main **(**void**)**

**{**

int fd**,** key\_number**;**

fd **=** open **(**"/dev/my\_device200"**,** O\_RDONLY**);**

**if** **(**fd **<** 0**)**

**{**

printf **(**"dev/my\_device200 open error!\n"**);**

exit **(**1**);**

**}**

read **(**fd**,** **&**key\_number**,** 4**);**

**if** **(**key\_number **==** 1**)**

printf **(**"key number: %d\n"**,** key\_number**);**

**else** **if** **(**key\_number **==** 2**)**

printf **(**"key number: %d\n"**,** key\_number**);**

**else**

printf **(**"key number: %d\n"**,** key\_number**);**

**return** 0**;**

**}**

# 专题02-总线设备驱动模型

## 总线设备驱动模型

随着技术的不断进步，系统的拓扑结构也越来越复杂，对热插拔，跨平台移植性的要求也越来越高，2.4内核已经难以满足这些需求。为了适应这种形势的需要，从2.6内核开始提供了全新的设备模型。

总线上面要挂载很多驱动，比如USB总线上，挂了USB鼠标驱动，挂了USB网卡驱动，挂了USB键盘驱动，总线的功能是能感知有新的设备上线了，比如新插入USB网卡了，于是USB总线从驱动列表中通过匹配来找到这个新插入的网卡驱动，比如该USB网卡拔掉了，USB总线也要对拔掉这个动作做相应的处理。

首先在这个模型当中会有一条总线，为了形象点，我们这里以USB总线为例，那么USB总线上面我们首先要事先挂载很多驱动上去，比如USB的鼠标驱动，USB的键盘驱动，USB网卡驱动等等。我们这里要事先介绍一下总线的功能，总线功能之一就是要能够感知有新的设备上到总线上，比如我们这里插入新网卡，如果感知到新设备USB网卡，总线就会逐个把事先挂载的驱动和这个新设备进行匹配，匹配成功之后，它就把控制权交给网卡驱动程序来处理USB的网卡了；同样当你从设备上把USB网卡拔掉以后，USB总线同样也要能够感知到，然后找到相应驱动程序，该驱动程序来处理模块卸载方面的处理。

从上面的流程描述得知，我们提取出重要的三个元素，就是总线、设备、驱动，下面我们接下来就围绕这三个元素来详解，

总线：描述结构

总线的描述结构定义在<linux/device.h>中，以bus\_type来描述一个总线，其中重要的成员分别是\*name，和 (\*match)，前者是总线名称，后者是驱动与设备的匹配函数。总线的注册与注销非常简单，总线注册使用bus\_register()函数即可，注册之后可以在/sys/bus目录下查看到相应目录；总线注销使用bus\_unregister()函数，这些所需要的头文件就linux/device.h, linux/kernel.h即可。

总线中需要包含的头文件，如下图：

#include <linux/init.h>

#include <linux/module.h>

#include <linux/miscdevice.h>

#include <linux/io.h> //ioremap, etc.

#include <linux/fs.h>

#include <linux/interrup.h> //request\_irq, etc.

#include <linux/irqreturn.h>

#include <linux/workqueue.h> //workqueue

#include <linux/timer.h> //timer

#include <linux/device.h>

#include <asm/uaccess.h>

#include <linux/wait.h>

#include <linux/sched.h>

#include <linux/kernel.h>

其次，初始化一个总线描述结构所定义的变量，如下图：

int mymatch **(**struct device **\***dev**,** struct device\_driver **\***drv**)**

**{**

**return** 0**;**

**}**

struct bus\_type mybus **=** **{**

**.**name **=** "mybus"**,**

**.**match **=** mymatch**,**

**};**

最后驱动的注册和注销当中也要加上总线的注册和注销工作，如下图：

static int key\_init **(**void**)**

**{**

my\_class **=** class\_create **(**THIS\_MODULE**,** "my\_device"**);**

device\_create **(**my\_class**,** **NULL,** MKDEV**(**10**,**200**),** **NULL,** "my\_device%d"**,** 200**);**

misc\_register**(&**mdev**);**

request\_irq **(**IRQ\_EINT0**,** key\_int**,** IRQF\_TRIGGER\_FALLING**,** "mdevice"**,** 0**);**

request\_irq **(**IRQ\_EINT11**,** key\_int**,** IRQF\_TRIGGER\_FALLING**,** "mdevice"**,** 0**);**

//Device hardware initialization

dev\_hw\_init **();**

INIT\_WORK**(&**work**,**key\_work\_func**);**

init\_timer**(&**key\_timer**);**

key\_timer**.**function **=** key\_timer\_func**;**

add\_timer **(&**key\_timer**);**

init\_waitqueue\_head **(&**wait\_queue**);**

bus\_register **(&**mybus**);**

**return** 0**;**

**}**

static void key\_exit **(**void**)**

**{**

misc\_deregister **(&**mdev**);**

free\_irq **(**IRQ\_EINT0**,** 0**);**

del\_timer **(&**key\_timer**);**

device\_destroy **(**my\_class**,** MKDEV**(**10**,** 200**));**

class\_destroy**(**my\_class**);**

bus\_unregister **(&**mybus**);**

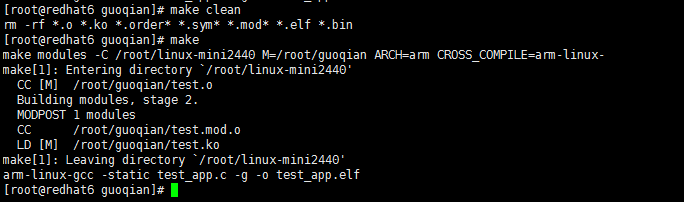
**return** **;**

**}**

module\_init **(**key\_init**);**

module\_exit **(**key\_exit**);**

以上代码流程就是我们为总线搭建的框架，接下来的工作就是基于上面的总线框架来进行的。我们搭建好总线框架后，然后就是编译调试把错误的地方解决，最后编译成功，如下图：



下面是驱动的描述结构device\_driver，其中比较重要的成员是\*name,这是驱动名，然后就是\*bus, 这是该驱动所在的总线名，然后就是(\*probe)函数，该函数是检测设备热插拔的检测函数，下面就是驱动的注册与注销操作，函数分别是driver\_register和driver\_unregister函数，根据我们总线和驱动的关心，总线被挂载上去之后，那么驱动也随之挂载上去，

我们根据上面分析的流程来，先初始化驱动描述结构，如下图所示：

int myprobe **(**struct device **\***dev**)**

**{**

printk**(**"driver has found the device can be handled\n"**);**

**return** 0**;**

**}**

extern struct bus\_type mybus**;**

struct device\_driver mydriver **=** **{**

**.**name **=** "mydev"**,**

**.**bus **=** **&**mybus**,**

**.**probe **=** myprobe

**};**

然后在注册和注销模块时候注销这个驱动，如下图所示：

static int key\_init **(**void**)**

**{**

my\_class **=** class\_create **(**THIS\_MODULE**,** "my\_device"**);**

device\_create **(**my\_class**,** **NULL,** MKDEV**(**10**,**200**),** **NULL,** "my\_device%d"**,** 200**);**

misc\_register**(&**mdev**);**

request\_irq **(**IRQ\_EINT0**,** key\_int**,** IRQF\_TRIGGER\_FALLING**,** "mdevice"**,** 0**);**

request\_irq **(**IRQ\_EINT11**,** key\_int**,** IRQF\_TRIGGER\_FALLING**,** "mdevice"**,** 0**);**

//Device hardware initialization

dev\_hw\_init **();**

INIT\_WORK**(&**work**,**key\_work\_func**);**

init\_timer**(&**key\_timer**);**

key\_timer**.**function **=** key\_timer\_func**;**

add\_timer **(&**key\_timer**);**

init\_waitqueue\_head **(&**wait\_queue**);**

bus\_register **(&**mybus**);**

driver\_register **(&**mydriver**);**

**return** 0**;**

**}**

static void key\_exit **(**void**)**

**{**

misc\_deregister **(&**mdev**);**

free\_irq **(**IRQ\_EINT0**,** 0**);**

del\_timer **(&**key\_timer**);**

device\_destroy **(**my\_class**,** MKDEV**(**10**,** 200**));**

class\_destroy**(**my\_class**);**

bus\_unregister **(&**mybus**);**

driver\_unregister **(&**mydriver**);**

**return** **;**

**}**

然后我们还要修改bus.c里面的相应的bus描述结构，因为我们的驱动就挂载在这个我们前面定义的总线上，所以我们这个驱动就需要使用这个总线，所以必须在bus.c中到处mybus这个符号，且driver.c中也要声明mybus这个变量：如下图所示：

下面是bus.c的相应修改：

int mymatch **(**struct device **\***dev**,** struct device\_driver **\***drv**)**

**{**

**return** 0**;**

**}**

struct bus\_type mybus **=** **{**

**.**name **=** "mybus"**,**

**.**match **=** mymatch**,**

**};**

EXPORT\_SYMBOL\_GPL**(**mybus**);**

下面是driver.c的相应修改：

int myprobe **(**struct device **\***dev**)**

**{**

printk**(**"driver has found the device can be handled\n"**);**

**return** 0**;**

**}**

extern struct bus\_type mybus**;** //extern from bus.c

struct device\_driver mydriver **=** **{**

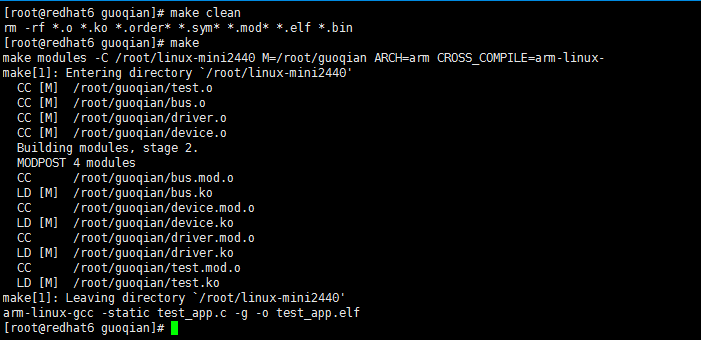
**.**name **=** "mydev"**,**

**.**bus **=** **&**mybus**,**

**.**probe **=** myprobe

**};**

上面的总线和驱动的框架搭建好之后，我们编译调试并得出最终编译成功，如下图：



好，下面我们继续来添加设备这个模块文件，首先我们要知道设备和总线以及驱动之间的关系，根据前面的理论知识我们得知，设备是热插拔的，且总线大多数根据设备名和驱动中对应的驱动名来匹配设备和驱动从而能是新插入的设备能够正常工作，所以我们接下来就是来构建设备这个模块文件。

设备的描述结构是device，其中比较重要的成员是init\_name，这个成员就是设备名，它必须和匹配的驱动名相同，其次就是bus成员，这个成员主要指定该设备属于哪条总线的；设备的注册和注销也非常简单，就是分别使用device\_register和device\_unregister函数来实现设备的注册和注销工作。

根据上面的结论，我们可以总结出来该驱动模块的编写流程，首先我们要初始化这个设备描述结构，其次就是完善模块挂载和卸载的部分。

首先我们来初始化这个设备描述结构，如下图：

extern struct bus\_type mybus**;**

struct device mydev **=** **{**

**.**init\_name **=** "mydevice"**,**

**.**bus **=** **&**mybus**,**

**};**

然后我们在驱动注册和注销模块分别填写设备的注册和注销工作，如下图：

static int dev\_init **(**void**)**

**{**

device\_register **(&**mydev**);**

**return** 0**;**

**}**

static void dev\_exit **(**void**)**

**{**

device\_unregister **(&**mydev**);**

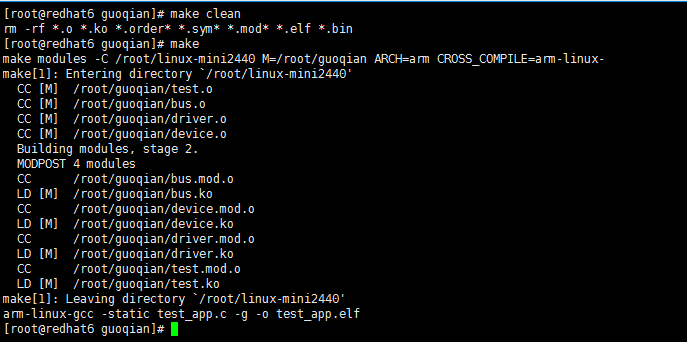
**return** **;**

**}**

module\_init **(**dev\_init**);**

module\_exit **(**dev\_exit**);**

最后，我们总体的总线模块、驱动模块、设备模块的基本框架都搭建完工！接下来我们就是编译调试并得出最终编译成功的结果，如下图所示：



当总线感知到一个设备去加入总线的时候，该总线会把上面的所有驱动都拿来和这个设备进行匹配，如果匹配成功，那么总线就会调用驱动里的probe函数；其中总线为设备和驱动匹配的功能主要是因为总线里面的match函数，就是调用match函数来进行匹配操作，当然不同总线匹配驱动和设备的方法不一样，比如说我们的USB总线，因为我们每一个USB设备里面都有一个该设备的ID，那么我们可以通过硬件方面的标识去匹配，那如果是虚拟总线，比如平台总线，那么我们通常会给这个设备去定义一个名字，我给设备和驱动都定义相同名字来作为匹配规则。

下面我们就来实现总线模块中的match函数了：

int mymatch**(**struct device **\***dev**,** struct device\_driver **\***drv**)**

**{**

**return** **(!**strcmp**(**dev**->**init\_name**,** drv**->**name**));**

**}**

struct bus\_type mybus **=** **{**

**.**name **=** "mybus"**,**

**.**match **=** mymatch**,**

**};**

EXPORT\_SYMBOL\_GPL**(**mybus**);**

**在开发板上挂载时候出错了，出现了段错误，并打印出一大堆段错误信息，我们要学会查回溯表，错误信息回溯表的最开始处就是出错源，下面就是调用它的函数。**

此处要特别注意，从流程分析中发现，dev->init\_name被赋值给device结构另一个成员，并清空这个init\_name，所以我们把焦点关注到dev->kobj.name上，如下图：

int mymatch**(**struct device **\***dev**,** struct device\_driver **\***drv**)**

**{**

**return** **(!**strcmp**(**dev**->**obj**.**name**,** drv**->**name**));**

**}**

struct bus\_type mybus **=** **{**

**.**name **=** "mybus"**,**

**.**match **=** mymatch**,**

**};**

EXPORT\_SYMBOL\_GPL**(**mybus**);**

### 总线文件：key\_bus.c

/\*

\* test.c

\* Copyright (c) 2017 Your Name <your@mail>

\*

\* This program is free software: you can redistribute it and/or modify

\* it under the terms of the GNU General Public License as published by

\* the Free Software Foundation, either version 3 of the License, or

\* (at your option) any later version.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program. If not, see <http://www.gnu.org/licenses/>.

\*/

#include <linux/init.h>

#include <linux/module.h>

#include <linux/miscdevice.h> //miscdevice

#include <linux/io.h> //ioremap,readw,writew,etc.

#include <linux/fs.h> //file\_operations

#include <linux/interrupt.h> //request\_irq,etc.

#include <linux/workqueue.h> //workqueue

#include <linux/timer.h> //timer

#include <linux/device.h> //class, device\_create

#include <asm/uaccess.h> //copy\_to\_user

#include <linux/wait.h> //wait queue (block)

#include <linux/sched.h> //TASK\_UNINTERRUPTIBLE

#include <linux/device.h> //bus relative

#include <linux/kernel.h> //bus relative

//From your board:K1:EINT8:GPG0:

#define GPGCON 0x56000060

#define GPGDAT 0x56000064

unsigned int **\***gpio\_config**;**

unsigned int **\***gpio\_data**;**

static struct work\_struct work**;**

static struct timer\_list dev\_timer**;**

struct class **\***my\_class**;**

unsigned int dev\_number**;**

wait\_queue\_head\_t wait\_queue**;**

//static struct workqueue\_struct \*workqueue;

MODULE\_LICENSE**(**"GPL"**);**

int mymatch**(**struct device **\***dev**,** struct device\_driver **\***drv**)**

**{**

**return** **(!**strcmp **(**dev**->**kobj**.**name**,** drv**->**name**));**

**}**

struct bus\_type mybus **=** **{**

**.**name **=** "mybus"**,**

**.**match **=** mymatch**,**

**};**

EXPORT\_SYMBOL\_GPL**(**mybus**);**

static int dev\_open **(**struct inode **\***node**,** struct file **\***filp**)**

**{**

**return** 0**;**

**}**

ssize\_t dev\_read **(**struct file **\***filp**,** char \_\_user **\***buf**,** size\_t size**,** loff\_t **\***pos**)**

**{**

//if condition be true, continue, or not be waited

//Following: if dev\_number!=0, go to execute following code

wait\_event **(**wait\_queue**,** dev\_number**!=**0**);**

copy\_to\_user**(**buf**,** **&**dev\_number**,** 4**);**

dev\_number **=** 0**;** //after read, clear this number

**return** 0**;**

**}**

struct file\_operations dev\_fops **=** **{**

**.**open **=** dev\_open**,**

**.**read **=** dev\_read**,**

**};**

struct miscdevice dev **=** **{**

**.**minor **=** 200**,**

**.**name **=** "device"**,**

**.**fops **=** **&**dev\_fops**,**

**};**

void dev\_timer\_func**(**unsigned long data**)**

**{**

unsigned int key\_value**;**

gpio\_data **=** ioremap **(**GPGDAT**,** 4**);**

key\_value **=** readw **(**gpio\_data**)&**0x01**;**//0001

**if** **(**key\_value **==** 0**)**

**{**

dev\_number **=** 1**;**

printk **(**"Key1 down!\n"**);**

**}**

key\_value **=** readw **(**gpio\_data**)&**0x04**;**//0100

**if** **(**key\_value **==** 0**)**

**{**

dev\_number **=** 2**;**

printk **(**"Key2 down!\n"**);**

**}**

wake\_up **(&**wait\_queue**);** //wake up queue if dev\_number be valued

**return** **;**

**}**

void dev\_work\_func**(**struct work\_struct **\***work**)**

**{**

mod\_timer **(&**dev\_timer**,** jiffies **+** HZ**/**10**);** //1000/10 = 100ms

**return** **;**

**}**

void dev\_hw\_init**(**void**)**

**{**

unsigned int data**;**

//Physical address --> virtual address

gpio\_config **=** **(**int **\*)**ioremap**(**GPGCON**,** 4**);**

data **=** readw**(**gpio\_config**);**

data **&=** **~(**0x3**);**

data **|=** 0x2**;** //K1:GPG0:[1:0]

data **|=** **(**0x2**<<**6**);** //K2:GPG3:[7:6]

writew **(**data**,** gpio\_config**);**

**return** **;**

**}**

irqreturn\_t dev\_interrupt**(**int irq**,** void **\***dev\_id**)**

**{**

//01.check whether irq or not for the device

printk**(**"check whether irq or not for the device\n"**);**

//02.clear irq issued

printk**(**"clear irq issued\n"**);**

//03.print value of key

schedule\_work**(&**work**);**

**return** 0**;**

**}**

static int dev\_init**(**void**)**

**{**

my\_class **=** class\_create **(**THIS\_MODULE**,** "my\_device"**);**

device\_create**(**my\_class**,** **NULL,** MKDEV**(**10**,** 200**),** **NULL,** "my\_device%d"**,** 200**);**

misc\_register**(&**dev**);**

//EINT8(k1), EINT11(k2)

request\_irq **(**IRQ\_EINT8**,** dev\_interrupt**,** IRQF\_TRIGGER\_FALLING**,** "device"**,** 0**);** //Falling down

request\_irq **(**IRQ\_EINT11**,**dev\_interrupt**,** IRQF\_TRIGGER\_FALLING**,** "device"**,** 0**);** //Falling down

//device hardware initialization:

dev\_hw\_init**();**

INIT\_WORK**(&**work**,** dev\_work\_func**);**

//init timer

init\_timer **(&**dev\_timer**);**

dev\_timer**.**function **=** dev\_timer\_func**;**

//register timer

add\_timer **(&**dev\_timer**);**

init\_waitqueue\_head **(&**wait\_queue**);**

bus\_register **(&**mybus**);**

**return** 0**;**

**}**

static void dev\_exit**(**void**)**

**{**

misc\_deregister**(&**dev**);**

free\_irq **(**IRQ\_EINT0**,** 0**);**

del\_timer **(&**dev\_timer**);**

device\_destroy**(**my\_class**,** MKDEV**(**10**,** 200**));**

class\_destroy**(**my\_class**);**

bus\_unregister **(&**mybus**);**

**return** **;**

**}**

module\_init **(**dev\_init**);**

module\_exit **(**dev\_exit**);**

### 驱动文件：key\_driver.c

/\*

\* test.c

\* Copyright (c) 2017 Your Name <your@mail>

\*

\* This program is free software: you can redistribute it and/or modify

\* it under the terms of the GNU General Public License as published by

\* the Free Software Foundation, either version 3 of the License, or

\* (at your option) any later version.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program. If not, see <http://www.gnu.org/licenses/>.

\*/

#include <linux/init.h>

#include <linux/module.h>

#include <linux/miscdevice.h> //miscdevice

#include <linux/io.h> //ioremap,readw,writew,etc.

#include <linux/fs.h> //file\_operations

#include <linux/interrupt.h> //request\_irq,etc.

#include <linux/workqueue.h> //workqueue

#include <linux/timer.h> //timer

#include <linux/device.h> //class, device\_create

#include <linux/wait.h> //wait queue (block)

#include <linux/sched.h> //TASK\_UNINTERRUPTIBLE

#include <linux/device.h> //bus relative

#include <linux/kernel.h> //bus relative

MODULE\_LICENSE**(**"GPL"**);**

int myprobe **(**struct device **\***dev**)**

**{**

printk **(**"driver has found the device can be handle\n"**);**

**return** 0**;**

**}**

extern struct bus\_type mybus**;** //extern from bus.c

struct device\_driver mydriver **=** **{**

**.**name **=** "mydev"**,**

**.**bus **=** **&**mybus**,**

**.**probe **=** myprobe

**};**

static int dev\_init**(**void**)**

**{**

driver\_register **(&**mydriver**);**

**return** 0**;**

**}**

static void dev\_exit**(**void**)**

**{**

driver\_unregister **(&**mydriver**);**

**return** **;**

**}**

module\_init **(**dev\_init**);**

module\_exit **(**dev\_exit**);**

### 设备文件：key\_device.c

/\*

\* key\_device.c

\* Copyright (c) 2017 seafly <seafly0616@qq.com>

\*

\* This program is free software: you can redistribute it and/or modify

\* it under the terms of the GNU General Public License as published by

\* the Free Software Foundation, either version 3 of the License, or

\* (at your option) any later version.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program. If not, see <http://www.gnu.org/licenses/>.

\*/

#include <linux/init.h>

#include <linux/module.h>

#include <linux/miscdevice.h> //miscdevice

#include <linux/io.h> //ioremap,readw,writew,etc.

#include <linux/fs.h> //file\_operations

#include <linux/interrupt.h> //request\_irq,etc.

#include <linux/workqueue.h> //workqueue

#include <linux/timer.h> //timer

#include <linux/device.h> //class, device\_create

#include <linux/wait.h> //wait queue (block)

#include <linux/sched.h> //TASK\_UNINTERRUPTIBLE

#include <linux/device.h> //bus relative

#include <linux/kernel.h> //bus relative

MODULE\_LICENSE**(**"GPL"**);**

extern struct bus\_type mybus**;** //extern from bus.c

struct device mydev **=** **{**

**.**init\_name **=** "mydevice"**,**

**.**bus **=** **&**mybus**,**

**};**

static int dev\_init**(**void**)**

**{**

device\_register **(&**mydev**);**

**return** 0**;**

**}**

static void dev\_exit**(**void**)**

**{**

device\_unregister **(&**mydev**);**

**return** **;**

**}**

module\_init **(**dev\_init**);**

module\_exit **(**dev\_exit**);**

总结：上面的实验是先挂载驱动，再挂载设备，那我们后面的实验于是先挂载设备再挂载驱动同样也能成功，所以两种都可行。

## 平台设备驱动设计

平台总线概述

平台总线是Linux2.6内核加入的一种虚拟总线，其优势在于采用了总线的模型对设备与驱动进行了管理，这样提高了程序的可移植性。

前面我们学习了总线的创建初始化等一系列总线的开发，在我们以后的开发中，我们只要基于这个虚拟总线开发设备，开发驱动即可。平台总线的描述结构是platform\_bus\_type结构，其中比较重要的有name成员，值为平台名称，还有match成员，指向一个总线匹配函数，该函数用来匹配设备和驱动的，因此我们来看看这个平台匹配函数到底是怎么来匹配设备和驱动的，我们进入平台总线匹配函数实现处，根据里面的两个条件判断语句得知平台匹配函数是通过id\_table以及设备名和驱动名来匹配的，其中主要还是以设备名和驱动名来匹配的。

平台设备的描述结构为platform\_device结构，其中比较重要的成员由\*name，是指设备名，另一个成员是resource成员，指向一个设备资源。其中资源有基地址、中断号等都是属于设备的资源。该资源的结构为resource结构，其中比较重要的成员有start，end，flags，分别表示起始地址，结束地址，以及资源的类型。资源类型以IORESOURCE关键字开头Ctrl-n可查找到其他关键字。

我们基于平台总线机制开发设备驱动的主要流程就是，首先定义和注册platform\_device，其次就是定义和注册platform\_driver，最后就是分别卸载驱动和设备，但我们这里，一般只卸载设备，驱动可以保留在平台总线上。

**平台总线匹配函数匹配规则**

static int platform\_match**(**struct device **\***dev**,** struct device\_driver **\***drv**)**

**{**

struct platform\_device **\***pdev **=** to\_platform\_device**(**dev**);**

struct platform\_driver **\***pdrv **=** to\_platform\_driver**(**drv**);**

/\* match against the id table first \*/

**if** **(**pdrv**->**id\_table**)** //but most devices without id\_table

**return** platform\_match\_id**(**pdrv**->**id\_table**,** pdev**)** **!=** **NULL;**

/\* fall-back to driver name match \*/

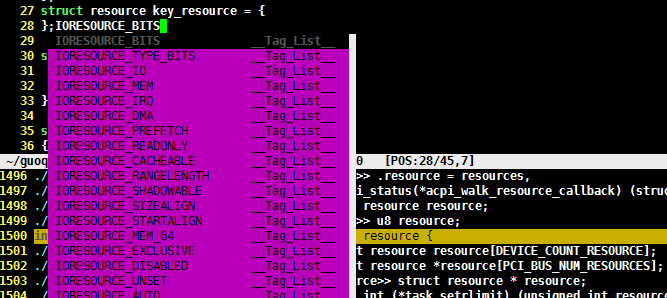
**return** **(**strcmp**(**pdev**->**name**,** drv**->**name**)** **==** 0**);**

**}**

平台设备描述结构platform\_device中比较重要的成员是\*name, id, \*resource，

资源resource结构中比较中的成员就是start, end, flags,

其中flags我们可以通过开始关键字索引来查找其它，如下图所示：



**平台设备注册与注销**

**int platform\_device\_register (struct platform\_device \*pdev);**

**void platform\_device\_unregister (struct platform\_device \*pdev);**

**EXPORT\_SYMBOL\_GPL (platform\_device\_unregister);**

(/sys/bus/platform/device/...)

平台驱动描述结构platform\_driver中比较重要的成员是(\*probe)和(\*remove)这两个函数指针。

**平台驱动描述结构**

struct platform\_driver {

**int (\*probe)(struct platform\_device \*);**

**int (\*remove)(struct platform\_device \*);**

void (\*shutdown)(struct platform\_device \*);

int (\*suspend)(struct platform\_device \*, pm\_message\_t state);

int (\*resume)(struct platform\_device \*);

**struct device\_driver driver;**

struct platform\_device\_id \*id\_table;

};

**平台驱动注册与注销**

**int platform\_driver\_register (struct platform\_driver \*pdrv);**

**EXPORT\_SYMBOL\_GPL(platform\_driver\_register);**

**void platform\_driver\_unregister (struct platform\_driver \*pdrv);**

**EXPORT\_SYMBOL\_GPL(platform\_driver\_unregister);**

通常为了提高驱动的可移植性，驱动和设备是分开的，我们以平台总线为例，所以我们编写设备模块和驱动模块的时候就需要分开写，而且驱动中不能包含硬件信息的直接获取，而是从设备模块中获取硬件相关信息，相当于说，设备模块的主要工作就是平台设备的注册和注销，也就是该设备的各种硬件资源信息的定义。

下面是平台设备模块的范例代码：

#include <linux/init.h>

#include <linux/module.h>

#include <linux/interrupt.h>

#include <linux/platform\_device.h>

MODULE\_LICENSE**(**"GPL"**);**

#define GPGCON 0x56000060 //for mini2440

struct resource key\_resource**[]** **=** **{**

**[**0**]** **=** **{**

**.**start **=** GPGCON**,**

**.**end **=** GPGCON**+**8**,**

**.**flags **=** IORESOURCE\_MEM**,**

**},**

**[**1**]** **=** **{**

**.**start **=** IRQ\_EINT0**,**

**.**end **=** IRQ\_EINT11**,**

**.**flags **=** IORESOURCE\_IRQ**,**

**},**

**};**

struct platform\_device key\_device **=** **{**

**.**name **=** "my-key"**,**

**.**id **=** 0**,**

**.**num\_resources **=** ARRAY\_SIZE**(**key\_resource**),**

**.**resource **=** key\_resource**,**

**};**

static int key\_init **(**void**)**

**{**

platform\_device\_register**(&**key\_device**);**

**return** 0**;**

**}**

static void key\_exit **(**void**)**

**{**

platform\_device\_unregister**(&**key\_device**);**

**return** **;**

**}**

module\_init **(**key\_init**);**

module\_exit **(**key\_exit**);**

而平台驱动模块主要就是围绕平台驱动的注册和注销来工作的，所以我们编写的时候基本也是围绕平台驱动的注册和注销来逐步深入编写的，逐个实现各个功能函数。所以我们之前的混杂设备注册、注册中断、硬件初始化、阻塞功能添加等都在平台驱动中完成。

下面是平台驱动的范例代码：

#include <linux/init.h>

#include <linux/module.h>

#include <linux/miscdevice.h>

#include <linux/interrupt.h>

#include <linux/io.h>

#include <linux/fs.h>

#include <asm/uaccess.h>

#include <linux/platform\_device.h>

#include <linux/workqueue.h>

#include <linux/timer.h>

#include <linux/device.h>

#include <linux/wait.h>

#include <linux/sched.h>

#include <linux/kernel.h>

#include <linux/ioport.h>

#include <linux/pci.h>

MODULE\_LICENSE**(**"GPL"**);**

struct work\_struct **\***work**;**

struct timer\_list key\_timer**;**

unsigned int key\_num **=** 0**;**

wait\_queue\_head\_t key\_queue**;**

struct resource **\***res\_mem**;**

struct resource **\***res\_irq**;**

unsigned int **\***key\_base**;**

struct class **\***my\_class**;**

void work\_func **(**struct work\_struct **\***work**)**

**{**

mod\_timer**(&**key\_timer**,** jiffies**+(**HZ**/**10**));**

**return** **;**

**}**

void key\_timer\_func**(**unsigned long data**)**

**{**

unsigned int key\_value**;**

//gpio\_data = ioremap(GPGDAT, 4);

key\_value **=** readw **(**gpio\_data**)&**0x01**;** //0001

**if** **(**key\_value **==** 0**)**

**{**

key\_number **=** 1**;**

printk**(**"Key1 down!\n"**);**

**}**

key\_value **=** readw **(**gpio\_data**)&**0x04**;** //0100

**if** **(**key\_value **==** 0**)**

**{**

key\_number **=** 2**;**

printk**(**"Key2 down!\n"**);**

**}**

wake\_up **(&**key\_queue**);** //wake up queue if key\_number be value

**return** **;**

**}**

static void key\_hw\_init **(**void**)**

**{**

//gpio\_config = (int \*)ioremap(GPGCON, 4);

data **=** readw**(**key\_base**);**

data **&=** **~(**0x3**);**

data **|=** 0x2**;** //K1:GPG0:[1:0]

data **|=** **(**0x2**<<**6**);** //K2:GPG3:[7:6]

writew **(**data**,** key\_base**);**

**return** **;**

**}**

irqreturn\_t key\_int**(**int irq**,** void **\***dev\_id**)**

**{**

//01.check whether irq or not for the device

printk**(**"check whether irq or not for the device\n"**);**

//02.clear irq issued

printk**(**"clear irq issued\n"**);**

//hand on work

schedule\_work**(**work**);**

**return** IRQ\_HANDLED**;**

**}**

int key\_open **(**struct inode **\***node**,** struct file **\***filp**)**

**{**

**return** 0**;**

**}**

ssize\_t key\_read **(**struct file **\***filp**,** char \_\_user **\***buf**,** size\_t size**,** loff\_t **\***pos**)**

**{**

//if condition be true, continue, or not be waited

//following:if key\_number!=0, go to execute following code

wait\_event**(**key\_queue**,** key\_num**!=**0**);**

copy\_to\_user**(**buf**,** **&**key\_num**,** 4**);**

key\_num **=** 0**;**

**return** 0**;**

**}**

struct file\_operations key\_fops **=** **{**

**.**open **=** key\_open**,**

**.**read **=** key\_read**,**

**};**

struct miscdevice key\_miscdev **=** **{**

**.**minor **=** 200**,**

**.**name **=** "key"**,**

**.**fops **=** **&**key\_fops**,**

**};**

接着来实现key\_probe函数，只有设备挂上去了，平台总线检测到了，才会调用驱动模块里面的probe函数，在这里面进行各种初始化：

int key\_probe **(**struct platform\_device **\***pdev**)**

**{**

int size**,** ret**;**

my\_class **=** class\_create **(**THIS\_MODULE**,** "my-key"**);**

device\_create **(**my\_class**,** **NULL,** MKDEV**(**10**,** 200**),** **NULL,** "my-key%d"**,** 200**);**

//add miscdevice

ret **=** misc\_register **(&**key\_miscdev**);**

//EINT8(K1), EINT11(K2) (for mini2440)

res\_irq **=** platform\_get\_resource**(**pdev**,** IORESOURCE\_IRQ**,** 1**);**

request\_irq **(**res\_irq**->**start**,** key\_int**,** IRQF\_TRIGGER\_FALLING**,** "my-key1"**,** **(**void **\*)**1**);**

request\_irq **(**res\_irq**->**end**,** key\_int**,** IRQF\_TRIGGER\_FALLING**,** "my-key2"**,** **(**void **\*)**2**);**

//device hardware initialization

res\_mem **=** platform\_get\_resource**(**pdev**,** IORESOURCE\_MEM**,** 0**);**

size **=** **(**res\_mem**->**end **-** res\_mem**->**start**)** **+** 1**;**

key\_base **=** ioremap **(**res\_mem**->**start**,** size**);**

key\_hw\_init**();**

//create a work

work **=** kmalloc**(sizeof(**struct work\_struct**),** GFP\_KERNEL**);**

INIT\_WORK**(**work**,** work\_func**);**

//init timer

init\_timer **(&**key\_timer**);**

key\_timer**.**function **=** key\_timer\_func**;**

//register timer

add\_timer **(&**key\_timer**);**

//initialization wait queue

init\_waitqueue\_head **(&**key\_queue**);**

**return** ret**;**

**}**

然后就是实现key\_driver里面的key\_remove函数：

int key\_remove **(**struct platform\_device **\***dev**)**

**{**

free\_irq **(**res\_irq**->**start**,** **(**void **\*)**4**);**

free\_irq **(**res\_irq**->**end**,** **(**void **\*)**3**);**

iounmap **(**key\_base**);**

misc\_deregister **(&**key\_miscdev**);**

**return** 0**;**

**}**

然后完善平台驱动注册的那个key\_driver参数：（参考内核相关代码）

struct platform\_driver key\_driver **=** **{**

**.**probe **=** key\_probe**,**

**.**remove **=** key\_remove**,**

**.**driver **=** **{**

**.**owner **=** THIS\_MODULE**,**

**.**name **=** "my-key"**,**

**},**

**};**

注意上面的name必须和设备模块里面的name相同，否则无法匹配成功。

首先我们搭建好平台驱动的注册和注销工作：

static int key\_init **(**void**)**

**{**

platform\_driver\_register**(&**key\_driver**);**

**return** 0**;**

**}**

static void key\_exit **(**void**)**

**{**

platform\_driver\_unregister**(&**key\_driver**);**

**return** **;**

**}**

module\_init **(**key\_init**);**

module\_exit **(**key\_exit**);**

# 专题03-串口驱动程序设计

[主目录](#_目录)

## 串口驱动分析：tty驱动架构

tty概念解析

在Linux系统中，终端是一类字符型设备，它包括多种类型，通常用tty来简称各种类型的终端设备。

串口终端（/dev/ttyS\*）

串口终端是使用计算机串口连接的终端设备。Linux把每个串行端口都看作是一个字符设备。这些串行端口所对应的设备名称是/dev/ttySAC0,ttySAC1,...

控制台终端（/dev/console）

在Linux系统中，计算机的输出设备通常被称为控制台终端，这里特指printk信息输出到的设备。/dev/console是一个虚拟的设备，它需要映射到真正的tty上，比如通过内核启动参数console=ttySAC0就把console映射到串口0上了。

虚拟终端（/dev/tty\*）

当用户登录时，使用的是虚拟终端。使用Ctrl-Alt-[F1~F6]组合键时，我们就可以切换到tty1,tty2,tty3等上面去。这几个称为虚拟终端，而tty0则是当前使用虚拟终端的一个别名。一般表示当前虚拟终端。

串口终端、控制台终端、虚拟终端这三者有什么关系呢？

由我们的串口终端引出了我们的控制台终端，我们的Linux内核通过printk()函数把所有信息输出到我们的控制台终端，然后我们控制台终端和串口终端关联，所以这些信息就全部到了串口终端上面。所以控制台终端是专门给内核使用的。

（串口终端是控制台终端和虚拟终端最终发送的目的地）

然而对应我们的应用程序，使用虚拟终端，这些虚拟终端同样是和串口关联起来的，而虚拟终端更多的是给我们应用程序来使用的。比如应用程序把信息输出到tty3上或tty4上，然后这些虚拟终端最终全部都要把信息输出到硬件上面也就是串口终端上面。

tty架构分析

TTY子系统主要是由3部分构成，从用户空间到硬件至上而下，分别是tty核心、tty线路规程(Line discipline)、tty驱动，其中tty核心对用户空间提供统一接口；tty线路规程属于这个子系统中间部分，那么它可以对驱动发送上来的数据或用户空间发送下来的数据进行相应格式化，比如说我们要基于这个tty驱动去实现某一种协议，那么就可以把这个协议的实现放到tty线路规程里面来，比如说驱动传上来的数据我们就按照这个协议对它进行格式化，那么下来的数据我们也可以按照这个协议对它进行格式化，这就是我们tty线路规程的作用；最后我们来看看tty驱动，tty驱动是和我们实际硬件密切相关的，说的更直白点它就是控制我们实际硬件的，比如说我们现在这个硬件是串口，那么我们这个驱动就是串口驱动程序，所以说我们这个串口驱动实际上是位于我们这个tty驱动部分的，我们的串口驱动实际上就是tty驱动。

为什么我们首先要来介绍tty架构？

原因很简单，因为我们接下来要学习的这个串口驱动是作为我们这个tty模型当中的一个组成部分，那么既然串口驱动是这个模型当中的一员，你必然会和这个模型的其它部分有一定的交互关系，比如说和我们这个tty核心会有一定的交互关系，比如说和我们这个tty线路规程会有一定的交互关系，如果说我们不把这个关系理清楚，那么我们很难写出一个正确的Linux下面的这个串口驱动程序。

当我们的tty设备被读或被写的时候的函数调用关系

比如说现在我们的硬件层是串口，现在我要对这个串口进行写入操作或者说我要对串口进行读出的操作，那么会经历什么过程。

什么叫做回溯呢？

所谓回溯，是指你知道最底层的这个函数，然后向上一步一步去把它的调用关系给找出来，那么这个就叫做回溯。

我们通过回溯来揭秘串口发送流程

当我们向串口发送数据时候，也就是我们希望串口输出数据的时候，那么我们串口最底层的函数是什么呢，是串口驱动里面的发送函数，所以说我们要去找到这个串口驱动里面的发送函数。那么我们首先打开内核代码（drivers/serial/），那么这一部分代码实际上就是我们的串口驱动代码了，我们在这个目录打开samsung.c文件，我们就从这个文件里面去找串口驱动当中最底层的这个发送函数(\_serial\_start\_tx)，那么我现在知道了这个最底层的发送函数(s3c24xx\_serial\_start\_tx())，我们怎么去把它上面的调用关系给理出来呢？这里我们用一个最简单最方便的办法，就是在这个最底层函数的开始处添加一个函数(dump\_stack();)，然后我们编译这个内核，把编译好的内核弄到开发板并启动开发板，启动到根文件系统位置串口终端就会打印出一大堆回溯信息，从最上面的dump\_backtrace,dump\_stack标签开始往下，依次是层层被调用关系。下面我就来通过回溯来通过内核代码查阅方式来看看串口数据发送流程。

下面这个函数是串口驱动中最底层的函数中添加回溯函数：

static void s3c24xx\_serial\_start\_tx(struct uart\_port \*port)

{

struct s3c24xx\_uart\_port \*ourport = to\_ourport(port);

dump\_stack();

if (!tx\_enabled(port)) {

if (port->flags & UPF\_CONS\_FLOW)

s3c24xx\_serial\_rx\_disable(port);

enable\_irq(ourport->tx\_irq);

tx\_enabled(port) = 1;

}

}

以下是串口终端打印的回溯信息范例：

dump\_backtrace()

dump\_stack()

s3c24xx\_serial\_start\_tx()

uart\_start()

uart\_write()

n\_tty\_write()

tty\_write()

redirected\_tty\_write()

vfs\_write()

sys\_write()

我们根据回溯信息从上层redirected\_tty\_write()开始往下跟踪：

redirected\_tty\_write()

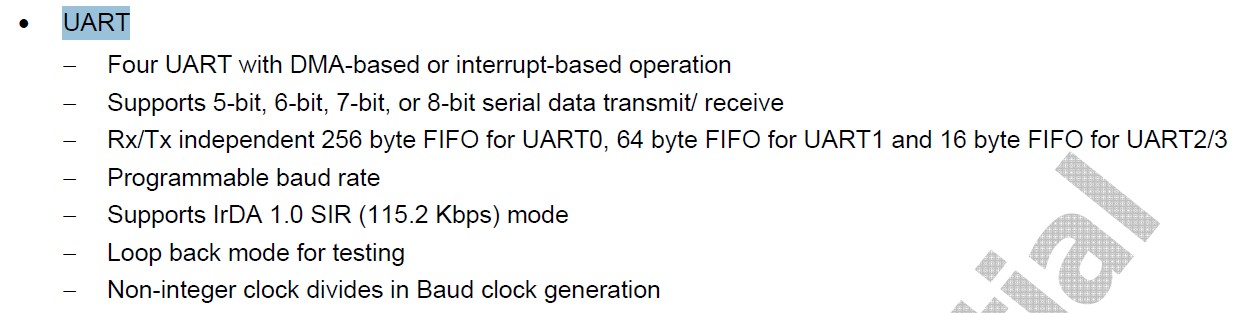
tty\_write()

do\_tty\_write(ld->ops->write,......)

可以通过搜索n\_tty\_write被复制给write的位置可知它被赋值给一个write指针。

## [ARM-Linux S5PV210 UART驱动（1）----用户手册中的硬件知识](http://www.cnblogs.com/hello2mhb/p/3340611.html)

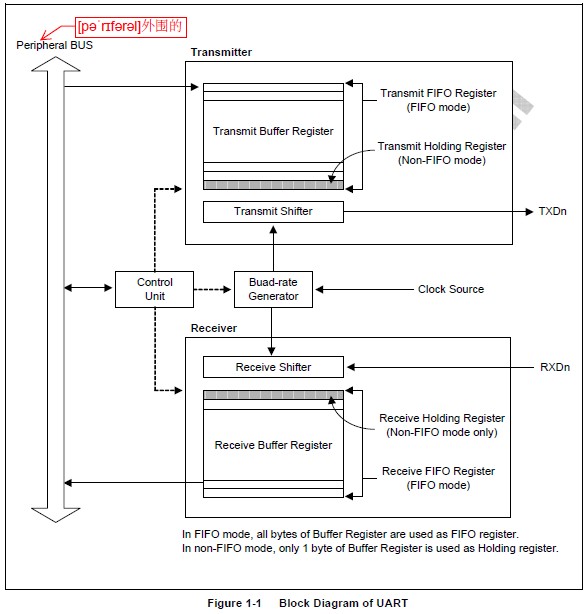
一、概述



The Universal Asynchronous Receiver and Transmitter (UART) in S5PV210 provide four independent asynchronous, and serial input/output (I/O) ports. All the ports operate in an interrupt-based or a DMA-based mode. The UART generates an interrupt or a DMA request to transfer data to and from the CPU and the UART. The UART supports bit rates up to 3Mbps. Each UART channel contains two FIFOs to receive and transmit data: 256 bytes in ch0, 64 bytes in ch1 and 16 bytes in ch2 and ch3.

UART includes programmable baud rates, infrared (IR) transmitter/receiver, one or two stop bit insertion, 5-bit, 6-bit, 7-bit, or 8-bit data width and parity checking.

Each UART contains a baud-rate generator, a transmitter, a receiver and a control unit, as shown in Figure 1-1. The baud-rate generator uses PCLK or SCLK\_UART. The transmitter and the receiver contain FIFOs and data shifters. The data to be transmitted is written to Tx FIFO, and copied to the transmit shifter. The data is then shifted out by the transmit data pin (TxDn). The received data is shifted from the receive data pin (RxDn), and copied to Rx FIFO from the shifter.



二、寄存器

## [ARM-Linux S5PV210 UART驱动（2）---- 终端设备驱动](http://www.cnblogs.com/hello2mhb/p/3333474.html)

在Linux中，UART串口驱动完全遵循tty驱动的框架结构，但是进行了底层操作的再次封装，所以先介绍tty终端设备驱动。

一、终端设备

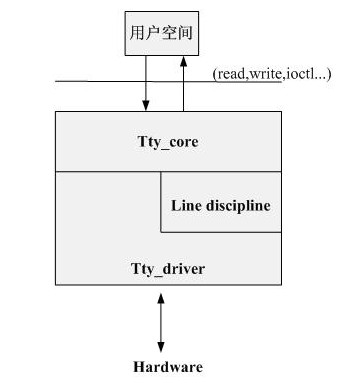
　　1.串行端口终端（/dev/ttySACn）

　　2.伪终端（/dev/pty/）

　　3.控制台终端（/dev/ttyn,/dev/console）

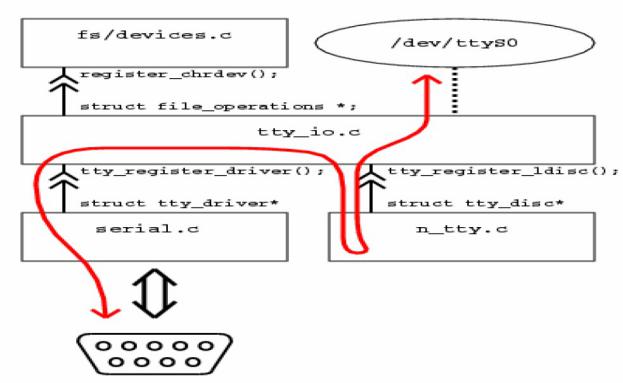
二、驱动结构

　　1.tty分层结构如下图所示：



包含tty核心、tty线路规程、tty驱动，其中tty规程的工作是以特殊的方式格式化从一个用户或者硬件接收到的数据，常采用一个协议转换的形式，如PPP、Bluetooth。

　　2.tty主要源文件关系及数据流向如下图：



**特定的tty设备驱动的主体工作是填充tty\_driver结构体中的成员，实现tty\_operations结构体中的一系列成员函数**

[复制代码](javascript:void(0);)

struct tty\_driver {

int magic; /\* magic number for this structure \*/

struct kref kref; /\* Reference management \*/

struct cdev cdev;

struct module \*owner;

constchar \*driver\_name;//驱动名字

constchar \*name;//驱动设备结点名字

int name\_base; /\* offset of printed name \*/

int major; /\* major device number \*/

int minor\_start; /\* start of minor device number \*/

int minor\_num; /\* number of \*possible\* devices \*/

int num; /\* number of devices allocated \*/

short type; /\* type of tty driver \*/

short subtype; /\* subtype of tty driver \*/

struct ktermios init\_termios; /\* Initial termios \*/

int flags; /\* tty driver flags \*/

struct proc\_dir\_entry \*proc\_entry; /\* /proc fs entry \*/

struct tty\_driver \*other; /\* only used for the PTY driver \*/

/\*

\* Pointer to the tty data structures

\*/

struct tty\_struct \*\*ttys;

struct ktermios \*\*termios;

struct ktermios \*\*termios\_locked;

void \*driver\_state;

/\*

\* Driver methods

\*/

conststruct tty\_operations \*ops;

struct list\_head tty\_drivers;

};

[复制代码](javascript:void(0);)

tty\_operations中的成员函数需在特定设备tty驱动模块初始化函数中被赋值；

 tty\_struct结构体被tty核心用来保存当前tty端口的状态。

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif struct tty\_operations

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif struct tty\_struct

**tty\_io.c：**

【1】定义了tty 设备通用的file\_operations结构体：

[复制代码](javascript:void(0);)

staticconststruct file\_operations tty\_fops = {

.llseek = no\_llseek,

.read = tty\_read,

.write = tty\_write,

.poll = tty\_poll,

.unlocked\_ioctl = tty\_ioctl,

.compat\_ioctl = tty\_compat\_ioctl,

.open = tty\_open,

.release = tty\_release,

.fasync = tty\_fasync,

};

[复制代码](javascript:void(0);)

【2】实现了接口函数alloc\_tty\_driver()用于分配tty驱动：

[复制代码](javascript:void(0);)

/\*tty driver 的所有操作都包含在 tty\_driver 中。内核即供了一个名叫 alloc\_tty\_driver()

来分配这个 tty\_driver。当然我们也可以在自己的驱动中将它定义成一个静态的结构。对

tty\_driver 进行一些必要的初始化之后，调用 tty\_register\_driver()将其注册. \*/

struct tty\_driver \*alloc\_tty\_driver(int lines)

{

struct tty\_driver \*driver;

driver = kzalloc(sizeof(struct tty\_driver), GFP\_KERNEL);

if (driver) {

kref\_init(&driver->kref);

driver->magic = TTY\_DRIVER\_MAGIC;

driver->num = lines;

/\* later we'll move allocation of tables here \*/

}

return driver;

}

[复制代码](javascript:void(0);)

【3】实现了接口函数tty\_set\_operations()用于设置tty驱动操作：

[复制代码](javascript:void(0);)

/\*将tty\_operations结构体中的函数指针拷贝给tty\_driver对应的函数指针\*/

void tty\_set\_operations(struct tty\_driver \*driver,

conststruct tty\_operations \*op)

{

driver->ops = op;

};

[复制代码](javascript:void(0);)

【4】实现了接口函数tty\_register\_driver()用于注册tty设备：

[复制代码](javascript:void(0);)

/\*

\* Called by a tty driver to register itself.

\* 注册 tty 驱动成功时返回 0；参数为由 alloc\_tty\_driver()分配的 tty\_driver 结构体指针。

\*这个函数操作比较简单。就是为 tty\_driver 创建字符设备。然后将字符设备的操作集指定

\*为 tty\_fops.并且将 tty\_driver 挂载到 tty\_drivers 链表中。以设备号为关键字找到对应

\*的 driver.

\*/

int tty\_register\_driver(struct tty\_driver \*driver)

{

int error;

int i;

dev\_t dev;

void \*\*p = NULL;

/\*TTY\_DRIVER\_DEVPTS\_MEM:使用 devpts 进行动态内存映射\*/

if (!(driver->flags & TTY\_DRIVER\_DEVPTS\_MEM) && driver->num) {

p = kzalloc(driver->num \* 2 \* sizeof(void \*), GFP\_KERNEL);

if (!p)

return -ENOMEM;

}

if (!driver->major) {/\*如果没有指定 driver->major,动态申请字符设备号 \*/

error = alloc\_chrdev\_region(&dev, driver->minor\_start,

driver->num, driver->name);

if (!error) {

driver->major = MAJOR(dev);

driver->minor\_start = MINOR(dev);

}

} else {/\*否则根据主设备号申请字符设备号\*/

dev = MKDEV(driver->major, driver->minor\_start);

error = register\_chrdev\_region(dev, driver->num, driver->name);

}

if (error <0) {

kfree(p);/\*如果失败则释放申请的内存空间\*/

return error;

}

if (p) {/\*设置tty数据结构指针\*/

driver->ttys = (struct tty\_struct \*\*)p;/\*tty\_struct结构体被tty核心用来保存当前端口的状态\*/

driver->termios = (struct ktermios \*\*)(p + driver->num);/\*ktermios保存当前的线路设置\*/

} else {

driver->ttys = NULL;

driver->termios = NULL;

}

//注册字符设备

cdev\_init(&driver->cdev, &tty\_fops);

driver->cdev.owner = driver->owner;

error = cdev\_add(&driver->cdev, dev, driver->num);

if (error) {

unregister\_chrdev\_region(dev, driver->num);

driver->ttys = NULL;

driver->termios = NULL;

kfree(p);

return error;

}

mutex\_lock(&tty\_mutex);

list\_add(&driver->tty\_drivers, &tty\_drivers);/\*将 tty\_driver 挂载到 tty\_drivers 链表中\*/

mutex\_unlock(&tty\_mutex);

/\*如果没有指定 TTY\_DRIVER\_DYNAMIC\_DEV.即动态设备管理 \*/

if (!(driver->flags & TTY\_DRIVER\_DYNAMIC\_DEV)) {

for (i = 0; i < driver->num; i++)

tty\_register\_device(driver, i, NULL);/\*注册tty设备\*/

}

/\*

\* This function is called by tty\_register\_driver() to handle

\* registering the driver's /proc handler into /proc/tty/driver/<foo>

\*/

proc\_tty\_register\_driver(driver);

driver->flags |= TTY\_DRIVER\_INSTALLED;

return0;

}

## [ARM-Linux S5PV210 UART驱动（3）----串口核心层、关键结构体、接口关系](http://www.cnblogs.com/hello2mhb/p/3341291.html)

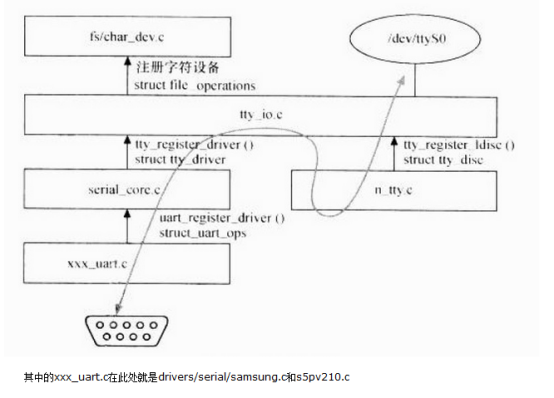
尽管一个特定的UART设备驱动完全可以按照tty驱动的设计方法来设计，即定义tty\_driver并实现tty\_operations其中的成员函数，但是Linux已经在文件serial\_core.c中实现了UART设备的通用tty驱动层，称为串口核心层，这样，UART驱动的主要任务变成了实现serial\_core.c中定义的一组uart\_xxx接口而非tty\_xxx接口。

　　uart设备是继tty\_driver的又一层封装.实际上uart\_driver就是对应tty\_driver.在它的操作函数中,将操作转入uart\_port.在写操作的时候,先将数据放入一个叫做circ\_buf的环形缓存区.然后uart\_port从缓存区中取数据,将其写入到串口设备中.当uart\_port从serial设备接收到数据时,会将设备放入对应line discipline的缓存区中.这样.用户在编写串口驱动的时候,只先要注册一个uart\_driver.它的主要作用是定义设备节点号.然后将对设备的各项操作封装在uart\_port.驱动工程师没必要关心上层的流程,只需按硬件规范将uart\_port中的接口函数完成就可以了.

1.下图描述了串行系统间的层次结构关系,可以概括为：用户应用层 --> 线路规划层 --> TTY层 --> 底层驱动层 --> 物理硬件层

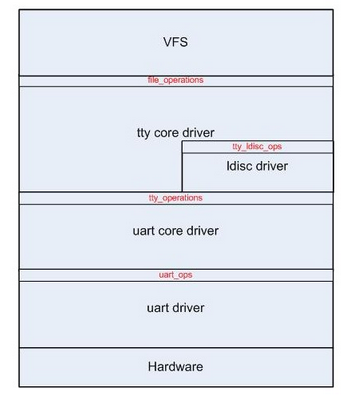


2.下图是串口核心层在整个tty源文件关系及数据流向中的位置：



其中的xxx\_uart.c在此处就是drivers/serial/samsung.c和s5pv210.c

3.接口关系：

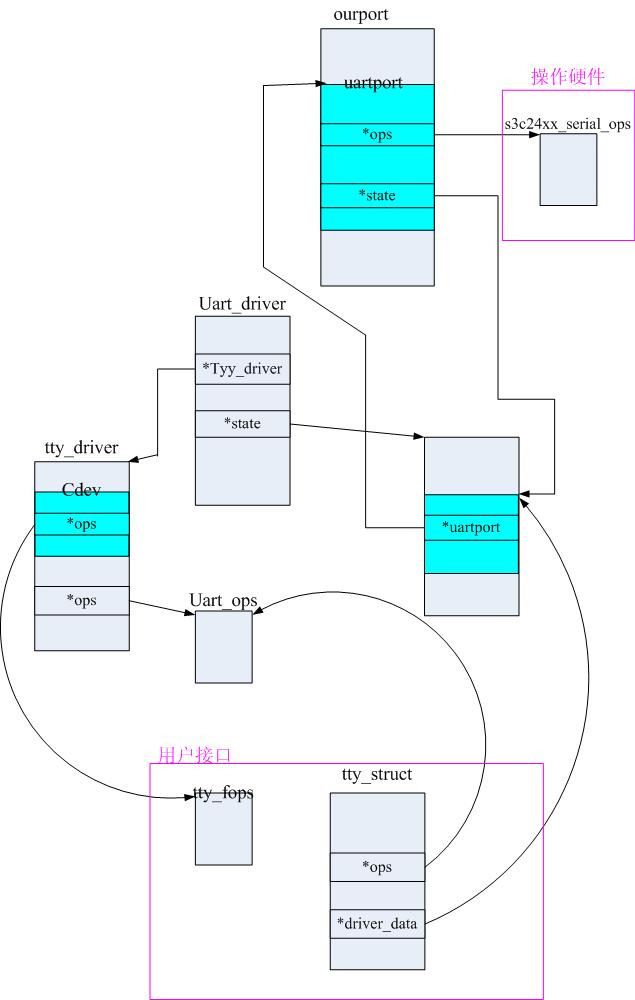


从接口关系图可以看出，用户对uart设备操作的调用关系非常简单，

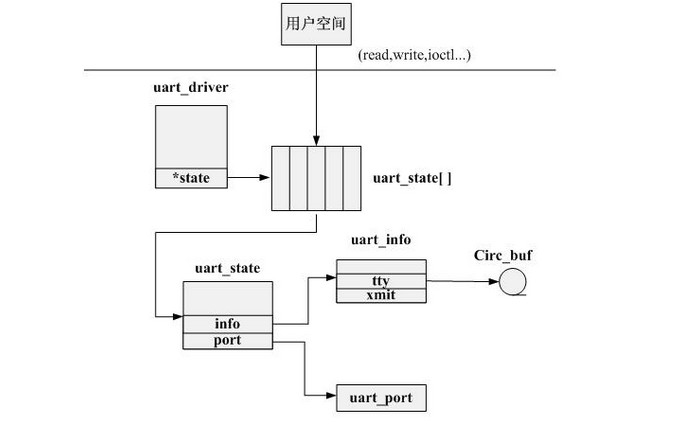
file\_operations => [tty\_ldisc\_ops] => tty\_operations => uart ops

其中tty\_ldisc\_ops线路规程并不是必要的，依赖于应用层设置是否使用ldisc处理数据。

4.UART驱动的总图：



 5.uart驱动常用的数据结构表示如下：



6.Uart驱动程序主要围绕三个关键的数据结构展开(include/linux/serial\_core.h中定义)：

**UART特定的驱动程序结构定义：struct uart\_driver s3c24xx\_uart\_drv;**

**UART端口结构定义： struct s3c24xx\_uart\_port s3c24xx\_serial\_ports；**

**UART相关操作函数结构定义: struct uart\_ops s3c24xx\_serial\_ops;**

【1】uart\_driver 封装了tty\_driver，使得底层的UART驱动无需关心tty\_driver

[复制代码](javascript:void(0);)

struct uart\_driver {

struct module \*owner;

constchar \*driver\_name;

constchar \*dev\_name;

int major;

int minor;

int nr;

struct console \*cons;

/\*

\* these are private; the low level driver should not

\* touch these; they should be initialised to NULL

\*/

structuart\_state \*state;

structtty\_driver \*tty\_driver;

};

其中的uart\_state是设备私有信息结构体，

 在uart\_open()中:

 tty->driver\_data = state;

 在其他uart\_xxx()中：

 struct uart\_state \*state = tty->driver\_data;

就可以获取设备私有信息结构体。

[复制代码](javascript:void(0);)

staticstruct uart\_driver **s3c24xx\_uart\_drv**= {

.owner =THIS\_MODULE,

.dev\_name = "s3c2440\_serial", //具体设备名称

.nr =CONFIG\_SERIAL\_SAMSUNG\_UARTS, //定义有几个端口

.cons = S3C24XX\_SERIAL\_CONSOLE, //console接口

.driver\_name =S3C24XX\_SERIAL\_NAME, //串口名：ttySAC

.major =S3C24XX\_SERIAL\_MAJOR, //主设备号

.minor =S3C24XX\_SERIAL\_MINOR, //次设备号

};

一个tty驱动必须注册/注销tty\_driver，而一个UART驱动则变为注册/注销uart\_driver，使用如下接口：

int uart\_register\_driver(struct uart\_driver \*drv)；

void uart\_unregister\_driver(struct uart\_driver \*drv)；

【2】uart\_port用于描述一个UART端口（直接对应于一个串口）的I/O端口或者IO内存地址等信息。

[复制代码](javascript:void(0);)

struct uart\_port {

spinlock\_t lock; /\* port lock \*/

unsigned long iobase; /\* in/out[bwl] \*/

unsigned char \_\_iomem \*membase; /\* read/write[bwl] \*/

unsigned int (\*serial\_in)(struct uart\_port \*, int);

void (\*serial\_out)(struct uart\_port \*, int, int);

unsigned int irq; /\* irq number \*/

unsigned long irqflags; /\* irq flags \*/

unsigned int uartclk; /\* base uart clock \*/

unsigned int fifosize; /\* tx fifo size \*/

unsigned char x\_char; /\* xon/xoff char \*/

unsigned char regshift; /\* reg offset shift \*/

unsigned char iotype; /\* io access style \*/

unsigned char unused1;

#define UPIO\_PORT (0)

#define UPIO\_HUB6 (1)

#define UPIO\_MEM (2)

#define UPIO\_MEM32 (3)

#define UPIO\_AU (4) /\* Au1x00 type IO \*/

#define UPIO\_TSI (5) /\* Tsi108/109 type IO \*/

#define UPIO\_DWAPB (6) /\* DesignWare APB UART \*/

#define UPIO\_RM9000 (7) /\* RM9000 type IO \*/

unsigned int read\_status\_mask; /\* driver specific \*/

unsigned int ignore\_status\_mask; /\* driver specific \*/

struct uart\_state \*state; /\* pointer to parent state \*/

struct uart\_icount icount; /\* statistics \*/

struct console \*cons; /\* struct console, if any \*/

#if defined(CONFIG\_SERIAL\_CORE\_CONSOLE) || defined(SUPPORT\_SYSRQ)

unsigned long sysrq; /\* sysrq timeout \*/

#endif

upf\_t flags;

#define UPF\_FOURPORT ((\_\_force upf\_t) (1 << 1))

#define UPF\_SAK ((\_\_force upf\_t) (1 << 2))

#define UPF\_SPD\_MASK ((\_\_force upf\_t) (0x1030))

#define UPF\_SPD\_HI ((\_\_force upf\_t) (0x0010))

#define UPF\_SPD\_VHI ((\_\_force upf\_t) (0x0020))

#define UPF\_SPD\_CUST ((\_\_force upf\_t) (0x0030))

#define UPF\_SPD\_SHI ((\_\_force upf\_t) (0x1000))

#define UPF\_SPD\_WARP ((\_\_force upf\_t) (0x1010))

#define UPF\_SKIP\_TEST ((\_\_force upf\_t) (1 << 6))

#define UPF\_AUTO\_IRQ ((\_\_force upf\_t) (1 << 7))

#define UPF\_HARDPPS\_CD ((\_\_force upf\_t) (1 << 11))

#define UPF\_LOW\_LATENCY ((\_\_force upf\_t) (1 << 13))

#define UPF\_BUGGY\_UART ((\_\_force upf\_t) (1 << 14))

#define UPF\_NO\_TXEN\_TEST ((\_\_force upf\_t) (1 << 15))

#define UPF\_MAGIC\_MULTIPLIER ((\_\_force upf\_t) (1 << 16))

#define UPF\_CONS\_FLOW ((\_\_force upf\_t) (1 << 23))

#define UPF\_SHARE\_IRQ ((\_\_force upf\_t) (1 << 24))

/\* The exact UART type is known and should not be probed. \*/

#define UPF\_FIXED\_TYPE ((\_\_force upf\_t) (1 << 27))

#define UPF\_BOOT\_AUTOCONF ((\_\_force upf\_t) (1 << 28))

#define UPF\_FIXED\_PORT ((\_\_force upf\_t) (1 << 29))

#define UPF\_DEAD ((\_\_force upf\_t) (1 << 30))

#define UPF\_IOREMAP ((\_\_force upf\_t) (1 << 31))

#define UPF\_CHANGE\_MASK ((\_\_force upf\_t) (0x17fff))

#define UPF\_USR\_MASK ((\_\_force upf\_t) (UPF\_SPD\_MASK|UPF\_LOW\_LATENCY))

unsigned int mctrl; /\* current modem ctrl settings \*/

unsigned int timeout; /\* character-based timeout \*/

unsigned int type; /\* port type \*/

conststruct uart\_ops \*ops;//UART操作集------->

unsigned int custom\_divisor;

unsigned int line; /\* port index \*/

resource\_size\_t mapbase; /\* for ioremap \*/

struct device \*dev; /\* parent device \*/

unsigned char hub6; /\* this should be in the 8250 driver \*/

unsigned char suspended;

unsigned char unused[2];

void \*private\_data; /\* generic platform data pointer \*/

};

s3c24xx\_uart\_port 封装了uart\_port：

[复制代码](javascript:void(0);)

struct s3c24xx\_uart\_port {

unsigned char rx\_claimed;

unsigned char tx\_claimed;

unsigned int pm\_level;

unsigned long baudclk\_rate;

unsigned int rx\_irq;

unsigned int tx\_irq;

structs3c24xx\_uart\_info \*info;

structs3c24xx\_uart\_clksrc \*clksrc;

struct clk \*clk;

struct clk \*baudclk;

struct uart\_port port;

#ifdef CONFIG\_CPU\_FREQ

struct notifier\_block freq\_transition;

#endif

int channelnum;

};

staticstruct s3c24xx\_uart\_port

**s3c24xx\_serial\_ports**[CONFIG\_SERIAL\_SAMSUNG\_UARTS] = {

[0] = {//串口0；

。。。。。。

在xxx\_probe()中：

struct s3c24xx\_uart\_port \*ourport;//s3c24xx\_uart\_port封装了uart\_port

ourport = &s3c24xx\_serial\_ports[dev->id];//s3c24xx\_serial\_ports是s3c24xx\_uart\_port结构体类型的

【3】uart\_ops定义了针对UART的一系列操作，

struct uart\_ops {

。。。。。。

[复制代码](javascript:void(0);)

//一般来说，实现下面的成员函数是UART驱动的主体工作  
staticstruct uart\_ops **s3c24xx\_serial\_ops**={

.pm =s3c24xx\_serial\_pm, //电源管理函数

.tx\_empty = s3c24xx\_serial\_tx\_empty, //检车发送FIFO缓冲区是否空

.get\_mctrl = s3c24xx\_serial\_get\_mctrl, //是否串口流控

.set\_mctrl = s3c24xx\_serial\_set\_mctrl, //是否设置串口流控cts

.stop\_tx =s3c24xx\_serial\_stop\_tx, //停止发送

.start\_tx =s3c24xx\_serial\_start\_tx, //启动发送

.stop\_rx =s3c24xx\_serial\_stop\_rx, //停止接收

.enable\_ms = s3c24xx\_serial\_enable\_ms, //空函数

.break\_ctl = s3c24xx\_serial\_break\_ctl, //发送break信号

.startup =s3c24xx\_serial\_startup, //串口发送/接收，以及中断申请初始配置函数

.shutdown = s3c24xx\_serial\_shutdown, //关闭串口

.set\_termios = s3c24xx\_serial\_set\_termios,//串口clk,波特率，数据位等参数设置

.type = s3c24xx\_serial\_type, // CPU类型关于串口

.release\_port =s3c24xx\_serial\_release\_port, //释放串口

.request\_port =s3c24xx\_serial\_request\_port, //申请串口

.config\_port = s3c24xx\_serial\_config\_port, //串口的一些配置信息info

.verify\_port = s3c24xx\_serial\_verify\_port, //串口检测

.wake\_peer = s3c24xx\_serial\_wake\_peer,

};

[复制代码](javascript:void(0);)

而在serial\_core.c中定义了tty\_operations的实例，包含uart\_open()；uart\_close()；uart\_send\_xchar()等成员函数，这些函数借助uart\_ops结构体中的成员函数来完成具体的操作。

staticconststruct tty\_operations uart\_ops = {

。。。。。。

从下面的例子中可以看出串口核心层的tty\_operations与uart\_ops的关系：

从下面的例子中可以看出串口核心层的tty\_operations与uart\_ops的关系：

[复制代码](javascript:void(0);)

/\*

\* This function is used to send a high-priority XON/XOFF character to

\* the device

\*/

staticvoid uart\_send\_xchar(struct tty\_struct \*tty, char ch)

{

struct uart\_state \*state = tty->driver\_data;

struct uart\_port \*port = state->uart\_port;

unsigned long flags;

if (port->ops->send\_xchar)/\*如果uart\_ops中实现了send\_xchar成员函数\*/

port->ops->send\_xchar(port, ch);

else {

port->x\_char = ch;

if (ch) {

spin\_lock\_irqsave(&port->lock, flags);

port->ops->start\_tx(port);

spin\_unlock\_irqrestore(&port->lock, flags);

}

}

}

这个例子的调用关系如下:

**send\_xchar ----> uart\_send\_xchar ----> start\_tx ---> s3c24xx\_serial\_start\_tx**

## [ARM-Linux S5PV210 UART驱动（4）----串口驱动初始化过程](http://www.cnblogs.com/hello2mhb/p/3344159.html)

对于S5PV210 UART驱动来说，主要关心的就是drivers/serial下的samsung.c和s5pv210.c连个文件。

由drivers/serial/Kconfig:

　　config SERIAL\_SAMSUNG  
　　　　depends on ARM && PLAT\_SAMSUNG

　　config SERIAL\_S5PV210  
　　　　depends on SERIAL\_SAMSUNG && (CPU\_S5PV210 || CPU\_S5P6442) && SERIAL\_SAMSUNG\_CONSOLE

可以看出模块的依赖关系，先加载samsung.ko，然后再加载s5pv210.ko。

所以**串口的初始化**的简要过程如下：

samsung.c中模块加载函数s3c24xx\_serial\_modinit调用uart\_register\_driver(&s3c24xx\_uart\_drv)，注册了s3c24xx\_uart\_drv这个uart\_driver；

s5pv210.c中模块加载函数s5p\_serial\_init ----> s3c24xx\_serial\_init(&s5p\_serial\_driver, \*s5p\_uart\_inf) ---->  platform\_driver\_register(drv) 注册了s5p\_serial\_driver这个平台驱动，

                 有了平台驱动后，当平台设备与平台驱动match之后，调用s5p\_serial\_probe ----> s3c24xx\_serial\_probe(pdev, s5p\_uart\_inf[pdev->id]) ---->  s3c24xx\_serial\_init\_port初始化UART端口

                                                                                                                                                                                                              >  uart\_add\_one\_port添加端口

**《《《《====接下来具体分析====》》》》**

一、注册uart\_driver

[复制代码](javascript:void(0);)

/\* 模块加载函数\*/

staticint \_\_init s3c24xx\_serial\_modinit(void)

{

int ret;

ret = **uart\_register\_driver**(&s3c24xx\_uart\_drv);/\*注册uart\_driver\*/

if (ret <0) {

printk(KERN\_ERR "failed to register UART driver\n");

return -1;

}

return0;

}

在模块加载函数中调用serial\_core.c中的**uart\_register\_driver()**注册s3c24xx\_uart\_drv这个uart\_driver，实际上在uart\_register\_driver()中包含了tty\_register\_driver()，代码如下：

/\*实际上是填充uart\_driver结构体\*/

intuart\_register\_driver(struct uart\_driver \*drv)

{

。。。。。。

二、注册平台驱动

s5p\_serial\_init ----> s3c24xx\_serial\_init(&s5p\_serial\_driver, \*s5p\_uart\_inf) ---->  platform\_driver\_register(drv)

[复制代码](javascript:void(0);)

int s3c24xx\_serial\_init(struct platform\_driver \*drv,

struct s3c24xx\_uart\_info \*info)

{

dbg("s3c24xx\_serial\_init(%p,%p)\n", drv, info);

#ifdef CONFIG\_PM

drv->suspend = s3c24xx\_serial\_suspend;

drv->resume = s3c24xx\_serial\_resume;

#endif

return platform\_driver\_register(drv);

}

直接调用platform\_driver\_register，注册了 s5p\_serial\_driver这个平台驱动。

三、平台驱动的探测函数probe()

因为把uart驱动注册为platform驱动，当平台驱动与平台设备进行匹配的时候会调用平台总线的match函数，匹配成功后就会调用平台驱动的xxx\_probe()函数来进行一系列的初始化工作。

UART驱动的probe()调用过程如下：

s5p\_serial\_probe ----> s3c24xx\_serial\_probe(pdev, s5p\_uart\_inf[pdev->id])

最终调用的是s3c24xx\_serial\_probe();

详细的代码分析如下：

/\* Device driver serial port probe \*/

int s3c24xx\_serial\_probe(struct platform\_device \*dev,

struct s3c24xx\_uart\_info \*info)

{

。。。。。。

其中s3c24xx\_serial\_init\_port函数的分析如下：

staticint s3c24xx\_serial\_init\_port(struct s3c24xx\_uart\_port \*ourport,

struct s3c24xx\_uart\_info \*info,

struct platform\_device \*platdev)

{

。。。。。。

**《《======总结=======》》**

到这一步所有的tty和uart初始化的部分算是完成了。也就是完成了下图的构建：



但是目前还不能进行read、write操作。

因为还有一个重要的函数tty\_open没有分析，作为tty设备的uart，这个tty\_open函数也是非常重要的，下篇分析。

初始化的工作主要是

1.初始化uart\_driver结构体，包括初始化uart\_driver结构体中的tty\_driver uart\_state。

2.uart\_state部分的初始化主要是初始化其中的uart\_port，这部分的初始化在probe函数总完成

修改驱动需要设计的数据结构

1.uart\_driver

uart\_driver中的数据用于初始化tty\_driver

2.s3c24xx\_uart\_info

用于初始化uart\_port

3.s3c24xx\_uart\_port或者说uart\_port。

uart\_port的初始化，在s3c24xx的uart驱动中uart\_port是内嵌在s3c24xx\_uart\_port中的

4.uart\_ops

最底层的硬件操作。

## [ARM-Linux S5PV210 UART驱动（5）----串口的open操作（tty\_open、uart\_open）](http://www.cnblogs.com/hello2mhb/p/3359547.html)

串口驱动初始化后，串口作为字符驱动也已经注册到系统了，/dev目录下也有设备文件节点了。

那接下来uart的操作是如何进行的呢?

操作硬件之前都是要先open设备，先来分析下这里的open函数具体做了那些工作。

s3c24xx\_serial\_modinit -->uart\_register\_driver -->tty\_register\_driver 中有如下语句：

cdev\_init(&driver->cdev, &tty\_fops);

此处将 driver->cdev->ops=&tty\_fops

而tty\_fops如下：

[复制代码](javascript:void(0);)

staticconststruct file\_operations tty\_fops = {

.llseek = no\_llseek,

.read = tty\_read,

.write = tty\_write,

.poll = tty\_poll,

.unlocked\_ioctl = tty\_ioctl,

.compat\_ioctl = tty\_compat\_ioctl,

.open = tty\_open,

.release = tty\_release,

.fasync = tty\_fasync,

};

所以应用层通过open系统调用open(“/dev/s3c2410\_serial0”，)一层一层调用到会调用到tty\_open。

staticint tty\_open(struct inode \*inode, struct file \*filp)

{

。。。。。。

staticint uart\_open(struct tty\_struct \*tty, struct file \*filp)

{

。。。。。。

staticint uart\_startup(struct uart\_state \*state, int init\_hw)

{。。。。。。

retval = uport->ops->startup(uport); ==========>>>>>>>>>调用s3c24xx\_serial\_startup()

。。。。。。

至此，通过open函数后s3c24xx的uart硬件部分也初始化好了，接着可以通过write、read函数收发数据了。

《《《《=============总结===============》》》》

open的主要作用是 在内核通过创建并初始化一个tty\_struct来描述具体对应的一个硬件设备，比如这里就是用一个tty\_struct来描述s3c24xx上的uart0的，然后找到uart\_port

中ops的startup方法初始化uart的硬件。

具体的tty\_struct初始化过程中最重要的几步如下

1.初始化tty-struct的ops，就是将tty\_driver中的ops赋值给tty\_struct

2.初始化tty线路规程操作集

3.初始化tty\_struct中的uart\_state，uart\_state中包含uart\_port信息，这一步通过步骤1中ops中的open方法来完成。

4.根据步骤3中找到的uart\_state，找到里面的uart\_port的ops中的startup方法来初始化uart硬件。

open的流程大致如下：

open

--tty\_open

    |

    --get\_tty\_driver

    --tty\_init\_dev

    --tty->ops->open(uart\_open)

        |

        --uart\_startup

            |

            --uport->ops->startup(s3c24xx\_serial\_startup())

                |

                --request\_irq(rx\_irq)

                --request\_irq(tx\_irq)

## [ARM-Linux S5PV210 UART驱动（6）----platform device的添加](http://www.cnblogs.com/hello2mhb/p/3337555.html)

开发板是飞凌OK210

**arch/arm/mach-s5pv210/mach-smdkc110.c**

首先是UART的寄存器默认配置信息：。。。。。。

下面是添加platform device的具体过程：

1.

1.

[复制代码](javascript:void(0);)

/\*调用MACHINE\_START宏

MACHINE\_START和MACHINE\_END框起了一个machine\_desc结构体的声明并根据MACHINE\_START宏的参数初始化其.nr和.name成员

并将该结构体标记编译到.arch.info.init段

在MACHINE\_START和MACHINE\_END宏之间可以初始化machine\_desc结构体的剩余成员

\*/

#ifdef CONFIG\_MACH\_SMDKC110

MACHINE\_START(SMDKC110, "SMDKC110")

#elif CONFIG\_MACH\_SMDKV210

MACHINE\_START(SMDKV210, "SMDKV210")

#endif

/\* Maintainer: Kukjin Kim <kgene.kim@samsung.com> \*/

.phys\_io = S3C\_PA\_UART &0xfff00000,

.io\_pg\_offst = (((u32)S3C\_VA\_UART) >>18) &0xfffc,

.boot\_params = S5P\_PA\_SDRAM + 0x100,

.init\_irq = s5pv210\_init\_irq,//板级中断初始化函数

.map\_io = **smdkc110\_map\_io**,//板级io初始化函数

.init\_machine = smdkc110\_machine\_init,//板级初始化函数

.timer = &s5p\_systimer,

MACHINE\_END

2.

[复制代码](javascript:void(0);)

staticvoid \_\_init smdkc110\_map\_io(void)

{

s5p\_init\_io(NULL, 0, S5P\_VA\_CHIPID);

s3c24xx\_init\_clocks(24000000);

s5pv210\_gpiolib\_init();

**s3c24xx\_init\_uarts(smdkc110\_uartcfgs, ARRAY\_SIZE(smdkc110\_uartcfgs))**;

s5p\_reserve\_bootmem(smdkc110\_media\_devs, ARRAY\_SIZE(smdkc110\_media\_devs));

#ifdef CONFIG\_MTD\_ONENAND

s5pc110\_device\_onenand.name = "s5pc110-onenand";

#endif

#ifdef CONFIG\_MTD\_NAND

s3c\_device\_nand.name = "s5pv210-nand";

#endif

s5p\_device\_rtc.name = "smdkc110-rtc";

}

3.

[复制代码](javascript:void(0);)

/\* table of supported CPUs \*/  
staticstruct cpu\_table cpu\_ids[] \_\_initdata = {

{

.idcode = 0x43110000,

.idmask = 0xfffff000,

.map\_io = s5pv210\_map\_io,

.init\_clocks = s5pv210\_init\_clocks,

.init\_uarts = **s5pv210\_init\_uarts**,

.init = s5pv210\_init,

.name = name\_s5pv210,

},

};

arch/arm/plat-samsung/init.c

[复制代码](javascript:void(0);)

void \_\_init s3c24xx\_init\_uarts(struct s3c2410\_uartcfg \*cfg, int no)

{

if (cpu == NULL)

return;

if (cpu->init\_uarts == NULL) {

printk(KERN\_ERR "s3c24xx\_init\_uarts: cpu has no uart init\n");

} else

(cpu->init\_uarts)(cfg, no);//这里最终会调用上面的s5pv210\_init\_uarts

}

 4.

#define s5pv210\_init\_uarts s5pv210\_common\_init\_uarts

[复制代码](javascript:void(0);)

/\* uart registration process \*/

void \_\_init s5pv210\_common\_init\_uarts(struct s3c2410\_uartcfg \*cfg, int no)

{

struct s3c2410\_uartcfg \*tcfg = cfg;

u32 ucnt;

for (ucnt = 0; ucnt < no; ucnt++, tcfg++) {

if (!tcfg->clocks) {

tcfg->clocks = s5pv210\_serial\_clocks;

tcfg->clocks\_size = ARRAY\_SIZE(s5pv210\_serial\_clocks);

}

}

**s3c24xx\_init\_uartdevs**("s5pv210-uart", s5p\_uart\_resources, cfg, no);

}

5.

arch/arm/plat-samsung/init.c

[复制代码](javascript:void(0);)

void \_\_init s3c24xx\_init\_uartdevs(char \*name,

struct s3c24xx\_uart\_resources \*res,

struct s3c2410\_uartcfg \*cfg, int no)

{

struct platform\_device \*platdev;

struct s3c2410\_uartcfg \*cfgptr = uart\_cfgs;

struct s3c24xx\_uart\_resources \*resp;

int uart;

memcpy(cfgptr, cfg, sizeof(struct s3c2410\_uartcfg) \* no);

for (uart = 0; uart < no; uart++, cfg++, cfgptr++) {

platdev = s3c24xx\_uart\_src[cfgptr->hwport];

resp = res + cfgptr->hwport;

s3c24xx\_uart\_devs[uart] = platdev;

platdev->name = name;

platdev->resource = resp->resources;

platdev->num\_resources = resp->nr\_resources;

platdev->dev.platform\_data = cfgptr;//将cfg挂到platdev->dev.platform\_data上

}

nr\_uarts = no;

}

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

staticint \_\_init s3c\_arch\_init(void)

{

int ret;

// do the correct init for cpu

if (cpu == NULL)

panic("s3c\_arch\_init: NULL cpu\n");

ret = (cpu->init)();

if (ret != 0)

return ret;

ret = platform\_add\_devices(s3c24xx\_uart\_devs, nr\_uarts);

return ret;

}

arch\_initcall(s3c\_arch\_init);

## 终端设备tty中的数据结构

### 结构体：tty\_struct

struct tty\_struct {

int magic;

struct kref kref;

struct tty\_driver \*driver;

const struct tty\_operations \*ops;

int index;

/\* Protects ldisc changes: Lock tty not pty \*/

struct mutex ldisc\_mutex;

struct tty\_ldisc \*ldisc;

struct mutex termios\_mutex;

spinlock\_t ctrl\_lock;

/\* Termios values are protected by the termios mutex \*/

struct ktermios \*termios, \*termios\_locked;

struct termiox \*termiox; /\* May be NULL for unsupported \*/

char name[64];

struct pid \*pgrp; /\* Protected by ctrl lock \*/

struct pid \*session;

unsigned long flags;

int count;

struct winsize winsize; /\* termios mutex \*/

unsigned char stopped:1, hw\_stopped:1, flow\_stopped:1, packet:1;

unsigned char low\_latency:1, warned:1;

unsigned char ctrl\_status; /\* ctrl\_lock \*/

unsigned int receive\_room; /\* Bytes free for queue \*/

struct tty\_struct \*link;

struct fasync\_struct \*fasync;

struct tty\_bufhead buf; /\* Locked internally \*/

int alt\_speed; /\* For magic substitution of 38400 bps \*/

wait\_queue\_head\_t write\_wait;

wait\_queue\_head\_t read\_wait;

struct work\_struct hangup\_work;

void \*disc\_data;

void \*driver\_data;

struct list\_head tty\_files;

#define N\_TTY\_BUF\_SIZE 4096

/\*

\* The following is data for the N\_TTY line discipline. For

\* historical reasons, this is included in the tty structure.

\* Mostly locked by the BKL.

\*/

unsigned int column;

unsigned char lnext:1, erasing:1, raw:1, real\_raw:1, icanon:1;

unsigned char closing:1;

unsigned char echo\_overrun:1;

unsigned short minimum\_to\_wake;

unsigned long overrun\_time;

int num\_overrun;

unsigned long process\_char\_map[256/(8\*sizeof(unsigned long))];

char \*read\_buf;

int read\_head;

int read\_tail;

int read\_cnt;

unsigned long read\_flags[N\_TTY\_BUF\_SIZE/(8\*sizeof(unsigned long))];

unsigned char \*echo\_buf;

unsigned int echo\_pos;

unsigned int echo\_cnt;

int canon\_data;

unsigned long canon\_head;

unsigned int canon\_column;

struct mutex atomic\_read\_lock;

struct mutex atomic\_write\_lock;

struct mutex output\_lock;

struct mutex echo\_lock;

unsigned char \*write\_buf;

int write\_cnt;

spinlock\_t read\_lock;

/\* If the tty has a pending do\_SAK, queue it here - akpm \*/

struct work\_struct SAK\_work;

struct tty\_port \*port;

};

### 结构体：tty\_buffer

struct tty\_buffer {

struct tty\_buffer \*next;

char \*char\_buf\_ptr;

unsigned char \*flag\_buf\_ptr;

int used;

int size;

int commit;

int read;

/\* Data points here \*/

unsigned long data[0];

};

### 结构体：tty\_port

struct tty\_port {

struct tty\_struct \*tty; /\* Back pointer \*/

const struct tty\_port\_operations \*ops; /\* Port operations \*/

spinlock\_t lock; /\* Lock protecting tty field \*/

int blocked\_open; /\* Waiting to open \*/

int count; /\* Usage count \*/

wait\_queue\_head\_t open\_wait; /\* Open waiters \*/

wait\_queue\_head\_t close\_wait; /\* Close waiters \*/

wait\_queue\_head\_t delta\_msr\_wait; /\* Modem status change \*/

unsigned long flags; /\* TTY flags ASY\_\*/

struct mutex mutex; /\* Locking \*/

unsigned char \*xmit\_buf; /\* Optional buffer \*/

unsigned int close\_delay; /\* Close port delay \*/

unsigned int closing\_wait; /\* Delay for output \*/

int drain\_delay; /\* Set to zero if no pure time

based drain is needed else

set to size of fifo \*/

};

### 结构体：tty\_port\_operations

struct tty\_port\_operations {

/\* Return 1 if the carrier is raised \*/

int (\*carrier\_raised)(struct tty\_port \*port);

/\* Control the DTR line \*/

void (\*dtr\_rts)(struct tty\_port \*port, int raise);

/\* Called when the last close completes or a hangup finishes

IFF the port was initialized. Do not use to free resources \*/

void (\*shutdown)(struct tty\_port \*port);

void (\*drop)(struct tty\_port \*port);

};

### 结构体：tty\_operations

struct tty\_operations {

struct tty\_struct \* (\*lookup)(struct tty\_driver \*driver,

struct inode \*inode, int idx);

int (\*install)(struct tty\_driver \*driver, struct tty\_struct \*tty);

void (\*remove)(struct tty\_driver \*driver, struct tty\_struct \*tty);

int (\*open)(struct tty\_struct \* tty, struct file \* filp);

void (\*close)(struct tty\_struct \* tty, struct file \* filp);

void (\*shutdown)(struct tty\_struct \*tty);

void (\*cleanup)(struct tty\_struct \*tty);

int (\*write)(struct tty\_struct \* tty,

const unsigned char \*buf, int count);

int (\*put\_char)(struct tty\_struct \*tty, unsigned char ch);

void (\*flush\_chars)(struct tty\_struct \*tty);

int (\*write\_room)(struct tty\_struct \*tty);

int (\*chars\_in\_buffer)(struct tty\_struct \*tty);

int (\*ioctl)(struct tty\_struct \*tty, struct file \* file,

unsigned int cmd, unsigned long arg);

long (\*compat\_ioctl)(struct tty\_struct \*tty, struct file \* file,

unsigned int cmd, unsigned long arg);

void (\*set\_termios)(struct tty\_struct \*tty, struct ktermios \* old);

void (\*throttle)(struct tty\_struct \* tty);

void (\*unthrottle)(struct tty\_struct \* tty);

void (\*stop)(struct tty\_struct \*tty);

void (\*start)(struct tty\_struct \*tty);

void (\*hangup)(struct tty\_struct \*tty);

int (\*break\_ctl)(struct tty\_struct \*tty, int state);

void (\*flush\_buffer)(struct tty\_struct \*tty);

void (\*set\_ldisc)(struct tty\_struct \*tty);

void (\*wait\_until\_sent)(struct tty\_struct \*tty, int timeout);

void (\*send\_xchar)(struct tty\_struct \*tty, char ch);

int (\*tiocmget)(struct tty\_struct \*tty, struct file \*file);

int (\*tiocmset)(struct tty\_struct \*tty, struct file \*file,

unsigned int set, unsigned int clear);

int (\*resize)(struct tty\_struct \*tty, struct winsize \*ws);

int (\*set\_termiox)(struct tty\_struct \*tty, struct termiox \*tnew);

#ifdef CONFIG\_CONSOLE\_POLL

int (\*poll\_init)(struct tty\_driver \*driver, int line, char \*options);

int (\*poll\_get\_char)(struct tty\_driver \*driver, int line);

void (\*poll\_put\_char)(struct tty\_driver \*driver, int line, char ch);

#endif

const struct file\_operations \*proc\_fops;

};

### 结构体：tty\_driver

struct tty\_driver {

int magic; /\* magic number for this structure \*/

struct kref kref; /\* Reference management \*/

struct cdev cdev;

struct module \*owner;

const char \*driver\_name;

const char \*name;

int name\_base; /\* offset of printed name \*/

int major; /\* major device number \*/

int minor\_start; /\* start of minor device number \*/

int minor\_num; /\* number of \*possible\* devices \*/

int num; /\* number of devices allocated \*/

short type; /\* type of tty driver \*/

short subtype; /\* subtype of tty driver \*/

struct ktermios init\_termios; /\* Initial termios \*/

int flags; /\* tty driver flags \*/

struct proc\_dir\_entry \*proc\_entry; /\* /proc fs entry \*/

struct tty\_driver \*other; /\* only used for the PTY driver \*/

/\*

\* Pointer to the tty data structures

\*/

struct tty\_struct \*\*ttys;

struct ktermios \*\*termios;

struct ktermios \*\*termios\_locked;

void \*driver\_state;

/\*

\* Driver methods

\*/

const struct tty\_operations \*ops;

struct list\_head tty\_drivers;

};

## tty线路规程中的数据结构(Line discipline)

### 结构体：tty\_ldisc\_ops

struct tty\_ldisc\_ops tty\_ldisc\_N\_TTY = {

.magic = TTY\_LDISC\_MAGIC,

.name = "n\_tty",

.open = n\_tty\_open,

.close = n\_tty\_close,

.flush\_buffer = n\_tty\_flush\_buffer,

.chars\_in\_buffer = n\_tty\_chars\_in\_buffer,

.read = n\_tty\_read,

.write = n\_tty\_write,

.ioctl = n\_tty\_ioctl,

.set\_termios = n\_tty\_set\_termios,

.poll = n\_tty\_poll,

.receive\_buf = n\_tty\_receive\_buf,

.write\_wakeup = n\_tty\_write\_wakeup

};

### 结构体：tty\_ldisc

struct tty\_ldisc {

struct tty\_ldisc\_ops \*ops;

atomic\_t users;

};

## 串口驱动中的数据结构

串口驱动提供的功能：初始化串口、打开串口、读串口（接收数据）、写串口（发送数据）等功能。

uart\_driver里面包含uart\_state结构里面包含uart\_port结构里面包含uart\_ops结构里面包含各种串口操作函数。

**uart驱动程序结构：struct uart\_driver**

**uart端口结构：struct uart\_port**

**uart相关操作函数结构：struct uart\_ops**

uart状态结构：struct uart\_state

uart信息结构：struct uart\_info

#### 结构体：uart\_driver

struct uart\_driver {

**struct module \*owner; //THIS\_MODULE**

**const char \*driver\_name; //serial\_name:ttySAC**

**const char \*dev\_name; //device name**

**int major; //major**

**int minor; //minor**

**int nr; //number of ports**

**struct console \*cons; //console interface**

/\*

\* these are private; the low level driver should not

\* touch these; they should be initialised to NULL

\*/

struct uart\_state \*state;

struct tty\_driver \*tty\_driver;

};

#### 结构体：uart\_state

struct uart\_state {

struct tty\_port port;

int pm\_state;

struct circ\_buf xmit;

struct tasklet\_struct tlet;

struct uart\_port \*uart\_port;

};

#### 结构体：uart\_port

一个uart\_port代表一个串口，例如一个板子上有3个串口，那么只有一个uart\_driver，有3个uart\_port结构，每一个uart\_port对应一个串口。

**uart\_port结构**

struct uart\_port {

spinlock\_t lock; /\* port lock \*/

unsigned long iobase; /\* in/out[bwl] \*/

unsigned char \_\_iomem \*membase; /\* read/write[bwl] \*/

unsigned int (\*serial\_in)(struct uart\_port \*, int);

void (\*serial\_out)(struct uart\_port \*, int, int);

unsigned int irq; /\* irq number \*/

unsigned long irqflags; /\* irq flags \*/

unsigned int uartclk; /\* base uart clock \*/

unsigned int fifosize; /\* tx fifo size \*/

unsigned char x\_char; /\* xon/xoff char \*/

unsigned char regshift; /\* reg offset shift \*/

unsigned char iotype; /\* io access style \*/

unsigned char unused1;

#define UPIO\_PORT (0)

#define UPIO\_HUB6 (1)

#define UPIO\_MEM (2)

#define UPIO\_MEM32 (3)

#define UPIO\_AU (4) /\* Au1x00 type IO \*/

#define UPIO\_TSI (5) /\* Tsi108/109 type IO \*/

#define UPIO\_DWAPB (6) /\* DesignWare APB UART \*/

#define UPIO\_RM9000 (7) /\* RM9000 type IO \*/

unsigned int read\_status\_mask; /\* driver specific \*/

unsigned int ignore\_status\_mask; /\* driver specific \*/

struct uart\_state \*state; /\* pointer to parent state \*/

struct uart\_icount icount; /\* statistics \*/

struct console \*cons; /\* struct console, if any \*/

#if defined(CONFIG\_SERIAL\_CORE\_CONSOLE) || defined(SUPPORT\_SYSRQ)

unsigned long sysrq; /\* sysrq timeout \*/

#endif

upf\_t flags;

#define UPF\_FOURPORT ((\_\_force upf\_t) (1 << 1))

#define UPF\_SAK ((\_\_force upf\_t) (1 << 2))

#define UPF\_SPD\_MASK ((\_\_force upf\_t) (0x1030))

#define UPF\_SPD\_HI ((\_\_force upf\_t) (0x0010))

#define UPF\_SPD\_VHI ((\_\_force upf\_t) (0x0020))

#define UPF\_SPD\_CUST ((\_\_force upf\_t) (0x0030))

#define UPF\_SPD\_SHI ((\_\_force upf\_t) (0x1000))

#define UPF\_SPD\_WARP ((\_\_force upf\_t) (0x1010))

#define UPF\_SKIP\_TEST ((\_\_force upf\_t) (1 << 6))

#define UPF\_AUTO\_IRQ ((\_\_force upf\_t) (1 << 7))

#define UPF\_HARDPPS\_CD ((\_\_force upf\_t) (1 << 11))

#define UPF\_LOW\_LATENCY ((\_\_force upf\_t) (1 << 13))

#define UPF\_BUGGY\_UART ((\_\_force upf\_t) (1 << 14))

#define UPF\_NO\_TXEN\_TEST ((\_\_force upf\_t) (1 << 15))

#define UPF\_MAGIC\_MULTIPLIER ((\_\_force upf\_t) (1 << 16))

#define UPF\_CONS\_FLOW ((\_\_force upf\_t) (1 << 23))

#define UPF\_SHARE\_IRQ ((\_\_force upf\_t) (1 << 24))

/\* The exact UART type is known and should not be probed. \*/

#define UPF\_FIXED\_TYPE ((\_\_force upf\_t) (1 << 27))

#define UPF\_BOOT\_AUTOCONF ((\_\_force upf\_t) (1 << 28))

#define UPF\_FIXED\_PORT ((\_\_force upf\_t) (1 << 29))

#define UPF\_DEAD ((\_\_force upf\_t) (1 << 30))

#define UPF\_IOREMAP ((\_\_force upf\_t) (1 << 31))

#define UPF\_CHANGE\_MASK ((\_\_force upf\_t) (0x17fff))

#define UPF\_USR\_MASK ((\_\_force upf\_t) (UPF\_SPD\_MASK|UPF\_LOW\_LATENCY))

unsigned int mctrl; /\* current modem ctrl settings \*/

unsigned int timeout; /\* character-based timeout \*/

unsigned int type; /\* port type \*/

const struct uart\_ops \*ops;

unsigned int custom\_divisor;

unsigned int line; /\* port index \*/

resource\_size\_t mapbase; /\* for ioremap \*/

struct device \*dev; /\* parent device \*/

unsigned char hub6; /\* this should be in the 8250 driver \*/

unsigned char suspended;

unsigned char unused[2];

void \*private\_data; /\* generic platform data pointer \*/

};

#### 结构体：uart\_ops

struct uart\_ops {

unsigned int (\*tx\_empty)(struct uart\_port \*);

void (\*set\_mctrl)(struct uart\_port \*, unsigned int mctrl);

unsigned int (\*get\_mctrl)(struct uart\_port \*);

void (\*stop\_tx)(struct uart\_port \*);

void (\*start\_tx)(struct uart\_port \*);

void (\*send\_xchar)(struct uart\_port \*, char ch);

void (\*stop\_rx)(struct uart\_port \*);

void (\*enable\_ms)(struct uart\_port \*);

void (\*break\_ctl)(struct uart\_port \*, int ctl);

int (\*startup)(struct uart\_port \*);

void (\*shutdown)(struct uart\_port \*);

void (\*flush\_buffer)(struct uart\_port \*);

void (\*set\_termios)(struct uart\_port \*, struct ktermios \*new,

struct ktermios \*old);

void (\*set\_ldisc)(struct uart\_port \*);

void (\*pm)(struct uart\_port \*, unsigned int state,

unsigned int oldstate);

int (\*set\_wake)(struct uart\_port \*, unsigned int state);

/\*

\* Return a string describing the type of the port

\*/

const char \*(\*type)(struct uart\_port \*);

/\*

\* Release IO and memory resources used by the port.

\* This includes iounmap if necessary.

\*/

void (\*release\_port)(struct uart\_port \*);

/\*

\* Request IO and memory resources used by the port.

\* This includes iomapping the port if necessary.

\*/

int (\*request\_port)(struct uart\_port \*);

void (\*config\_port)(struct uart\_port \*, int);

int (\*verify\_port)(struct uart\_port \*, struct serial\_struct \*);

int (\*ioctl)(struct uart\_port \*, unsigned int, unsigned long);

#ifdef CONFIG\_CONSOLE\_POLL

void (\*poll\_put\_char)(struct uart\_port \*, unsigned char);

int (\*poll\_get\_char)(struct uart\_port \*);

#endif

};

## 串口驱动分析：初始化

**串口驱动：串口驱动初始化流程**

uart\_register\_driver();

platform\_driver\_register();

s3c24xx\_serial\_probe();

取出相应uar\_port结构

初始化uart\_port结构

获取串口虚拟基地址

获取中断号:platform\_get\_irq

复位fifo:s3c24xx\_serial\_resetport()

添加uart\_port:uart\_add\_one\_port()

创建属性文件:device\_create\_file()

初始化动态频率调节:s3c24xx\_serial\_cpufreq\_register()

uart\_unregister\_driver();

## 串口驱动分析：打开设备

**串口驱动：打开设备流程**

/dev/ttySAC

uart\_register\_driver();

tty\_register\_driver();

cdev\_init();

tty\_fops

uart\_ops:open

uart\_start()

s3c24xx\_serial\_startup();

使能串口接收功能:rx\_enable();

注册数据接收中断:request\_irq();

使能串口发送功能:tx\_enable();

注册数据发送中断:request\_irq();

cdev\_add();

s3c24xx\_serial\_ports[]= {...};

s3c24xx\_serial\_ops

## 串口驱动分析：数据发送

**串口驱动：数据发送流程**

用户程序:write()>>>tty>>>串口驱动:file\_operation:从驱动注册函数处开始切入

s3c24xx\_serial\_modinit()

tty\_register\_driver():tty\_fops{}:tty\_write()

tty\_write()

do\_tty\_write():tty\_ldisc\_N\_TTY{}:n\_tty\_write()

n\_tty\_write()

uart\_ops{}:uart\_write()

uart\_write()

uart\_start():s3c24xx\_uart\_port:ops:uart\_ops

:s3c24xx\_serial\_ops{}:

start\_tx:s3c24xx\_serial\_start\_tx

step02 串口驱动是如何发送数据的

s3c24xx\_serial\_start\_tx():

s3c24xx\_serial\_rx\_disable();

enable\_irq();

tx\_enabled()=1;

由上看出start\_tx感觉并没有写数据（操作寄存器）的操作，那发送数据怎么实现呢？

看出串口驱动是在中断处理程序中来完成对数据的处理（发送）的，从中断注册关键字（request\_irq()）寻找入口，于是我们找到了s3c24xx\_serial\_tx\_chars()函数，然后定位到该函数的实现处，于是我们发现这个函数开始处理数据了（操作寄存器:wr\_regb()）了。

但是发送的数据从什么地方来的呢？

实际上串口有个非常重要的部分，叫循环缓冲，

向串口发送数据文字描述

应用程序使用write()系统调用，

系统调用根据文件描述符获取该文件属性为字符设备文件且为串口设备，

经过tty处理(tty\_write():n\_tty\_write():uart\_write())，

tty处理过程中(uart\_write())就会把数据放到给这个串口开辟的循环缓冲中去，

然后uart\_write():uart\_start():\_\_uart\_start():port->ops->start\_tx(),

通过追踪port->ops->定位到uart\_ops{}结构，

然后根据uart\_ops结构定位到s3c24xx\_serial\_ops{}定义，

通过s3c24xx\_serial\_ops{}中.startup项定位到s3c24xx\_serial\_startup()，

然后定位到s3c24xx\_serial\_startup()实现处，

s3c24xx\_serial\_startup()做了注册读中断和写中断，并使能读端口和写端口，

从这个中断函数request\_irq()入口，分别有读串口和写串口的函数，

两个硬件读写函数:s3c24xx\_serial\_rx\_chars(),s3c24xx\_serial\_tx\_chars()，

而这个两个函数分别将数据从循环缓冲中写入串口相应寄存器。

最终找到了驱动s3c24xx\_serial\_start\_tx()函数，

那么s3c24xx\_serial\_start\_tx函数呢就激活了中断(enable\_irq())，

然后它发现fifo里没数据，于是中断就被激活，

那么就调用中断处理程序来发送数据，

中断处理程序就会从这个循环缓冲中把数据取走。

最后通过写入到相应寄存器最终把数据发送走。

## 串口驱动分析：数据接收

## 内核中串口驱动范例源代码分析

### drivers/serial/timbuart.h

/\*

\* timbuart.c timberdale FPGA GPIO driver

\* Copyright (c) 2009 Intel Corporation

\*

\* This program is free software; you can redistribute it and/or modify

\* it under the terms of the GNU General Public License version 2 as

\* published by the Free Software Foundation.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program; if not, write to the Free Software

\* Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

\*/

/\* Supports:

\* Timberdale FPGA UART

\*/

#ifndef \_TIMBUART\_H

#define \_TIMBUART\_H

#define TIMBUART\_FIFO\_SIZE 2048

#define TIMBUART\_RXFIFO 0x08

#define TIMBUART\_TXFIFO 0x0c

#define TIMBUART\_IER 0x10

#define TIMBUART\_IPR 0x14

#define TIMBUART\_ISR 0x18

#define TIMBUART\_CTRL 0x1c

#define TIMBUART\_BAUDRATE 0x20

#define TIMBUART\_CTRL\_RTS 0x01

#define TIMBUART\_CTRL\_CTS 0x02

#define TIMBUART\_CTRL\_FLSHTX 0x40

#define TIMBUART\_CTRL\_FLSHRX 0x80

#define TXBF 0x01

#define TXBAE 0x02

#define CTS\_DELTA 0x04

#define RXDP 0x08

#define RXBAF 0x10

#define RXBF 0x20

#define RXTT 0x40

#define RXBNAE 0x80

#define TXBE 0x100

#define RXFLAGS (RXDP | RXBAF | RXBF | RXTT | RXBNAE)

#define TXFLAGS (TXBF | TXBAE)

#define TIMBUART\_MAJOR 204

#define TIMBUART\_MINOR 192

#endif /\* \_TIMBUART\_H \*/

### drivers/serial/timbuart.c

/\*

\* timbuart.c timberdale FPGA UART driver

\* Copyright (c) 2009 Intel Corporation

\*

\* This program is free software; you can redistribute it and/or modify

\* it under the terms of the GNU General Public License version 2 as

\* published by the Free Software Foundation.

\*

\* This program is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program; if not, write to the Free Software

\* Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

\*/

/\* Supports:

\* Timberdale FPGA UART

\*/

#include <linux/pci.h>

#include <linux/interrupt.h>

#include <linux/serial\_core.h>

#include <linux/kernel.h>

#include <linux/platform\_device.h>

#include <linux/ioport.h>

#include "timbuart.h"

struct timbuart\_port **{**

struct uart\_port port**;**

struct tasklet\_struct tasklet**;**

int usedma**;**

u32 last\_ier**;**

struct platform\_device **\***dev**;**

**};**

static int baudrates**[]** **=** **{**9600**,** 19200**,** 38400**,** 57600**,** 115200**,** 230400**,** 460800**,**

921600**,** 1843200**,** 3250000**};**

static void timbuart\_mctrl\_check**(**struct uart\_port **\***port**,** u32 isr**,** u32 **\***ier**);**

static irqreturn\_t timbuart\_handleinterrupt**(**int irq**,** void **\***devid**);**

static void timbuart\_stop\_rx**(**struct uart\_port **\***port**)**

**{**

/\* spin lock held by upper layer, disable all RX interrupts \*/

u32 ier **=** ioread32**(**port**->**membase **+** TIMBUART\_IER**)** **&** **~**RXFLAGS**;**

iowrite32**(**ier**,** port**->**membase **+** TIMBUART\_IER**);**

**}**

static void timbuart\_stop\_tx**(**struct uart\_port **\***port**)**

**{**

/\* spinlock held by upper layer, disable TX interrupt \*/

u32 ier **=** ioread32**(**port**->**membase **+** TIMBUART\_IER**)** **&** **~**TXBAE**;**

iowrite32**(**ier**,** port**->**membase **+** TIMBUART\_IER**);**

**}**

static void timbuart\_start\_tx**(**struct uart\_port **\***port**)**

**{**

struct timbuart\_port **\***uart **=**

container\_of**(**port**,** struct timbuart\_port**,** port**);**

/\* do not transfer anything here -> fire off the tasklet \*/

tasklet\_schedule**(&**uart**->**tasklet**);**

**}**

static void timbuart\_flush\_buffer**(**struct uart\_port **\***port**)**

**{**

u8 ctl **=** ioread8**(**port**->**membase **+** TIMBUART\_CTRL**)** **|** TIMBUART\_CTRL\_FLSHTX**;**

iowrite8**(**ctl**,** port**->**membase **+** TIMBUART\_CTRL**);**

iowrite32**(**TXBF**,** port**->**membase **+** TIMBUART\_ISR**);**

**}**

static void timbuart\_rx\_chars**(**struct uart\_port **\***port**)**

**{**

struct tty\_struct **\***tty **=** port**->**state**->**port**.**tty**;**

**while** **(**ioread32**(**port**->**membase **+** TIMBUART\_ISR**)** **&** RXDP**)** **{**

u8 ch **=** ioread8**(**port**->**membase **+** TIMBUART\_RXFIFO**);**

port**->**icount**.**rx**++;**

tty\_insert\_flip\_char**(**tty**,** ch**,** TTY\_NORMAL**);**

**}**

spin\_unlock**(&**port**->**lock**);**

tty\_flip\_buffer\_push**(**port**->**state**->**port**.**tty**);**

spin\_lock**(&**port**->**lock**);**

dev\_dbg**(**port**->**dev**,** "%s - total read %d bytes\n"**,**

\_\_func\_\_**,** port**->**icount**.**rx**);**

**}**

static void timbuart\_tx\_chars**(**struct uart\_port **\***port**)**

**{**

struct circ\_buf **\***xmit **=** **&**port**->**state**->**xmit**;**

**while** **(!(**ioread32**(**port**->**membase **+** TIMBUART\_ISR**)** **&** TXBF**)** **&&**

**!**uart\_circ\_empty**(**xmit**))** **{**

iowrite8**(**xmit**->**buf**[**xmit**->**tail**],**

port**->**membase **+** TIMBUART\_TXFIFO**);**

xmit**->**tail **=** **(**xmit**->**tail **+** 1**)** **&** **(**UART\_XMIT\_SIZE **-** 1**);**

port**->**icount**.**tx**++;**

**}**

dev\_dbg**(**port**->**dev**,**

"%s - total written %d bytes, CTL: %x, RTS: %x, baud: %x\n"**,**

\_\_func\_\_**,**

port**->**icount**.**tx**,**

ioread8**(**port**->**membase **+** TIMBUART\_CTRL**),**

port**->**mctrl **&** TIOCM\_RTS**,**

ioread8**(**port**->**membase **+** TIMBUART\_BAUDRATE**));**

**}**

static void timbuart\_handle\_tx\_port**(**struct uart\_port **\***port**,** u32 isr**,** u32 **\***ier**)**

**{**

struct timbuart\_port **\***uart **=**

container\_of**(**port**,** struct timbuart\_port**,** port**);**

struct circ\_buf **\***xmit **=** **&**port**->**state**->**xmit**;**

**if** **(**uart\_circ\_empty**(**xmit**)** **||** uart\_tx\_stopped**(**port**))**

**return;**

**if** **(**port**->**x\_char**)**

**return;**

**if** **(**isr **&** TXFLAGS**)** **{**

timbuart\_tx\_chars**(**port**);**

/\* clear all TX interrupts \*/

iowrite32**(**TXFLAGS**,** port**->**membase **+** TIMBUART\_ISR**);**

**if** **(**uart\_circ\_chars\_pending**(**xmit**)** **<** WAKEUP\_CHARS**)**

uart\_write\_wakeup**(**port**);**

**}** **else**

/\* Re-enable any tx interrupt \*/

**\***ier **|=** uart**->**last\_ier **&** TXFLAGS**;**

/\* enable interrupts if there are chars in the transmit buffer,

\* Or if we delivered some bytes and want the almost empty interrupt

\* we wake up the upper layer later when we got the interrupt

\* to give it some time to go out...

\*/

**if** **(!**uart\_circ\_empty**(**xmit**))**

**\***ier **|=** TXBAE**;**

dev\_dbg**(**port**->**dev**,** "%s - leaving\n"**,** \_\_func\_\_**);**

**}**

void timbuart\_handle\_rx\_port**(**struct uart\_port **\***port**,** u32 isr**,** u32 **\***ier**)**

**{**

**if** **(**isr **&** RXFLAGS**)** **{**

/\* Some RX status is set \*/

**if** **(**isr **&** RXBF**)** **{**

u8 ctl **=** ioread8**(**port**->**membase **+** TIMBUART\_CTRL**)** **|**

TIMBUART\_CTRL\_FLSHRX**;**

iowrite8**(**ctl**,** port**->**membase **+** TIMBUART\_CTRL**);**

port**->**icount**.**overrun**++;**

**}** **else** **if** **(**isr **&** **(**RXDP**))**

timbuart\_rx\_chars**(**port**);**

/\* ack all RX interrupts \*/

iowrite32**(**RXFLAGS**,** port**->**membase **+** TIMBUART\_ISR**);**

**}**

/\* always have the RX interrupts enabled \*/

**\***ier **|=** RXBAF **|** RXBF **|** RXTT**;**

dev\_dbg**(**port**->**dev**,** "%s - leaving\n"**,** \_\_func\_\_**);**

**}**

void timbuart\_tasklet**(**unsigned long arg**)**

**{**

struct timbuart\_port **\***uart **=** **(**struct timbuart\_port **\*)**arg**;**

u32 isr**,** ier **=** 0**;**

spin\_lock**(&**uart**->**port**.**lock**);**

isr **=** ioread32**(**uart**->**port**.**membase **+** TIMBUART\_ISR**);**

dev\_dbg**(**uart**->**port**.**dev**,** "%s ISR: %x\n"**,** \_\_func\_\_**,** isr**);**

**if** **(!**uart**->**usedma**)**

timbuart\_handle\_tx\_port**(&**uart**->**port**,** isr**,** **&**ier**);**

timbuart\_mctrl\_check**(&**uart**->**port**,** isr**,** **&**ier**);**

**if** **(!**uart**->**usedma**)**

timbuart\_handle\_rx\_port**(&**uart**->**port**,** isr**,** **&**ier**);**

iowrite32**(**ier**,** uart**->**port**.**membase **+** TIMBUART\_IER**);**

spin\_unlock**(&**uart**->**port**.**lock**);**

dev\_dbg**(**uart**->**port**.**dev**,** "%s leaving\n"**,** \_\_func\_\_**);**

**}**

static unsigned int timbuart\_tx\_empty**(**struct uart\_port **\***port**)**

**{**

u32 isr **=** ioread32**(**port**->**membase **+** TIMBUART\_ISR**);**

**return** **(**isr **&** TXBE**)** **?** TIOCSER\_TEMT **:** 0**;**

**}**

static unsigned int timbuart\_get\_mctrl**(**struct uart\_port **\***port**)**

**{**

u8 cts **=** ioread8**(**port**->**membase **+** TIMBUART\_CTRL**);**

dev\_dbg**(**port**->**dev**,** "%s - cts %x\n"**,** \_\_func\_\_**,** cts**);**

**if** **(**cts **&** TIMBUART\_CTRL\_CTS**)**

**return** TIOCM\_CTS **|** TIOCM\_DSR **|** TIOCM\_CAR**;**

**else**

**return** TIOCM\_DSR **|** TIOCM\_CAR**;**

**}**

static void timbuart\_set\_mctrl**(**struct uart\_port **\***port**,** unsigned int mctrl**)**

**{**

dev\_dbg**(**port**->**dev**,** "%s - %x\n"**,** \_\_func\_\_**,** mctrl**);**

**if** **(**mctrl **&** TIOCM\_RTS**)**

iowrite8**(**TIMBUART\_CTRL\_RTS**,** port**->**membase **+** TIMBUART\_CTRL**);**

**else**

iowrite8**(**TIMBUART\_CTRL\_RTS**,** port**->**membase **+** TIMBUART\_CTRL**);**

**}**

static void timbuart\_mctrl\_check**(**struct uart\_port **\***port**,** u32 isr**,** u32 **\***ier**)**

**{**

unsigned int cts**;**

**if** **(**isr **&** CTS\_DELTA**)** **{**

/\* ack \*/

iowrite32**(**CTS\_DELTA**,** port**->**membase **+** TIMBUART\_ISR**);**

cts **=** timbuart\_get\_mctrl**(**port**);**

uart\_handle\_cts\_change**(**port**,** cts **&** TIOCM\_CTS**);**

wake\_up\_interruptible**(&**port**->**state**->**port**.**delta\_msr\_wait**);**

**}**

**\***ier **|=** CTS\_DELTA**;**

**}**

static void timbuart\_enable\_ms**(**struct uart\_port **\***port**)**

**{**

/\* N/A \*/

**}**

static void timbuart\_break\_ctl**(**struct uart\_port **\***port**,** int ctl**)**

**{**

/\* N/A \*/

**}**

static int timbuart\_startup**(**struct uart\_port **\***port**)**

**{**

struct timbuart\_port **\***uart **=**

container\_of**(**port**,** struct timbuart\_port**,** port**);**

dev\_dbg**(**port**->**dev**,** "%s\n"**,** \_\_func\_\_**);**

iowrite8**(**TIMBUART\_CTRL\_FLSHRX**,** port**->**membase **+** TIMBUART\_CTRL**);**

iowrite32**(**0x1ff**,** port**->**membase **+** TIMBUART\_ISR**);**

/\* Enable all but TX interrupts \*/

iowrite32**(**RXBAF **|** RXBF **|** RXTT **|** CTS\_DELTA**,**

port**->**membase **+** TIMBUART\_IER**);**

**return** request\_irq**(**port**->**irq**,** timbuart\_handleinterrupt**,** IRQF\_SHARED**,**

"timb-uart"**,** uart**);**

**}**

static void timbuart\_shutdown**(**struct uart\_port **\***port**)**

**{**

struct timbuart\_port **\***uart **=**

container\_of**(**port**,** struct timbuart\_port**,** port**);**

dev\_dbg**(**port**->**dev**,** "%s\n"**,** \_\_func\_\_**);**

free\_irq**(**port**->**irq**,** uart**);**

iowrite32**(**0**,** port**->**membase **+** TIMBUART\_IER**);**

**}**

static int get\_bindex**(**int baud**)**

**{**

int i**;**

**for** **(**i **=** 0**;** i **<** ARRAY\_SIZE**(**baudrates**);** i**++)**

**if** **(**baud **<=** baudrates**[**i**])**

**return** i**;**

**return** **-**1**;**

**}**

static void timbuart\_set\_termios**(**struct uart\_port **\***port**,**

struct ktermios **\***termios**,**

struct ktermios **\***old**)**

**{**

unsigned int baud**;**

short bindex**;**

unsigned long flags**;**

baud **=** uart\_get\_baud\_rate**(**port**,** termios**,** old**,** 0**,** port**->**uartclk **/** 16**);**

bindex **=** get\_bindex**(**baud**);**

dev\_dbg**(**port**->**dev**,** "%s - bindex %d\n"**,** \_\_func\_\_**,** bindex**);**

**if** **(**bindex **<** 0**)**

bindex **=** 0**;**

baud **=** baudrates**[**bindex**];**

/\* The serial layer calls into this once with old = NULL when setting

up initially \*/

**if** **(**old**)**

tty\_termios\_copy\_hw**(**termios**,** old**);**

tty\_termios\_encode\_baud\_rate**(**termios**,** baud**,** baud**);**

spin\_lock\_irqsave**(&**port**->**lock**,** flags**);**

iowrite8**((**u8**)**bindex**,** port**->**membase **+** TIMBUART\_BAUDRATE**);**

uart\_update\_timeout**(**port**,** termios**->**c\_cflag**,** baud**);**

spin\_unlock\_irqrestore**(&**port**->**lock**,** flags**);**

**}**

static const char **\***timbuart\_type**(**struct uart\_port **\***port**)**

**{**

**return** port**->**type **==** PORT\_UNKNOWN **?** "timbuart" **:** **NULL;**

**}**

/\* We do not request/release mappings of the registers here,

\* currently it's done in the proble function.

\*/

static void timbuart\_release\_port**(**struct uart\_port **\***port**)**

**{**

struct platform\_device **\***pdev **=** to\_platform\_device**(**port**->**dev**);**

int size **=**

resource\_size**(**platform\_get\_resource**(**pdev**,** IORESOURCE\_MEM**,** 0**));**

**if** **(**port**->**flags **&** UPF\_IOREMAP**)** **{**

iounmap**(**port**->**membase**);**

port**->**membase **=** **NULL;**

**}**

release\_mem\_region**(**port**->**mapbase**,** size**);**

**}**

static int timbuart\_request\_port**(**struct uart\_port **\***port**)**

**{**

struct platform\_device **\***pdev **=** to\_platform\_device**(**port**->**dev**);**

int size **=**

resource\_size**(**platform\_get\_resource**(**pdev**,** IORESOURCE\_MEM**,** 0**));**

**if** **(!**request\_mem\_region**(**port**->**mapbase**,** size**,** "timb-uart"**))**

**return** **-**EBUSY**;**

**if** **(**port**->**flags **&** UPF\_IOREMAP**)** **{**

port**->**membase **=** ioremap**(**port**->**mapbase**,** size**);**

**if** **(**port**->**membase **==** **NULL)** **{**

release\_mem\_region**(**port**->**mapbase**,** size**);**

**return** **-**ENOMEM**;**

**}**

**}**

**return** 0**;**

**}**

static irqreturn\_t timbuart\_handleinterrupt**(**int irq**,** void **\***devid**)**

**{**

struct timbuart\_port **\***uart **=** **(**struct timbuart\_port **\*)**devid**;**

**if** **(**ioread8**(**uart**->**port**.**membase **+** TIMBUART\_IPR**))** **{**

uart**->**last\_ier **=** ioread32**(**uart**->**port**.**membase **+** TIMBUART\_IER**);**

/\* disable interrupts, the tasklet enables them again \*/

iowrite32**(**0**,** uart**->**port**.**membase **+** TIMBUART\_IER**);**

/\* fire off bottom half \*/

tasklet\_schedule**(&**uart**->**tasklet**);**

**return** IRQ\_HANDLED**;**

**}** **else**

**return** IRQ\_NONE**;**

**}**

/\*

\* Configure/autoconfigure the port.

\*/

static void timbuart\_config\_port**(**struct uart\_port **\***port**,** int flags**)**

**{**

**if** **(**flags **&** UART\_CONFIG\_TYPE**)** **{**

port**->**type **=** PORT\_TIMBUART**;**

timbuart\_request\_port**(**port**);**

**}**

**}**

static int timbuart\_verify\_port**(**struct uart\_port **\***port**,**

struct serial\_struct **\***ser**)**

**{**

/\* we don't want the core code to modify any port params \*/

**return** **-**EINVAL**;**

**}**

static struct uart\_ops timbuart\_ops **=** **{**

**.**tx\_empty **=** timbuart\_tx\_empty**,**

**.**set\_mctrl **=** timbuart\_set\_mctrl**,**

**.**get\_mctrl **=** timbuart\_get\_mctrl**,**

**.**stop\_tx **=** timbuart\_stop\_tx**,**

**.**start\_tx **=** timbuart\_start\_tx**,**

**.**flush\_buffer **=** timbuart\_flush\_buffer**,**

**.**stop\_rx **=** timbuart\_stop\_rx**,**

**.**enable\_ms **=** timbuart\_enable\_ms**,**

**.**break\_ctl **=** timbuart\_break\_ctl**,**

**.**startup **=** timbuart\_startup**,**

**.**shutdown **=** timbuart\_shutdown**,**

**.**set\_termios **=** timbuart\_set\_termios**,**

**.**type **=** timbuart\_type**,**

**.**release\_port **=** timbuart\_release\_port**,**

**.**request\_port **=** timbuart\_request\_port**,**

**.**config\_port **=** timbuart\_config\_port**,**

**.**verify\_port **=** timbuart\_verify\_port

**};**

static struct uart\_driver timbuart\_driver **=** **{**

**.**owner **=** THIS\_MODULE**,**

**.**driver\_name **=** "timberdale\_uart"**,**

**.**dev\_name **=** "ttyTU"**,**

**.**major **=** TIMBUART\_MAJOR**,**

**.**minor **=** TIMBUART\_MINOR**,**

**.**nr **=** 1

**};**

static int timbuart\_probe**(**struct platform\_device **\***dev**)**

**{**

int err**;**

struct timbuart\_port **\***uart**;**

struct resource **\***iomem**;**

dev\_dbg**(&**dev**->**dev**,** "%s\n"**,** \_\_func\_\_**);**

uart **=** kzalloc**(sizeof(\***uart**),** GFP\_KERNEL**);**

**if** **(!**uart**)** **{**

err **=** **-**EINVAL**;**

**goto** err\_mem**;**

**}**

uart**->**usedma **=** 0**;**

uart**->**port**.**uartclk **=** 3250000 **\*** 16**;**

uart**->**port**.**fifosize **=** TIMBUART\_FIFO\_SIZE**;**

uart**->**port**.**regshift **=** 2**;**

uart**->**port**.**iotype **=** UPIO\_MEM**;**

uart**->**port**.**ops **=** **&**timbuart\_ops**;**

uart**->**port**.**irq **=** 0**;**

uart**->**port**.**flags **=** UPF\_BOOT\_AUTOCONF **|** UPF\_IOREMAP**;**

uart**->**port**.**line **=** 0**;**

uart**->**port**.**dev **=** **&**dev**->**dev**;**

iomem **=** platform\_get\_resource**(**dev**,** IORESOURCE\_MEM**,** 0**);**

**if** **(!**iomem**)** **{**

err **=** **-**ENOMEM**;**

**goto** err\_register**;**

**}**

uart**->**port**.**mapbase **=** iomem**->**start**;**

uart**->**port**.**membase **=** **NULL;**

uart**->**port**.**irq **=** platform\_get\_irq**(**dev**,** 0**);**

**if** **(**uart**->**port**.**irq **<** 0**)** **{**

err **=** **-**EINVAL**;**

**goto** err\_register**;**

**}**

tasklet\_init**(&**uart**->**tasklet**,** timbuart\_tasklet**,** **(**unsigned long**)**uart**);**

err **=** uart\_register\_driver**(&**timbuart\_driver**);**

**if** **(**err**)**

**goto** err\_register**;**

err **=** uart\_add\_one\_port**(&**timbuart\_driver**,** **&**uart**->**port**);**

**if** **(**err**)**

**goto** err\_add\_port**;**

platform\_set\_drvdata**(**dev**,** uart**);**

**return** 0**;**

err\_add\_port**:**

uart\_unregister\_driver**(&**timbuart\_driver**);**

err\_register**:**

kfree**(**uart**);**

err\_mem**:**

printk**(**KERN\_ERR "timberdale: Failed to register Timberdale UART: %d\n"**,**

err**);**

**return** err**;**

**}**

static int timbuart\_remove**(**struct platform\_device **\***dev**)**

**{**

struct timbuart\_port **\***uart **=** platform\_get\_drvdata**(**dev**);**

tasklet\_kill**(&**uart**->**tasklet**);**

uart\_remove\_one\_port**(&**timbuart\_driver**,** **&**uart**->**port**);**

uart\_unregister\_driver**(&**timbuart\_driver**);**

kfree**(**uart**);**

**return** 0**;**

**}**

static struct platform\_driver timbuart\_platform\_driver **=** **{**

**.**driver **=** **{**

**.**name **=** "timb-uart"**,**

**.**owner **=** THIS\_MODULE**,**

**},**

**.**probe **=** timbuart\_probe**,**

**.**remove **=** timbuart\_remove**,**

**};**

/\*--------------------------------------------------------------------------\*/

static int \_\_init timbuart\_init**(**void**)**

**{**

**return** platform\_driver\_register**(&**timbuart\_platform\_driver**);**

**}**

static void \_\_exit timbuart\_exit**(**void**)**

**{**

platform\_driver\_unregister**(&**timbuart\_platform\_driver**);**

**}**

module\_init**(**timbuart\_init**);**

module\_exit**(**timbuart\_exit**);**

MODULE\_DESCRIPTION**(**"Timberdale UART driver"**);**

MODULE\_LICENSE**(**"GPL v2"**);**

MODULE\_ALIAS**(**"platform:timb-uart"**);**

# 专题04-网卡驱动程序设计

[主目录](#_目录)

## drivers/net/dm9000.c

static int \_\_init

dm9000\_init**(**void**)**

**{**

#if defined(CONFIG\_ARCH\_S3C2410)

unsigned int oldval\_bwscon **=** **\*(**volatile unsigned int **\*)**S3C2410\_BWSCON**;**

unsigned int oldval\_bankcon4 **=** **\*(**volatile unsigned int **\*)**S3C2410\_BANKCON4**;**

**\*((**volatile unsigned int **\*)**S3C2410\_BWSCON**)** **=**

**(**oldval\_bwscon **&** **~(**3**<<**16**))** **|** S3C2410\_BWSCON\_DW4\_16 **|** S3C2410\_BWSCON\_WS4 **|** S3C2410\_BWSCON\_ST4**;**

**\*((**volatile unsigned int **\*)**S3C2410\_BANKCON4**)** **=** 0x1f7c**;**

#endif

printk**(**KERN\_INFO "%s Ethernet Driver, V%s\n"**,** CARDNAME**,** DRV\_VERSION**);**

**return** platform\_driver\_register**(&**dm9000\_driver**);**

**}**

static void \_\_exit

dm9000\_cleanup**(**void**)**

**{**

platform\_driver\_unregister**(&**dm9000\_driver**);**

**}**

module\_init**(**dm9000\_init**);**

module\_exit**(**dm9000\_cleanup**);**

MODULE\_AUTHOR**(**"Sascha Hauer, Ben Dooks"**);**

MODULE\_DESCRIPTION**(**"Davicom DM9000 network driver"**);**

MODULE\_LICENSE**(**"GPL"**);**

MODULE\_ALIAS**(**"platform:dm9000"**);**

# 专题05-触摸屏驱动程序设计

[主目录](#_目录)

## drivers/input/touchscreen/usbtouchscreen.c

# 专题06-LCD驱动程序设计

[主目录](#_目录)

# 专题07-FLASH驱动程序设计

[主目录](#_目录)

# 专题08-I2C驱动程序设计

[主目录](#_目录)

浅谈I2C总线：<http://www.51hei.com/mcu/4012.html>

# 专题09-SPI驱动程序设计

[主目录](#_目录)

浅谈SPI总线：<http://www.51hei.com/mcu/4011.html>

**SPI总线概述**

SPI全称是串行外设接口（Serial Peripheral Interface），是由Motorola提出的一种全双工同步串行通信接口，通信波特率可以高达5Mbps，但具体速度大小取决于SPI硬件。SPI接口具有全双工操作，操作简单，数据传输速率较高的优点，但也存在没有指定的流控制，没有应答机制确认是否接收到数据的缺点。

**SPI总线的构成及信号类型**

SPI总线只需四条线（如图1所示）就可以完成MCU与各种外围器件的通讯:

1）MOSI – Master数据输出,Slave数据输入

2）MISO – Master数据输入,Slave数据输出

3）SCK – 时钟信号,由Master产生

4）/CS – Slave使能信号,由Master控制。

# 专题10-USB驱动程序设计

[主目录](#_目录)

[主目录](#_目录)

## 第5课-USB下载线驱动设计

[主目录](#_目录)

## USB下载驱动dnw的移植

//dnw\_usb.c的修改

//lsusb check the usb device code and add the code to the array

static struct usb\_device\_id dnw\_id\_table [] = {

{ USB\_DEVICE(0x04E8, 0x1234) },

{ USB\_DEVICE(0x5345, 0x1234) },

{ }

};

//Makefile的修改

ifneq ($(KERNELRELEASE),)

obj-m := dnw\_usb.o

else

KDIR := /lib/modules/$(shell uname -r)/build

all:

make -C $(KDIR) M=$(PWD) modules

clean:

rm -f \*.ko \*.o \*.mod.o \*.mod.c \*.symvers \*.order \*.unsigned \*.cmd

endif

## insdnw.sh

cwd**=**"/home/redhat6/dnw"

**echo** ".ko path: ${cwd}"

**lsmod** **|** **grep** "dnw\_usb" 2**>/**dev**/**null 1**>/**dev**/**null

**if** **[** **$?** **-**eq 0 **];**

**then**

**rmmod** dnw\_usb

**insmod** **/**home**/**redhat6**/**dnw**/**dnw\_usb.ko

**else**

**insmod** **/**home**/**redhat6**/**dnw**/**dnw\_usb.ko

**fi**

## dnw\_usb.c

#include <linux/module.h>

#include <linux/kernel.h>

#include <linux/usb.h>

#include <linux/fs.h>

#include <linux/uaccess.h>

#include <linux/slab.h>

#define BULKOUT\_BUFFER\_SIZE 512

char **\***bulkout\_buffer**;**

struct usb\_device **\***udev**;**

\_\_u8 bulk\_out\_endaddr**;**

//lsusb -->

static struct usb\_device\_id dnw\_id\_table **[]** **=** **{**

**{** USB\_DEVICE**(**0x5345**,** 0x1234**)** **},**

**{** **}**

**};**

static int dnw\_open**(**struct inode**\*** inode**,** struct file **\***file**)**

**{**

bulkout\_buffer **=** kmalloc**(**BULKOUT\_BUFFER\_SIZE**,**GFP\_KERNEL**);**

**return** 0**;**

**}**

static int dnw\_release **(**struct inode**\*** inode**,** struct file **\***file**)**

**{**

kfree**(**bulkout\_buffer**);**

**return** 0**;**

**}**

static ssize\_t dnw\_write**(**struct file **\***file**,** const char \_\_user **\***buf**,** size\_t len**,** loff\_t **\***pos**)**

**{**

size\_t to\_write**;**

size\_t total\_write **=** 0**;**

size\_t act\_len**;**

**while(**len**>**0**)**

**{**

to\_write **=** min**(**len**,(**size\_t**)**BULKOUT\_BUFFER\_SIZE**);**

copy\_from\_user**(**bulkout\_buffer**,**buf**+**total\_write**,**to\_write**);**

usb\_bulk\_msg**(**udev**,**usb\_sndbulkpipe**(**udev**,**bulk\_out\_endaddr**),**bulkout\_buffer**,**to\_write**,&**act\_len**,**3**\***HZ**);**

len **-=** to\_write**;**

total\_write **+=** to\_write**;**

**}**

**return** total\_write**;**

**}**

static struct file\_operations dnw\_ops **=**

**{**

**.**owner **=** THIS\_MODULE**,**

**.**write **=** dnw\_write**,**

**.**open **=** dnw\_open**,**

**.**release **=** dnw\_release**,**

**};**

static struct usb\_class\_driver dnw\_class **=** **{**

**.**name **=** "secbulk%d"**,**

**.**fops **=** **&**dnw\_ops**,**

**.**minor\_base **=** 100**,**

**};**

static int dnw\_probe**(**struct usb\_interface **\***intf**,** const struct usb\_device\_id **\***id**)**

**{**

/\* 接口设置描述 \*/

struct usb\_host\_interface **\***interface**;**

struct usb\_endpoint\_descriptor **\***endpoint**;**

int i**;**

interface **=** intf**->**cur\_altsetting**;**

**for(**i**=**0**;**i**<**interface**->**desc**.**bNumEndpoints**;**i**++)**

**{**

endpoint **=** **&**interface**->**endpoint**[**i**].**desc**;**

**if(**usb\_endpoint\_is\_bulk\_out**(**endpoint**))**

**{**

bulk\_out\_endaddr **=** endpoint**->**bEndpointAddress**;**

**break;**

**}**

**}**

usb\_register\_dev**(**intf**,&**dnw\_class**);**

udev **=** usb\_get\_dev**(**interface\_to\_usbdev**(**intf**));**

**}**

static void dnw\_disconnect**(**struct usb\_interface **\***intf**)**

**{**

usb\_deregister\_dev**(**intf**,&**dnw\_class**);**

**}**

struct usb\_driver dnw\_driver **=** **{**

**.**name **=** "dnw"**,** /\* 驱动名 \*/

**.**probe **=** dnw\_probe**,** /\* 捕获函数 \*/

**.**disconnect **=** dnw\_disconnect**,** /\* 卸载函数 \*/

**.**id\_table **=** dnw\_id\_table**,** /\* 设备列表 \*/

**};**

int dnw\_init**()**

**{**

usb\_register**(&**dnw\_driver**);**

**return** 0**;**

**}**

void dnw\_exit**()**

**{**

usb\_deregister**(&**dnw\_driver**);**

**}**

module\_init**(**dnw\_init**);**

module\_exit**(**dnw\_exit**);**

MODULE\_LICENSE**(**"GPL"**);**

## dnw.c

#include <stdio.h>

#include <stdlib.h>

#include <malloc.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>

#include <fcntl.h>

#include <string.h>

const char**\*** dev **=** "/dev/dnw0"**;**

int main**(**int argc**,** char**\*** argv**[])**

**{**

unsigned char**\*** file\_buffer **=** **NULL;**

long int addr **=** 0**;**

**if(** 3 **!=** argc **)** **{**

printf**(**"Usage: dwn <filename> address\n"**);**

**return** 1**;**

**}**

int fd **=** open**(**argv**[**1**],** O\_RDONLY**);**

**if(-**1 **==** fd**)** **{**

printf**(**"Can not open file - %s\n"**,** argv**[**1**]);**

**return** 1**;**

**}**

addr **=** strtol**((**char **\*)** argv**[**2**]** **,NULL,** 16**);**

printf**(**"addr = %x \n"**,** addr**);**

// get file size

struct stat file\_stat**;**

**if(** **-**1 **==** fstat**(**fd**,** **&**file\_stat**)** **)** **{**

printf**(**"Get file size filed!\n"**);**

**return** 1**;**

**}**

file\_buffer **=** **(**unsigned char**\*)**malloc**(**file\_stat**.**st\_size**+**10**);**

**if(NULL** **==** file\_buffer**)** **{**

printf**(**"malloc failed!\n"**);**

**goto** error**;**

**}**

//memset(file\_buffer, '\0', sizeof(file\_buffer)); // bad code ! corrected by Qulory

memset**(**file\_buffer**,** '\0'**,** **sizeof(**char**)\*(**file\_stat**.**st\_size**+**10**));**

// the first 8 bytes in the file\_buffer is reserved, the last 2 bytes also;

**if(** file\_stat**.**st\_size **!=** read**(**fd**,** file\_buffer**+**8**,** file\_stat**.**st\_size**))** **{**

printf**(**"Read file failed!\n"**);**

**goto** error**;**

**}**

printf**(**"File name : %s\n"**,** argv**[**1**]);**

printf**(**"File size : %ld bytes\n"**,** file\_stat**.**st\_size**);**// off\_t is long int

int fd\_dev **=** open**(**dev**,** O\_WRONLY**);**

**if(** **-**1 **==** fd\_dev**)** **{**

printf**(**"Can not open %s\n"**,** dev**);**

**goto** error**;**

**}**

/\*

\* Note: the first 4 bytes store the dest addr ;

\* the following 4 bytes store the file size ;

\* and the last 2 bytes store the sum of each bytes of the file ;

\*/

**\*((**unsigned long**\*)**file\_buffer**)** **=** addr**;** //load address

**\*((**unsigned long**\*)**file\_buffer**+**1**)** **=** file\_stat**.**st\_size**+**10**;** //file size

unsigned short sum **=** 0**;**

int i**;**

**for(**i**=**8**;** i**<**file\_stat**.**st\_size**+**8**;** i**++)** **{**

sum **+=** file\_buffer**[**i**];**

**}**

**\*((**unsigned short**\*)(**file\_buffer**+**8**+**file\_stat**.**st\_size**))** **=** sum**;**

printf**(**"Start Sending data...\n"**);**

size\_t remain\_size **=** file\_stat**.**st\_size**+**10**;**

size\_t block\_size **=** 512**;**

size\_t written **=** 0**;**

**while(**remain\_size **>** 0**)** **{**

size\_t to\_write **=** remain\_size **>** block\_size **?** block\_size**:**remain\_size**;**

size\_t real\_write **=** write**(**fd\_dev**,** file\_buffer**+**written**,** to\_write**);**

**if(** to\_write **!=** real\_write**)** **{**

printf**(**" write /dev/secbulk0 failed! to\_write = %u real\_write = %u \n" **,** to\_write **,**real\_write **);**

**return** 1**;**

**}**

remain\_size **-=** to\_write**;**

written **+=** to\_write**;**

printf**(**"\rSent %lu%% \t %u bytes !"**,** written**\***100**/(**file\_stat**.**st\_size**+**10**),** written**);**

fflush**(**stdout**);**

**}**

printf**(**"OK\n"**);**

**return** 0**;**

error**:**

**if(-**1 **!=** fd\_dev**)** **{**

close**(**fd\_dev**);**

**}**

**if(**fd **!=** **-**1**)** **{**

close**(**fd**);**

**}**

**if(** **NULL** **!=** file\_buffer **)** **{**

free**(**file\_buffer**);**

**}**

**return** **-**1**;**

**}**

# 专题11-CAN总线驱动程序设计

浅谈CAN总线：<http://blog.csdn.net/lu_embedded/article/details/51556040>

CAN总线-电子发烧友：<http://www.elecfans.com/tags/CAN总线/>

# 专题\*\*-Linux总线驱动总体框架总结

## I2C总线驱动核心模型

static int \_\_init ad7877\_init**(**void**)**

**{**

**return** spi\_register\_driver**(&**ad7877\_driver**);**

**}**

module\_init**(**ad7877\_init**);**

static void \_\_exit ad7877\_exit**(**void**)**

**{**

spi\_unregister\_driver**(&**ad7877\_driver**);**

**}**

module\_exit**(**ad7877\_exit**);**

MODULE\_AUTHOR**(**"Michael Hennerich <hennerich@blackfin.uclinux.org>"**);**

MODULE\_DESCRIPTION**(**"AD7877 touchscreen Driver"**);**

MODULE\_LICENSE**(**"GPL"**);**

MODULE\_ALIAS**(**"spi:ad7877"**);**

## SPI总线驱动核心模型

static int \_\_init ad7879\_init**(**void**)**

**{**

**return** i2c\_add\_driver**(&**ad7879\_driver**);**

**}**

module\_init**(**ad7879\_init**);**

static void \_\_exit ad7879\_exit**(**void**)**

**{**

i2c\_del\_driver**(&**ad7879\_driver**);**

**}**

module\_exit**(**ad7879\_exit**);**

#endif

MODULE\_AUTHOR**(**"Michael Hennerich <hennerich@blackfin.uclinux.org>"**);**

MODULE\_DESCRIPTION**(**"AD7879(-1) touchscreen Driver"**);**

MODULE\_LICENSE**(**"GPL"**);**

MODULE\_ALIAS**(**"spi:ad7879"**);**

## platform总线驱动核心模型

static int \_\_init da9034\_touch\_init**(**void**)**

**{**

**return** platform\_driver\_register**(&**da9034\_touch\_driver**);**

**}**

module\_init**(**da9034\_touch\_init**);**

static void \_\_exit da9034\_touch\_exit**(**void**)**

**{**

platform\_driver\_unregister**(&**da9034\_touch\_driver**);**

**}**

module\_exit**(**da9034\_touch\_exit**);**

MODULE\_DESCRIPTION**(**"Touchscreen driver for Dialog Semiconductor DA9034"**);**

MODULE\_AUTHOR**(**"Eric Miao <eric.miao@marvell.com>, Bin Yang <bin.yang@marvell.com>"**);**

MODULE\_LICENSE**(**"GPL"**);**

MODULE\_ALIAS**(**"platform:da9034-touch"**);**

## USB总线驱动核心模型

static int \_\_init usbtouch\_init**(**void**)**

**{**

**return** usb\_register**(&**usbtouch\_driver**);**

**}**

static void \_\_exit usbtouch\_cleanup**(**void**)**

**{**

usb\_deregister**(&**usbtouch\_driver**);**

**}**

module\_init**(**usbtouch\_init**);**

module\_exit**(**usbtouch\_cleanup**);**

MODULE\_AUTHOR**(**DRIVER\_AUTHOR**);**

MODULE\_DESCRIPTION**(**DRIVER\_DESC**);**

MODULE\_LICENSE**(**"GPL"**);**

MODULE\_ALIAS**(**"touchkitusb"**);**

MODULE\_ALIAS**(**"itmtouch"**);**

MODULE\_ALIAS**(**"mtouchusb"**);**

## Serial I/O总线驱动核心模型

static int \_\_init tw\_init**(**void**)**

**{**

**return** serio\_register\_driver**(&**tw\_drv**);**

**}**

static void \_\_exit tw\_exit**(**void**)**

**{**

serio\_unregister\_driver**(&**tw\_drv**);**

**}**

module\_init**(**tw\_init**);**

module\_exit**(**tw\_exit**);**

# 第五季-小试牛刀-项目开发

# 阶段1-岗前培训(1~3m)

[主目录](#_目录)

## 源码包的配置编译与安装

解压进入源码顶层目录(一般会有一个README文件或INSTALL文件)，配置细节和安装细节作者一般都会在里面注明。

Configure 参数一般有：--prefix=dir, --disable-shared, --disable-static, --host=arm-linux, CC=’gcc’

## 库移植的源码包配置编译与安装

编译静态库

# ./configure --prefix=/home/redhat6/jpeg-9/\_install--disable-shared--host=arm-linux

# vim Makefile 修改Makefile文件的相关编译器

# make

# make install

编译动态库

# ./configure --prefix=/home/redhat6/jpeg-9/\_install --disable-static --host=arm-linux

# vim Makefile 修改Makefile文件的相关编译器

# make

# make install

## 敏捷项目管理模型

用户需求

需求分析:需求文档

架构设计:架构设计文档

详细设计

编码

测试

瀑布管理模型范例:远程监控系统

用户需求

需求分析:图像采集功能,图像传输功能,图像显示功能

架构设计:摄像头-

摄像头驱动-详细设计-ioctl接口等等

V4L2接口-

图像采集模块-

图像压缩模块-

图像传输模块-

图像解压缩模块-

图像显示模块-

瀑布模型致命缺点:用户需求不清晰,导致修改时候会需要很大的改动,严重影响开发效率。

敏捷项目管理模型

Scrum管理模型：

需求池(确定部分,模糊部分)-

第1个迭代先把明确的需求设计编码测试出来-输出(将输出展示给用户,用户提出新需求)-

第2个迭代把新明确需求设计编码测试出来-输出(将输出展示给用户,用户提出新需求)-

同理:经过多个迭代开发逐步明确客户需求并完成产品开发

每日环节-scrum管理模型(把需求细化,用每一个迭代去实现每个确定的需求)

## 分布式版本控制系统Git

工具集通过:**分布式版本控制系统Git相关**找到相应目录

Git是一个开源的分布式版本控制系统，用以有效、高速的处理从很小到非常大的项目版本管理。

**开源中国 Git 代码托管平台**：<http://git.oschina.net/>

# 嵌入式项目：移动物体监控系统

[主目录](#_目录)

Leangoo敏捷项目管理网站

USB摄像头:

功能1-图像采集

功能2-移动图像判断

功能3-语音报警子系统开发(声卡驱动开发, 移植嵌入式播放器)

功能4-具备小型web服务器

## 声卡驱动开发

OSS架构

OSS全称是Open Sound System，叫做开放式音频系统，这种早期的音频系统这种基于文件系统的访问方式，这意味着对声音的操作完全可以像对普通文件那样执行open, read等操作。OSS中，主要提供了以下几种音频设备的抽象设备文件：/dev/mixer用来访问声卡中的混音器用于调整音量大小和选择音源，/dev/dsp、/dev/audio读这个设备就相当于录音，写这个设备就相当于放音。

ALSA架构

ALSA架构:由于OSS设计上的缺陷,导致其对混音的支持不好,再加上2002年以后,OSS成为商业不开源软件,这就催生了Linux下另一种音频系统ALSA的出现,全称Advanced Linux Sound Architecture, 叫做Linux系统高级音频架构,它主要为声卡提供的驱动组件,以替代原先的OSS,由于很多优秀播放器原先都是基于OSS架构开发的,所以它得向下兼容OSS。

ALSA架构借助如下设备文件工作:

/dev/pcmC0D0c:用于录音的pcm设备

/dev/pcmC0D0p:用于播放的pcm设备

/dev/timer:定时器

/dev/controlC0:用于声卡的控制,如通道选择

/dev/mixer:混音处理

配置Linux内核使能声卡驱动

使能声卡驱动:配置Linux内核,重新编译内核

Device Drivers >>>

Sound card support >>>

Advanced Linux Sound Architecture >>>

<\*> OSS Mixer API

<\*> OSS PCM (digital audio) API

<\*> OSS PCM xxx plugin system

ALSA for SoC audio support >>>(片上声卡支持)

Soc I2S Audio support for WM8960 on TQ210

重启开发板检查驱动设备文件

# ls /dev/dsp 如果发现没有这个文件

# ls /dev/mixer 如果发现没这个文件

这个是时候怎么办呢？其实这个现象是Linux内核并没有把该驱动编译进内核,而是以独立的ko文件提供的.

开发板光盘/Linux资源/Linux源码包/根文件系统目录/.../lib/modules/../kernel/drivers/sound/

多个模块安装的依赖问题:排除法来按顺序安装驱动模块文件

出现依赖问题会出错:xxx:unknown symbol xxx; cannot insert xxx

再来检查开发板下的相关设备文件: ls /dev/dsp && ls /dev/mixer

声卡驱动使能成功,声卡驱动开发完成！

## 嵌入式播放器移植

第一步:网上搜索关键字:嵌入式Linux播放器:mp3播放器madplay的移植

第二步:下载相应源码包（摘自）

zlib-1.2.3.tar.gz, libid3tag-0.15.1b.tar.gz, libmad-0.15.1b.tar.gz

madplay-0.15.2b.tar.gz

第三步:编译相应库:解压进入到zlib库目录,配置,修改Makefile,编译,安装

配置:./configure --prefix=${PWD}/\_install--shared

修改Makefile: CC=arm-linux-gcc,把关键字为单独gcc的改为arm-linux-gcc

AR=arm-linux-ar,把关键字为单独ar的改为arm-linux-ar

RANLIB=arm-linux-ranlib,把关键字为ranlib等工具链都改为交叉工具链

编译和安装: make && make install

查看里面的库: ls ${PWD}/\_install/lib

第四步:编译相应库:libid3tag-xxx.tgz:配置,修改Makefile,编译,安装

配置:./configure --host=arm-linux --prefix=${PWD}/\_install --disable-static

编译和安装: make && make install

查看安装好的库: ls ${PWD}/\_install/lib

第五步:编译相应库:libmad-xxx.tgz:配置,修改,Makefile,编译,安装

配置:./configure --host=arm-linux --prefix=${PWD}/\_install --disable-static

编译和安装: make && make install

编译出错:编译参数选项中有不支持的编译选项,修改Makefile中去掉该选项并重新编译安装。

查看安装好的库: ls ${PWD}/\_install/lib

第六步:编译该播放器:

配置:./configure --host=arm-linux --prefix=${PWD}/\_install --disable-static

编译和安装: make && make install

查看安装好的: ls ${PWD}/\_install/madplay

拷贝\_install/madplay到开发板

在开发板运行madplay: {argv[0]} not found, arm-linux-readelf –d mdplay,拷贝相应库。

另外的标准库位置: /usr/local/arm/4.3.2/arm-none-linux-gnueabi里面寻找相应的嵌入式标准库

[root@redhat6 lib]# pwd

/usr/local/arm/4.3.2/arm-none-linux-gnueabi/libc/lib

[root@redhat6 lib]#

并播放一首mp3文件

出现: audio: /dev/dsp: No such file or directory

安装声卡驱动: insmod wm8960.ko && insmod s5pv210\_hdmi.ko

再来测试软件并播放一首mp3测试文件

好了，我们的嵌入式播放器移植成功完成！

## 摄像头驱动开发

摄像头软件系统架构(接口的统一:移植性)

应用程序-----》V4L2核心-----》支持V4L2的摄像头驱动-----》摄像头(树状架构(分支点:V4L2核心))

配置Linux内核使能摄像头驱动并重新编译内核

Device Drivers >>>

multimedia support >>>(多媒体支持)

<\*> Video For Linux

Video capture adapters >>>

V4L USB devices >>>

<\*> GSPCA based webcams >>>

<\*> ZC3XX USB Camera Driver

重启开发板,插上USB摄像头,会看到成功的加载USB摄像头提示信息

测试摄像头驱动:camera.c

# arm-linux-gcc camera.c –o camera

复制测试程序到开发板并复制一张测试图片test.jpg到开发板

重启开发板,可能会提示大量崩溃出错信息,从第一处错误开始:ehci出错,在内核配置中把ehci改成ohci

Device Drivers >>>

USB support >>>

<> EHCI HCD support(取消EHCI)

<\*> OHCI HCD support(勾选OHCI)

开发板上测试:

# ./camera 运行成功后会自动抓取图像并保存在当前目录下的test.jpg中

## V4L2编程接口深度学习

### 参考资料:

Linux之V4L2基础编程:硬盘手册/资料书/GQ-资料/ Linux之V4L2基础编程.docx

<http://www.cnblogs.com/emouse/archive/2013/03/04/2943243.html>

和菜鸟一起学Linux之V4L2摄像头流程:硬盘手册/资料书/GQ-资料/和菜鸟一起学Linux之V4L2摄像头流程.docx

<http://blog.csdn.net/eastmoon502136/article/details/8190262>

### 分析camera.c测试程序

找到main函数

//1 打开设备文件:O\_NONBLOCK

fd = open (dev\_name, O\_RDWR | O\_NONBLOCK, 0);

//2 获取驱动信息:VIDIOC\_QUERYCAP

ioctl (fd, VIDIOC\_QUERYCAP, &cap);

//3 设置图像格式:VIDIOC\_S\_FMT

ioctl (fd, VIDIOC\_S\_FMT, &fmt) ;

驱动工作流程: (in)-(in)-(in)-(in)(帧缓冲队列) ---》驱动(帧缓冲:用来缓存一帧数据的) 》output(软件)

应用程序流程: 取出帧缓冲队列 》》读取 》》 放回帧缓冲队列

//4 申请图像缓冲区:VIDIOC\_REQBUFS

req.count = 4;

req.type = V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;

req.memory = V4L2\_MEMORY\_MMAP

ioctl (fd, VIDIOC\_REQBUFS, &req);

//5 获取帧缓冲地址长度信息:VIDIOC\_QUERYBUF

buf.type = V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;

buf.memory = V4L2\_MEMORY\_MMAP;

buf.index = n\_buffers;

ioctl (fd, VIDIOC\_QUERYBUF, &buf);

//6 把内核空间的帧缓冲映射到用户空间:mmap()

buffers[n\_buffers].start = mmap (NULL , //通过mmap建立映射关系

buf.length,

PROT\_READ | PROT\_WRITE ,

MAP\_SHARED ,

fd,

buf.m.offset);

}

//7 将4个帧缓冲放入队列:VIDIOC\_QBUF

for (i = 0; i < n\_buffers; ++i)

{

buf.type = V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;

buf.memory = V4L2\_MEMORY\_MMAP;

buf.index = i;

ioctl (fd, VIDIOC\_QBUF, &buf);

}

//8 开始采集图像

//开始捕捉图像数据:VIDIOC\_STREAMON

type = V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;

ioctl (fd, VIDIOC\_STREAMON, &type);

//通过select()判断是否有数据

select(fd + 1, &fds, NULL, NULL, NULL);

//9 取出帧缓冲(帧缓冲出队列):VIDIOC\_DQBUF

/\*帧出列\*/

buf.type = V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;

buf.memory = V4L2\_MEMORY\_MMAP;

ioctl (fd, VIDIOC\_DQBUF, &buf);

//10 访问数据并将读出的数据写入test.jpg文件:write()

write(file\_fd,buffers[buf.index].start,buffers[buf.index].length);

//11 帧缓冲入队列:VIDIOC\_QBUF

/\*buf入列\*/

ioctl(fd, VIDIOC\_QBUF, &buf);

//13取消了映射:munmap()

for (i = 0; i < n\_buffers; ++i)

munmap (buffers[i].start, buffers[i].length);

//14 关闭相应设备文件:close()

# 嵌入式项目：网络安全传输系统

[主目录](#_目录)

网络安全传输系统

[主目录](#_目录)

[主目录](#_目录)

公钥/密钥：有两个数字，一个是1，一个是2。我把2这个数字保留起来，不告诉大家，然后我告诉大家，1是我的公钥。我的好友甲用我的公钥1加密了文件a，放在网上。别人偷了这个文件，但是别人解不开，因为别人不知道2是我的私钥，只有我才能解密，解密后得到a。这样，甲就可以传送加密的数据给我了。

# 嵌入式项目：基于视频压缩的实时监控系统

# 嵌入式项目：基于ARM9的广告投放终端项目

# 嵌入式项目：WEB网络MP3终端项目

# 嵌入式项目：GPS电子地图项目

# 嵌入式项目：手机远程的智能家具终端项目

基于视频压缩的实时监控系统

[主目录](#_目录)

# 第六季-疑难汇总

## 【安装源码包】

其实，在linux下面安装一个源码包是最常用的，笔者在日常的管理工作中，大部分软件都是通过源码安装的。安装一个源码包，是需要我们自己把源代码编译成二进制的可执行文件。如果你读得懂这些源代码，那么你就可以去修改这些源代码自定义功能，然后再去编译成你想要的。使用源码包的好处除了可以自定义修改源代码外还可以定制相关的功能，因为源码包在编译的时候是可以附加额外的选项的。

源码包的编译用到了linux系统里的编译器，常见的源码包一般都是用C语言开发的，这也是因为C语言为linux上最标准的程序语言。Linux上的C语言编译器叫做gcc，利用它就可以把C语言变成可执行的二进制文件。所以如果你的机器上没有安装gcc就没有办法去编译源码。你可以使用 yum install -y gcc 来完成安装。

安装一个源码包，通常需要三个步骤：

1. ./config 在这一步可以定制功能，加上相应的选项即可，具有有什么选项可以通过”./config --help ”命令来查看。在这一步会自动检测你的linux系统与相关的套件是否有编译该源码包时需要的库，因为一旦缺少某个库就不能完成编译。只有检测通过后才会生成一个Makefile文件。

2. make 使用这个命令会根据Makefile文件中预设的参数进行编译，这一步其实就是gcc在工作了。

3. make install 安装步骤，生成相关的软件存放目录和配置文件的过程。

上面介绍的3步并不是所有的源码包软件都一样的，笔者以前也曾经遇到过，安装步骤并不是这样，也就是说源码包的安装并非具有一定的标准安装步骤。这就需要你拿到源码包解压后，然后进入到目录找相关的帮助文档，通常会以INSTALL或者README为文件名。所以，你一定要去看一下。下面笔者会编译安装一个源码包来帮你更深刻的去理解如何安装源码包。

1. 下载一个源码包

cd /usr/local/src

wget <http://sysxxxxxx/httpd-2.2.16.tar.gz>

这里要提一下，建议以后你把所有下载的源码包放到/usr/local/src/目录下，这个并不是必须的，只是一个约定。方便你和你的同事将来更好的去运维这台服务器。

2. 解压源码包

一般的源码包都是一个压缩包，如何解压一个.tar.gz的包上一章讲过的。

3. 配置相关的选项，并生成Makefile

使用./config --help 可以查看可用的选项。一般常用的有”--prefix=PREFIX “ 这个选项的意思是定义软件包安装到哪里。到这里，笔者再提一个小小的约定，通常源码包都是安装在/usr/local/目录下的。比如，我们把apache安装在/usr/local/apache2下，那么这里就应该这样写” --prefix=/usr/local/apache2”。其他还有好多选项，如果你有耐心你可以挨个去看一看都有什么作用。

等check结束后生成了Makefile文件

除了查看有没有生成Makefile文件来判定有没有完成./config 的操作外，还可以通过这个命令”echo $?”来判定，如果是0，则表示上一步操作成功完成，否则就是没有成功。

4. 进行编译

这一步操作，就是把源代码编译成二进制的可执行文件，这一步也是最漫长的一步，编译时间的长短取决于源代码的多少和机器配置。

5. 安装

在安装前，先确认上一步操作是否成功完成。

make install 会创建相应的目录以及文件。当完成安装后，会在/usr/local目录下多了一个apache2目录，这就是apache所安装的目录了。

其实在日常的源码安装工作中，并不是每个都像笔者这样顺利完成安装的，遇到错误不能完成安装的情况是很多的。通常都是因为缺少某一个库文件导致的。这就需要你仔细琢磨报错信息或者查看当前目录下的config.log去得到相关的信息。另外，如果自己不能解决那就去网上google一下吧，通常你会得到你想要的答案。

## 在VIM里调用多种浏览器预览html php 等文件

最近这段时间睡眠严重不足，脑袋发晕的时候就要放送下，so改善vim。产生个想法：把自己喜欢的editplus的功能都搞到vim上面来。这个就是其中之一：在浏览器中预览当前文件。有时间的话，可能要写点从editplus转型到VIM的东西。

优点1：  
囊括了主要的浏览器：chrome、firefox、oprea、ie、ietester（随自己喜欢可以增加更多，比如safari），浏览器的简称：cr、ff、op、ie、ie6、ie7、ie8、ie9、iea，简称、路径以键值对方式都保存到browsers中。  
上面的简称大部分人应该都明白了，ie就是系统默认的ie，最后一个iea是在ietester中使用所有版本的ie同时预览（IE5.5-IE9）。

优点2：  
本地文件自动以file://开头或http://开头的两种方式预览。如果文件在htdocs就用http方式打开，否则用file方式。  
file://开头的地址预览html一般没问题，但是预览php或aspx等就显得苍白无力了。这里可以设置一个htdocs/wwwroot的文件夹地址，然后预览的时候匹配文件是否在这个文件夹内(支持子目录)，如果在就用http://方式打开，否则就用file://方式。

无论前端开发者或者程序员都及其适合。自夸完毕，下面说下使用：  
在\_vimrc中加入下面的代码，然后按F4+cr - 在chrome预览，F4+ff  - 在firefox下预览……。方式就是F4+浏览器简称（应该比用F4+1234的数字形式便于记忆）。当然，这个完全可以自己diy的。

下面fuc里面的浏览器地址需要自己修改，我的文件夹目录和你的可能是有不同的。还有htdocs文件夹、本地的预览的端口号，我使用的是8090.

## 第十章文档的压缩与打包

Linux入门教程：<http://www.92csz.com/study/linux/>

.gz gzip 压缩工具压缩的文件

.bz2 bzip2 压缩工具压缩的文件

.tar tar 打包程序打包的文件(tar并没有压缩功能，只是把一个目录合并成一个文件)

.tar.gz 可以理解为先用tar打包，然后再gzip压缩

.tar.bz2 同上，先用tar打包，然后再bzip2压缩

**【gzip】**

语法： gzip [-d#] filename 其中#为1-9的数字

-d ：解压缩时使用

-# ：压缩等级，1压缩最差，9压缩最好，6为默认

用-d解压缩

要注意的是，gzip不可以压缩目录

**【bzip2】**

语法：bzip2 [-dz] filename

-d ：解压缩

-z ：压缩

其实-z参数是可以省略掉的，你不妨试试,跟gzip的解压类似，也是用-d解压。

**【tar】**

语法：tar [-zjxcvfpP] filename

-z ：是否同时用gzip压缩

-j ：是否同时用bzip2压缩

-x ：解包或者解压缩

-t ：查看tar包里面的文件

-c ：建立一个tar包或者压缩文件包

-v ：可视化

-f ：后面跟文件名，压缩时跟-f文件名，意思是压缩后的文件名为filename，解压时跟-f文件名，意思是解压filename。请注意，如果是多个参数组合的情况下带有-f，请把f写到最后面。

-p ：使用原文件的属性，压缩前什么属性压缩后还什么属性。（不常用）

-P ：可以使用绝对路径。（不常用）

--exclude filename ：在打包或者压缩时，不要将filename文件包括在内。（不常用）

首先在test目录下建立test111目录，然后在test111目录下建立test2.txt，并写入”nihao”到test2.txt中，接着是用tar把test111打包成test111.tar。请记住-f参数后跟的是打包后的文件名。

--exclude参数的作用就是打包的时候过滤掉某些文件，如果想过滤多个文件怎么办

只能是继续跟 --exclude filename了。

## 第九章-vim编辑器基础操作表

|  |
| --- |
| **一般模式下移动光标** |

|  |  |
| --- | --- |
| h或向左方向键 | 光标向左移动一个字符 |
| j或者向下方向键 | 光标向下移动一个字符 |
| K或者向上方向键 | 光标向上移动一个字符 |
| l或者向右方向键 | 光标向右移动一个字符 |
| Ctrl + f 或者pageUP键 | 屏幕向前移动一页 |
| Ctrl + b 或者pageDOWN键 | 屏幕向后移动一页 |
| Ctrl + d | 屏幕向前移动半页 |
| Ctrl + u | 屏幕向后移动半页 |
| + | 光标移动到非空格符的下一列 |
| - | 光标移动到非空格符的上一列 |
| n空格（n是数字） | 按下数字n然后按空格，则光标向右移动n个字符，如果该行字符数小于n，则光标继续从下行开始向右移动，一直到n |
| 0（数字0）或者Shift+6 | 移动到本行行首 |
| Shift+4 | 即’$’移动到本行行尾 |
| H | 光标移动到当前屏幕的最顶行 |
| M | 光标移动到当前屏幕的中央那一行 |
| L | 光标移动到当前屏幕的最底行 |
| G | 光标移动到文本的最末行 |
| nG（n是数字） | 移动到该文本的第n行 |
| gg | 移动带该文本的首行 |
| n回车（n是数字） | 光标向下移动n行 |

|  |
| --- |
| **一般模式下查找与替换** |

|  |  |
| --- | --- |
| /word | 向光标之后寻找一个字符串名为word的字符串，当找到第一个word后，按”n”继续搜后一个 |
| ?word | 想光标之前寻找一个字符串名为word的字符串，当找到第一个word后，按”n”继续搜前一个 |
| :n1,n2s/word1/word2/g | 在n1和n2行间查找word1这个字符串并替换为word2，你也可以把”/”换成”#” |
| :1,$s/word1/word2/g | 从第一行到最末行，查找word1并替换成word2 |
| :1,$s/word1/word2/gc | 加上c的作用是，在替换前需要用户确认 |

|  |
| --- |
| **一般模式下删除、复制粘贴** |

|  |  |
| --- | --- |
| x,X | x为向后删除一个字符，X为向前删除一个字符 |
| nx（n为数字） | 向后删除n个字符 |
| dd | 删除光标所在的那一行 |
| ndd（n为数字） | 删除光标所在的向下n行 |
| d1G | 删除光标所在行到第一行的所有数据 |
| dG | 删除光标所在行到末行的所有数据 |
| yy | 复制光标所在的那行 |
| nyy | 复制从光标所在行起向下n行 |
| p,P | p复制的数据从光标下一行粘贴，P则从光标上一行粘贴 |
| y1G | 复制光标所在行到第一行的所有数据 |
| yG | 复制光标所在行到末行的所有数据 |
| J | 讲光标所在行与下一行的数据结合成同一行 |
| u | 还原过去的操作 |

|  |
| --- |
| **进入编辑模式** |

|  |  |
| --- | --- |
| i | 在当前字符前插入字符 |
| I | 在当前行行首插入字符 |
| a | 在当前字符后插入字符 |
| A | 在当前行行末插入字符 |
| o | 在当前行下插入新的一行 |
| O | 在当前行上插入新的一行 |
| r | 替换光标所在的字符，只替换一次 |
| R | 一直替换光标所在的字符，一直到按下ESC |

|  |
| --- |
| **命令模式** |

|  |  |
| --- | --- |
| :w | 将编辑过的文本保存 |
| :w! | 若文本属性为只读时，强制保存 |
| :q | 退出vim |
| :q! | 不管编辑或未编辑都不保存退出 |
| :wq | 保存，退出 |
| :e! | 将文档还原成最原始状态 |
| ZZ | 若文档没有改动，则不储存离开，若文档改动过，则储存后离开，等同于:wq |
| :w [filename] | 编辑后的文档另存为filename |
| :r [filename] | 在当前光标所在行的下面读入filename文档的内容 |
| :set nu | 在每行的行首显示行号 |
| :set nonu | 取消行号 |
| n1,n2 w [filename] | 将n1到n2的内容另存为filename这个文档 |
| :! command | 暂时离开vim运行某个linux命令，例如 :! ls /home 暂时列出/home目录下的文件，然后会提示按回车回到vim |

暂时就讲这么多了。如果你全部掌握，你就是vim高手啦。如果你觉得太多，只要记住笔者标红部分即可，其他的用时再过来查就ok啦。下面笔者给你留一个小作业，希望你要认真完成！

## 第八章 Linux磁盘管理

<http://www.92csz.com/study/linux/8.htm>

【**查看磁盘或者目录的容量 df 和 du**】

**df**查看已挂载磁盘的总容量、使用容量、剩余容量等，可以不加任何参数，默认是按k为单位显示的

df常用参数有 –i -h -k –m等

-i 使用inodes 显示结果

-h 使用合适的单位显示，例如G

-k -m 分别为使用K，M为单位显示

**du 用来查看某个目录所占空间大小**

语法：du [-abckmsh] [文件或者目录名] 常用的参数有：

-a：全部文件与目录大小都列出来。如果不加任何选项和参数只列出目录（包含子目录）大小。

-b：列出的值以bytes为单位输出，默认是以Kbytes

-c：最后加总

-k：以KB为单位输出

-m：以MB为单位输出

-s：只列出总和

-h：系统自动调节单位，例如文件太小可能就几K，那么就以K为单位显示，如果大到几G，则就以G为单位显示。笔者习惯用 du –sh filename 这样的形式。

**【磁盘的分区和格式化】**

笔者经常做的事情就是拿一个全新的磁盘来分区并格式化。这也说明了作为一个linux系统管理员，对于磁盘的操作必须要熟练。所以请你认真学习该部分内容。

fdisk linux下的硬盘分区工具

语法： fdisk [-l ] [设备名称]

-l ：后边不跟设备名会直接列出系统中所有的磁盘设备以及分区表，加上设备名会列出该设备的分区表。

如果不加-l 则进入另一个模式，在该模式下，可以对磁盘进行分区操作。

刚进入该模式下，会有一个提示Command (m for help): 此时按m则会打印出帮助列表，如果你英文好，我想你不难理解这些字母的功能。笔者常用的有p, n,d, w, q.

P：打印当前磁盘的分区情况。

n：重新建立一个新的分区。

w：保存操作。

q：退出。

d：删除一个分区

因为笔者的linux系统是安装在虚拟机上的，所以我可以增加一块新的磁盘。然后笔者会把新的磁盘分成多个分区。

当再次fdisk -l 查看时发现多了一个/dev/hdb 设备，并提示该设备没有可用的分区表。那么下面就来分一下这个/dev/hdb.

首先用p查看一下，并没有任何分区信息

用n创建一个新的分区，会提示要建立e （extended 扩展分区）或者p （primary partition主分区），这里笔者选择主分区，所以按了p回车后，又让输入First cylinder 你或者直接回车或者输入一个数字，因为这块磁盘是新的并没有任何分区，所以直接回车其实就是从1开始了。你也可以自定义输入，但不要超过2080，笔者这里输入1回车。此时会提示要分多大，可以写一个数值（2-2080），也可以输入+sizeK或者+sizeM，后者比较直观容易理解，所以笔者在这里输入+100M，即我分了一个100M的主分区。再用p查看时，果真多出来一个分区。然后笔者继续重复前面的操作，建立了4个主分区。当笔者再次输入n创建分区时，结果提示错了。

由此你会发现，在linux中最多只能创建4个主分区，那如果你想多创建几个分区如何做？很容易，在创建完第三个分区后，创建第四个分区时选择扩展分区。

先删除第四个主分区，然后建立一个扩展分区

在建立扩展分区时，会问你要分多少给这个扩展分区，笔者直接回车，即把所有空间都分给了这个扩展分区。这个扩展分区/dev/hdb4并不能往里写数据，它只是一个空壳子，需要我们继续在这个空壳中继续创建分区。

当建立完扩展分区，然后按n创建新分区时你会发现不再提示是要建立p还是e了，因为我们已经不能再创建p了。在这里需要你明白的是，hdb5 其实只是 hdb4 中的一个子分区，到目前为止可用的分区也才4个，那笔者就再创建第5个分区出来。

然后按w保存，该模式自动退出，如果你不想保存分区信息直接按q即可退出。

下面我们把刚分好的分区删除，重新建立分区。如何删除你还记得吧，对了就是直接按d然后选择合适的数字。删除完所有分区后，这块磁盘就恢复如初了。

第一个分区，我们就建立成扩展分区。并且分给它200M。

当再次新建分区时，发生了变化，不再是p或者e了，而是p或者l（逻辑分区），这是为什么呢？在上面也提到了，一个扩展分区只是一个空壳，在扩展分区下才可以继续划分小的分区，这个小的分区其实就是逻辑分区了。

而且这个逻辑分区默认都是从字数5开始的，因为前面的数字要么给主分区留着，要么给扩展分区留着。由此我们也可以得到，在linux中最多可以创建4个主分区，一旦创建4个主分区后就不能增加任何分区了。另外最多也只能创建一个扩展分区。扩展分区下的逻辑分区最多可以创建多少呢？IDE的硬盘（类似于hda, hdb, hdc 等）最多可以创建10个（hdb5-hdb15），这是笔者试验出来的结果。有的资料说linux下的逻辑分区是没有限制的，也有的说最大可以到64，至于对不对，需要你去近一步考察了，我们没有必要多么深入的研究这个问题，也没有什么意义。

通过以上操作，相信你也学会了用fdisk 来分区了吧。值得提出的是，不要闲着没事分区玩儿，这操作的危险性是很高的，一不留神就把你服务器上的数据全部给分没有了。如果有分区的操作，那么请保持百分之二百的细心，切记切记！

**mkfs.ext2 / mkfs.ext3 /mke2fs 格式化linux硬盘分区**

当用man查询这三个命令的帮助文档时，你会发现我们看到了同一个帮助文档，这说明三个命令是一样的。常用的选项有：

-b：分区时设定每个数据区块占用空间大小，目前支持1024, 2048 以及4096 bytes每个块。

-i：设定inode大小

-N：设定inode数量，有时使用默认的inode数不够用，所以要自定设定inode数量。

-c：在格式化前先检测一下磁盘是否有问题，加上这个选项后会非常慢

-L：预设该分区的标签label

-j：建立ext3格式的分区，如果使用mkfs.ext3 就不用加这个选项了

不加任何选项，直接格式化/dev/hdb1

上例中更改了block size为4096 默认是1024，而inode大小设定为4096。

下面的例子分区时自定义分区的label（标签）名。

**e2label 用来查看或者修改分区的标签（label）**

这个命令很简单，后边直接跟分区编号，即可查看该分区的label，当想要修改标签名时，分区编号后边跟想要的标签名即可。

**fsck 检查硬盘有没有坏道**

语法： fsck [-Aar] [分区]

-A ：加该参数时，后不需要跟分区名作为参数。它会自动检查/etc/fstab 文件下的所有分区（开机过程中就会执行一次该操作）；

-a ：自动修复检查到有问题的分区；

-r ：当检查到有坏道的分区时会让用户决定是否修复。

当你使用fsck检查磁盘有无坏道时，会提示用户“跑这个任务可能会导致某些挂载的文件系统损坏”，所以这个命令不要轻易运行。否则真的遇到问题，系统甚至都不能启动了。

**【挂载/卸载磁盘】**

在上面的内容中讲到了磁盘的分区和格式化，那么格式化完了后，如何去用它呢？这就涉及到了挂载这块磁盘。格式化后的磁盘其实是一个块设备文件，类型为b，也许你会想，既然这个块文件就是那个分区，那么直接在那个文件中写数据不就写到了那个分区中么？当然不行。

在挂载某个分区前需要先建立一个挂载点，这个挂载点是以目录的形式出现的。一旦把某一个分区挂载到了这个挂载点（目录）下，那么再往这个目录写数据使，则都会写到该分区中。这就需要你注意一下，在挂载该分区前，挂载点（目录）下必须是个空目录。其实目录不为空并不影响所挂载分区的使用，但是一旦挂载上了，那么该目录下以前的东西就不能看到了。只有卸载掉该分区后才能看到。

**mount 挂载设备**

先建立/test1 /test2 目录，然后在/test1目录下建立一个1.txt文件

把/dev/hdb1分区挂载到/test1目录，然后再查看/test1目录发下，1.txt不存在了。此时往/test1目录下写数据，则会写到/dev/hdb1分区中。在讲mount的-a选项时，我们有必要先了解一下这个文件 /etc/fstab

这个文件是系统启动时，需要挂载的各个分区。第一列就是分区的label；第二列是挂载点；第三列是分区的格式；第四列则是mount的一些挂载参数，等下会详细介绍一下有哪些参数，一般情况下，直接写defaults即可；第五列的数字表示是否被dump备份，是的话这里就是1，否则就是0；第六列是开机时是否自检磁盘，就是刚才讲过的那个fsck检测。1，2都表示检测，0表示不检测，在Redhat中，这个1，2还有个说法，/ 分区必须设为1，而且整个fstab中只允许出现一个1，这里有一个优先级的说法。1比2优先级高，所以先检测1，然后再检测2，如果有多个分区需要开机检测那么都设置成2吧，1检测完了后会同时去检测2。下面该说说第四列中常用到的参数了。

async/sync ：async表示和磁盘和内存不同步，系统每隔一段时间把内存数据写入磁盘中，而sync则会时时同步内存和磁盘中数据；

auto/noauto ：开机自动挂载/不自动挂载；

default：按照大多数永久文件系统的缺省值设置挂载定义，它包含了rw, suid, dev, exec, auto, nouser,async ；

ro：按只读权限挂载 ；

rw：按可读可写权限挂载 ；

exec/noexec ：允许/不允许可执行文件执行，但千万不要把根分区挂载为noexec，那就无法使用系统了，连mount命令都无法使用了，这时只有重新做系统了；

user/nouser ：允许/不允许root外的其他用户挂载分区，为了安全考虑，请用nouser ；

suid/nosuid ：允许/不允许分区有suid属性，一般设置nosuid ；

usrquota ：启动使用者磁盘配额模式，磁盘配额相关内容在后续章节会做介绍；

grquota ：启动群组磁盘配额模式；

学完这个/etc/fstab后，我们就可以自己修改这个文件，增加一行来挂载新增分区。例如，笔者增加了这样一行

/dev/hdb1 /test1 ext3 defaults 0 0

那么系统再重启时就会挂载这个分区了。

讲完了/etc/fstab 我们继续回来讲这个mount，mout -a 如果运行了这个命令，则会把/etc/fstab中出现的所有磁盘分区挂载上。所以当你在/etc/fstab文件中增加一行后，你完全可以直接运行mount -a 来挂载你增加的那行，这样就不用重启啦。

你可以使用mount -o 选项来重新挂载一个分区，并同时指定你想要的选项（即上边介绍fstab第六列中那些）

看到了吧，使用了ro选项，则不能新建文件了。

再重新挂载一次就恢复正常了，如果不加任何其他选项，则就是defaults。

笔者在日常的运维工作中遇到过这样的情况，一台服务器上新装了亮块磁盘，磁盘a（在服务器上显示为sdc）和磁盘b（在服务器上显示为sdd），有一次把这两块磁盘都拔掉了，然后再重新插上，重启机器，结果磁盘编号调换了，a变成了sdd，b变成了sdc（这是因为把磁盘插错了插槽），问题来了。通过上边的学习，你挂载磁盘是通过/dev/hdb1 这样的分区名字来挂载的，如果先前加入到了/etc/fstab 中，结果系统启动后则会挂载错分区。那么怎么样避免这样的情况发生？

**blkid**这个命令是用来显示磁盘分区uuid的，uuid其实就是一大串字符，在linux系统中每一个分区都会有唯一的一个uuid。说到这，聪明的你想到了吧，没有错，我们就用这唯一的uuid来挂载磁盘分区。

这个命令笔者只是用来显示uuid，没有其他用途所以不做详细介绍，当然你也可以在命令后边跟某一个分区，只显示该分区的uuid。

看到了吧，其实是很好用的。那么怎么让它也开机启动？很简单，把刚才敲的mount 磁盘的命令直接写到 /etc/rc.d/rc.local 文件即可。对了，笔者到现在还没有给你讲过这个rc.local文件的作用。简单点说，系统启动完后会执行这个文件中的命令。所以只要你想开机后运行什么命令统统写入到这个文件下面吧。

其实这个文件就是一个shell 脚本，以后笔者会单独用一章来介绍它。行开头的”#”是注释的意思，代表这行在这个脚本中不生效。你想让系统开机后运行什么命令，就把什么命令写到这里面来。就比如刚才笔者挂载的那条命令。你可以这样实现：

mount 还有一个比较常用的选项就是-t ，后边指定文件系统的类型，比如挂载软盘时就需要指定 vfat，而挂载光盘时就需要指定iso9660，但在笔者多年来的经验，目前的系统都是智能识别所要挂载分区的系统格式类别的。也就是说，用不着你去指定，它会自动判断的。

**umount 卸载设备**

现在你学会了如何挂载一个设备，那么如何去卸载一个设备呢，这就要用到umount了，这个命令也简单的很，后边可以跟挂载点，也可以跟分区名(/dev/hdb1)

有时也许你会遇到比较难卸载的设备，就像在windows下无法删除U盘一样，教你一个特管用的方法就是 umount -l /dev/hdb1 ，这个-l选项有强制卸载的意思，你一定要记住哦，非常有用的。

**【建立一个swap文件】**

从装系统时就接触过这个swap了，前面也说过它类似与windows的虚拟内存，分区的时候一般大小为内存的2倍，如果你的内存超过4G，那么你分8G似乎是没有必要了。分4G足够日常交换了。然而，还会有虚拟内存不够用的情况发生。如果真遇到了，莫非还要重新分一下磁盘？当然不能！那我们就增加一个虚拟的磁盘出来。

基本的思路就是：建立swapfile 🡪 格式化为swap格式 🡪 启用该虚拟磁盘

利用dd 来创建一个419M的文件/tmp/newdisk出来，其中if代表从哪个文件读，/dev/zero是linux下特有的一个0生成器，of表示输出到哪个文件，bs即块大小，count则定义有多少个块。

mkswap 这个命令是专门格式化swap格式的分区的，这个命令用的时候一定要看清楚了，否则把其他分区给格式化错了就只有哭了。

free 是用来查看系统内存以及虚拟内存使用情况的，-m选项是以M的形式查看。可以看到当前系统的。而swapon 是启用我们新建的swap文件，启用后再用free查看发现多了400M。

我们还可以用swapoff 关闭启用的swap文件。

**【磁盘配额】**

磁盘配合其实就是给每个用户分配一定的磁盘额度，只允许他使用这个额度范围内的磁盘空间。在linux系统中，是多用户多任务的环境，所以会有很多人共用一个磁盘的情况。针对每个用户去限定一定量的磁盘空间是有必要的，这样才显得公平。

在linux中，用来管理磁盘配额的东西就是quota了。如果你的linux上没有quota，则需要你安装这个软件包 quota-3.13-5.el5.RPM （其实版本是多少无所谓了，关键是这个软件包）。quota在实际应用中是针对整个分区进行限制的。如果你的/dev/hda3 是挂载在/home 目录下的，那么/home 所有目录都会受到限制。

quota 这个模块主要分为quota quotacheck quotaoff quotaon quotastats edquota setquota warnquota repquota这几个命令，下面就分别介绍这些命令。

quota 用来显示某个组或者某个使用者的限额。

语法：quota [-guvs] [user,group]

-g ：显示某个组的限额

-u ：显示某个用户的限额

-v ：显示的意思

-s ：选择inod或硬盘空间来显示

quotacheck 用来扫描某一个磁盘的quota空间。

语法：quotacheck [-auvg] /path

-a ：扫描所有已经mount的具有quota支持的磁盘

-u ：扫描某个使用者的文件以及目录

-g ：扫描某个组的文件以及目录

-v ：显示扫描过程

-m ：强制进行扫描

edquota 用来编辑某个用户或者组的quota值。

语法：edquota [-u user] [-g group] [-t]

edquota -p user -u user

-u ：编辑某个用户的quota

-g ：编辑某个组的quota

-t ：编辑宽限时间

-p ：拷贝某个用户或组的quta到另一个用户或组

当运行edquota -u user 时，系统会打开一个文件，你会看到这个文件中有7列，它们分别代表的含义是：

Filesystem ：磁盘分区，如/dev/hda3

blocks ：当前用户在当前的Filesystem中所占用的磁盘容量，单位是Kb。该值请不要修改。

soft/hard ：当前用户在该Filesystem内的quota值，soft指的是最低限额，可以超过这个值，但必须要在宽限时间内将磁盘容量降低到这个值以下。hard指的是最高限额，即不能超过这个值。当用户的磁盘使用量高于soft值时，系统会警告用户，提示其要在宽限时间内把使用空间降低到soft值之下。

inodes ：目前使用掉的inode的状态，不用修改。

quotaon 启动quta，在编辑好quota后，需要启动才能是quta生效

语法：quotaon [-a] [-uvg directory]

-a ：全部设定的quota启动

-u ：启动某个用户的quota

-g ：启动某个组的quota

-s ：显示相关信息

quotaoff 关闭quota

该命令常用只有一种情况 quotaoff -a 关闭全部的quota

以上讲了很多quota的相关命令，那么接下来笔者教你如何在实践应用中去做这个磁盘配额。整个执行过程如下：

首先先确认一下，你的/home目录是不是单独的挂载在一个分区下，用df 查看即可。如果不是则需要你跟我一起做。否则这一步即可省略。

笔者的linux系统中，/home并没有单独占用一个分区。所以需要把/home目录挂载在一个单独的分区下，因为quota是针对分区来限额的。

笔者用fdisk -l 查看目前/dev/hdb 磁盘有5个可用分区，所以笔者打算把/dev/hdb1挂载在/home 目录下

看到了吧，目前笔者的/home目录已经是一个单独的分区了。

1）建立测试用户

首先建立一个test用户，则同时建立了一个test组。可以在/etc/passwd中有以test为开头的行，其中uid和gid都为500 ，然后又建立一个test1账号，使其加入test组，查看/etc/passwd文件发现test和test1用户的gid都为500。（也许你对/etc/passwd文件、增加一个用户以及uid和gid等概念不熟悉，没有关系，在以后的章节中会做介绍，在这里只需要你明白即可）

2）打开磁盘的quota功能

默认linux并没有对任何分区做quota的支持，所以需要我们手动打开磁盘的quota功能，你是否记得，在前面内容中分析/etc/fstab文件的第四列时讲过这个quota选项（usrquota, grpquota）。没错，要想打开这个磁盘的quota支持就是需要修改这个第四列的。用vim编辑/etc/fstab 加入一行，如下图：

vim命令将会在后续章节详细介绍，前面介绍过如何进入编辑模式以及如何保存文件。如果你的linux系统已经有/home这一行，那么直接修改第四列，加上usrquota,grpguota（中间没有空格）。接下来需要重新挂载/home。

另外你也可以这样实现重新挂载/home

如何查看是否启用了quota呢？

只要查看/etc/mtab文件中/home所在那行是否有usrguota,grpquota即可。笔者的/dev/hdb1现在已经支持了quota

3）扫描磁盘的使用者使用状况，并产生重要的aquota.group与aquota.user

这一步就需要用到quotacheck了，aquota.group与aqouta.user分别是组以及用户磁盘配额需要的配置文件。如果没有这两个文件，则磁盘配额是不会生效的。

当首次使用quotacheck命令时，会提示“cannot stat old user quota file ……”其实这是在提示你在/home目录下没有aquota.user以及aquota.group两个文件。没有关系，因为以前并没有配置过磁盘配额，当然没有这两个文件了。当执行完quotacheck命令后，会在/home目录下生成这两个文件的。

4）启动quota配额

5）编辑用户磁盘配额

先来设定test账户的配额，然后直接把test的配额拷贝给test1即可。这里就需要用到edquota了

讲上面内容修改为

其中单位是Kb，所以soft 值大约为20Mb，hard值为30Mb，保存这个文件，保存的方式跟vim一个文件的方式一样的。

将test的配额复制给test1。下面继续设定宽限时间

默认是7days 在这里我们改为1days。下面查看一下test以及test1用户的配额吧。

6）编辑组磁盘配额

设定组test的soft配额值为40M，hard值为50M。下面查看组test的配额。

7）设定开机启动

前面已经讲到启动磁盘配额的命令是quotaon -aug ，所以要想开机启动，只需将这条命令加入到 /etc/rc.d/rc.local文件即可。

## 如何没有设置DNS服务器的话，那么系统就不能正常上网了。

通过编辑vi  /etc/sysconfig/network-scripts 网卡下面的ifcfg-eth0 配置文件添加DNS服务器地址

查看DNS服务器地址网卡配置，输入 ifconfig -a 可以查看所有设备配置

在终端输入 setup命令即可调出图形界面设置（配置信息会写入/etc/resolv.conf文件里面）

在Linux下面，有一个默认的DNS服务器地址配置文件的设置，存放在 /etc/resolv.conf

设置方法很简单，通过编辑 vi /etc/resolv.conf 设置首选DNS和次要DNS。如下图，排在前面的就是首选DNS，后面一行就是次要的DNS服务器DNS

跟Windows系统一样，Linux系统也有主机表文件。同样可以通过设置主机表地址进行特定主机的解析，优先DNS服务器地址。

配置文件，在/etc/hosts

**总结：**

Linux下设置DNS的位置主要是，

1网卡设置配置文件里面DNS服务器地址设置，2.系统默认DNS服务器地址设置。3,hosts文件指定

生效顺序是：

1 hosts文件 ---- 2 网卡配置文件DNS服务地址 ---3 /etc/resolv.conf

## 详述Linux配置静态IP、设置DNS和主机名

Linux配置静态IP、设置DNS和主机名首先要找到配置文件，这是在Linux系统下进行工作的必须知道工作方式。后面一步步的跟着这个范例来进行配置相信你最终也会完成Linux配置静态IP、设置DNS和主机名。

配置文件位于：

/etc/sysconfig/network-scripts/ifcfg-eth0

DEVICE=eth0

ONBOOT=yes

BOOTPROTO=static

IPADDR=192.168.0.3

NETMASK=255.255.255.0

GATEWAY=192.168.0.1

使IP地址生效：

/sbin/ifdown eth0

/sbin/ifup eth0

配置dns解析

echo "nameserver 211.98.1.28">> /etc/resolv.conf

通知网关更新信息：

/etc/init.d/network restart

下面我们配置服务器11.0.0.1为21PHP.COM的主DNS服务器

Linux下的dns功能是通过bind软件实现的。bind软件安装后，会产生几个固有文件，分为两类，一类是配置文件在/etc目录下，一类是 dns记录文件在/var/named目录下。加上其他相关文件，共同设置dns服务器。下面是所有和dns设置相关文件的列表与说明。

位于/etc目录下的有：hosts,host.conf,resolv.conf,named.boot,named.conf。

1、“hosts”文件，定义了主机名和ip地址的对应，其中也有将要运行dns这台电脑的ip地址和主机名。内容：

127.0.0.1 localhost.localdomain localhost

2、“host.conf”文件[Boban注：原来误写成hosts.conf]，"order hosts bind"语句，指定了对主机名的解析顺序是先到hosts中查找，然后到dns服务器的记录里查找。“multi on”则是允许一个主机名对应多个ip地址。内容：

order hosts, bind

multi on

nospoof on

3、“resolv.conf”文件，“nameserver 10.0.0.211”指定了dns服务器的地址。注意，这个文件对普通非dns服务器的电脑(非windows的系统；Windows系统是在“网络属性”中设置这项的)来说，是必不可少的。你如果没有设置本机为dns服务器，你又要能够解析域名，就必须指定一个dns服务器的地址。你可以最多写上三个地址，作为前一个失败时的候选dns服务器。“domain zrs.com”指定默认的域。文件内容：

domain 21php.com

nameserver 11.0.0.1

4、“named.boot”文件是早期版本的bind软件使用的配置文件，现在新版本中已经让位于“named.conf”。named.conf是dns server配置的核心文件。下面我们一段一段的来解说。

# named.conf - configuration for bind

#

# Generated automatically by bindconf, alchemist et al.

controls {

inet 127.0.0.1 allow { localhost; } keys { rndckey; };

};

include "/etc/rndc.key";options {

directory "/var/named/";

};

zone "." {

type hint;

file "named.ca";

};

zone "0.0.127.in-addr.arpa" {

type master;

file "0.0.127.in-addr.arpa.zone";

};

zone "localhost" {

type master;

file "localhost.zone";

};

zone "21php.com" {

type master;

notify yes;

file "21php.com";

};

上文中#为注释符号， 其他各行含义如下：

diretory /var/named

指定named从 /var/named 目录下读取DNS数据文件，这个目录用户可自行指定并创建，指定后所有的DNS数据文件都存放在此目录下；

zone "." {

type hint;

file "named.ca";

};

指定named从 named.ca 文件中获得Internet的顶层“根”服务器地址 。

zone "0.0.127.in-addr.arpa" {

type master;

file "0.0.127.in-addr.arpa.zone";

};

指定named作为127.0.0网段地址转换主服务器，named.local文件中包含了127.0.0.\*形式的地址到域名的转换数据(127.0.0网段地址是局域网接口的内部 loopback地址)；

zone "localhost" {

type master;

file "localhost.zone";

};

指定包含localhost的DNS文件数据存放在/var/named/localhost.zone中；

zone "21php.com" {

type master;

notify yes;

file "21php.com.zone";

};

以上语句表明域21php.com的DNS数据存放在/var/named/目录下的21php.com.zone中；

我们可以用文本编辑器打开/var/named/21php.com.zone，其内容如下：

$TTL 86400 @ IN SOA @ root.localhost (

2 ; serial

28800 ; refresh

7200 ; retry

604800 ; expire

86400 ; ttl

)

@ IN NS localhost

www IN A 11.0.0.233

www2 IN A 11.0.0.23

forum IN A 11.0.0.10

@ IN MX 5 mail.21php.com.

该文件的前部分是相应的参数设置，此部分不需要改动，后面的部分就是具体的DNS数据；

例如：

www IN A 11.0.0.233

将www.21php.com 解析到地址11.0.0.233；

www2 IN A 11.0.0.23

将www2.21php.com 解析到地址11.0.0.23；

club IN A 11.0.0.10

将club.21php.com 解析到地址11.0.0.10；

现在先说到这里，下篇文章接着介绍Linux配置静态IP、设置DNS和主机名的后续步骤。

## 因为你有按键中断了（内核自带按键中断）

【传说】答疑助手－杨(742506959)  21:12:17

在内核配置中make menuconfig找到driver里面的按键，然后取消掉，重新编译内核，下载  
就好了

【吐槽】小屁孩(532864082) 21:13:20

哦，原来是这样，我还在串口里面输入，rmmod卸载，原来是在那里做，

## 顶层Makefile和子Makefile如何建立关联

### 顶层Makefile范例:

OBJS := start.o main.o **dev/dev.o lib/lib.o**

**CFLAGS := -nostdinc -fno-builtin -I$(shell pwd)/include**

**export CFLAGS**

gboot.bin : gboot.elf

arm-linux-objcopy -O binary gboot.elf gboot.bin

gboot.elf : $(OBJS)

arm-linux-ld -Tgboot.lds -o gboot.elf $^

%.o : %.S

arm-linux-gcc -g -c $^

%.o : %.c

arm-linux-gcc $(CFLAGS) -c $^

**lib/lib.o :**

**make -C lib all**

**dev/dev.o :**

**make -C dev all**

**.PHONY: clean**

**clean:**

**rm \*.o \*.elf \*.bin**

**make -C lib clean**

**make -C dev clean**

### 子Makefile范例:lib/Makefile:

**objs := div64.o lib1funcs.o ctype.o muldi3.o printf.o string.o vsprintf.o**

**all : $(objs)**

**arm-linux-ld -r -o lib.o $^ #把该目录所有.o集成一个.o文件**

%.o : %.c

arm-linux-gcc **${CFLAGS}** -c $^

%.o : %.S

arm-linux-gcc **${CFLAGS}** -c $^

clean:

rm -f \*.o

## bash: ./configure: /bin/sh^M: bad interpreter: No such file or directory

输入命令：dos2unix configure 把它转化成linux文件

或者 cat ./configure.sh | tr -d ‘/r’ > temp.sh

## 关于-/bin/sh:xx(命令) not found 的几种原因和解决办法

关于-/bin/sh:xx(命令) not found  的几种原因：

1./bin/目录下没有这个命令；

2.bin/目录有这个命令，只是执行权限不够，或者程序执行权限不够；

3.程序需要的静态库或者动态库没有；

## Linux视频播放器或系统没有声音

重新配置编译内核，在内核配置中勾选上支持ALSA的相关驱动即可。

或者将老内核配置修改替换新内核配置即可(/usr/src/kernels/${VERSION}/.config)

## Linux中运行某程序时找不到某库，但该库确实安装了

方法1：把相应库所在的目录新增在ld.so.conf文件里(/etc/ld.so.conf)

方法2：export LD\_LIBRARY\_PATH=/usr/local/lib:$LD\_LIBRARY\_PATH

方法3：给xxx.so.n做个软链接至系统lib目录(ld –s ...).

## Linux中无法读写NTFS文件系统

第一步：首先要让Linux内核支持NTFS文件系统

\*\*内核配置选上<>FUSE(FileSystem in Userspace) support

\*\*内核配置选上DOS/FAT/NTFilesystems ---里的NTFS选项

\*\*重新编译内核并启动内核

第二步：这时我们重启系统并挂载ntfs分区:mount –t ntfs –o rw /dev/sda2 /mnt

\*\*上面我们能成功挂载，但进行测试写操作时失败，提示:Permission denied

第三步：由于上面的原因，我们还得安装一个工具:

\*\*检查是否安装了fuse工具(which fuse)，因为fuse是ntfs-3g的桥梁

\*\*然后通过源码安装ntfs-3g这个工具

第四步：使用这个ntfs-3g

\*\*mount要注意的参数：mount -t ntfs-3g<dev><dir>

\*\*或者直接使用: ntfs-3g <dev><dir>

\*\*范例1: mount -t ntfs-3g /dev/sda2 /media/windows/c

\*\*范例2: ntfs-3g /dev/sda2 /media/windows/c

## 内核模块:多个模块依赖编译成一个.ko文件

范例1:

obj-m:=**main.o**

**main**-objs:=f1.o f2.o f3.o

范例2:

obj-m +=**main.o**

**main**-objs +=f1.o f2.o f3.o

# 第七季-我的运筹帷幄

关于兴趣论

通过本笔记的修改、学习和总结，我们完全能从中提炼出具有经济价值的技术和信息。我们完全能把其中的技术信息转化为为我所用的经济信息。

我喜欢唱的歌曲：

《愿得一人心 – 李行亮》

《走着走着就散了 - 庄心妍》

《我爱的人 – 陈小春》

《说谎 – 林宥嘉》

《告白气球 - 周二珂》

## 嵌入式软件工程师技术要求

第一季-嵌入式的童年

熟练使用gcc、熟练使用gdb、熟练使用Makefile

1. 裸奔吧ARM

熟悉ARM或者MIPS汇编（非单片机方向）。

熟悉ARM体系结构及接口编程。

熟悉ARM处理器脚骨；

熟练掌握嵌入式交叉编译环境和软件调试工具；

具有数模基础，能看懂原理图；

熟练掌握嵌入式交叉编译环境和软件调试工具；

能熟练运用JTAG等各种软件开发和调试工具进行软件测试和调试；

熟悉基本的硬件设计原理，熟练阅读硬件原理图；

深入理解bootloader

1. 小应用大智慧

精通C语言,

精通C++程序设计

熟悉面向对象编程

具备良好的编码习惯。

熟悉数据结构及常用算法

熟悉Linux高级编程、

文件操作

多进程和多线程

TCP/IP网络

熟悉shell脚本语言；

了解基本的应用编程（系统编程、网络编程）。

1. 内核驱动奥秘多

移植uboot经验

裁剪过Linux内核

掌握网络设备驱动开发

有Linux操作系统内核移植优化

做过bootloader

做过内核的移植和裁剪。

做过根文件系统的移植和裁剪。

具有丰富的Linux驱动开发/移植经验，

camera嵌入式软件开发经验者优先；

混杂设备驱动模型：混杂设备

Linux中断处理：一对一中断注册

中断分层处理：上半部和下半部

按键定时器祛痘：定时器

驱动支持多按键：一对多中断注册

阻塞型驱动设计：等待队列

总线设备驱动模型：总线模型和平台总线模型

串口设备驱动开发

网卡芯片设备驱动进行开发、移植

Nand/Nor芯片设备驱动进行开发、移植、

I2C接口设备驱动进行开发、移植

SPI接口设备驱动进行开发、移植、

USB外设驱动的开发

SDIO驱动开发

MIPI驱动开发

WIFI驱动开发等驱动；

CAN外设驱动的开发

SD卡外设驱动的开发

SPC总线和接口、

PCI接口芯片设备驱动进行开发、移植、

CF卡口设备驱动进行开发、移植，

熟悉多种类型MCU；

熟练掌握Linux驱动程序的开发流程和工作原理；

熟悉uboot开发

熟悉Linux内核开发

对常见的嵌入式设备驱动能够进行开发、移植，

熟悉Linux驱动子系统。

1. 小试牛刀项目开发

掌握嵌入式开发的基本流程及IDE环境的使用；

有良好的编程习惯，逻辑分析能力；

具备良好的中英文专业文档阅读、编写能力；

熟悉Linux移植、裁剪、驱动开发和应用开发；

有丰富的嵌入式系统开发经验，良好开发习惯文档规范及编码习惯；

熟悉嵌入式软件接口规范如CMSIS；

熟悉嵌入式软件设计方法；

有程序架构设计经验；

能对代码进行面积优化或者性能优化；

对软件的模块化、可移植性有深入理解；

熟悉SVN和GIT代码管理工具；

学习能力强，有团队合作意识，待人热情

负责公司嵌入式设备驱动程序开发，以Linux驱动开发为主；

## 嵌入式入门学习总结

给新刚步入嵌入式同学们的一些忠告

2016被猴耍了，2017抓住机会，撸起袖子加油干！！！

1）学嵌入式，先学会动手，学会搜集资料再说进一步学习的事儿，假如感觉做不到，请绕行

2）查找自己开发板资料，千百度，万Google，不如先将手头上的《用户手册》以及帮助文档来的快

3）当你将开发板拿到手的那一刻，先将开发板光盘资料浏览一遍再说去做别的实验，保证将来你事半功倍

4）原理性的东西虽然枯燥乏味，但万变不离其宗。技再好，也需要内力的支持

5）做开发，谁也不敢保证自己遇到过所有问题，多查看帮助文档，你会少走很多弯路

6）动手操作性的东西易出错，错不在人而在己，保证是你某些细节注意的不到

7）别人提供的范例仅仅是一个Demo，好多事，虽然繁琐，最终还是需要你自己去做

8）出的错，往往比你想掌握的东西学到的还要多，经验就是在这些错里面总结出来的

9）学习容易遗忘，做好笔记狠重要

10）做事给自己做个周志与周报，这样收获的效果你往往意想不到