用户态（User Space）：应用程序

内核态（Kernel Space）：操作系统

裸机驱动程序

     在linux中，为了更好地保护内核空间，将程序运行空间分为内核空间和用户空间（也就是内核态和用户态），它们分别运行在不同的级别上，逻辑上相互隔离。通常情况下，用户进程不允许访问内核，只能在用户空间操作用户数据，调用用户空间的函数。但在有些情况下，用户空间的进程需要获得一定的系统调用（调用内核空间程序），这时操作系统就必须提供给用户特殊的接口----系统调用。调用时程序运行空间从用户空间进入内核空间，处理完后返回用户空间。

操作系统 = 内核+驱动程序

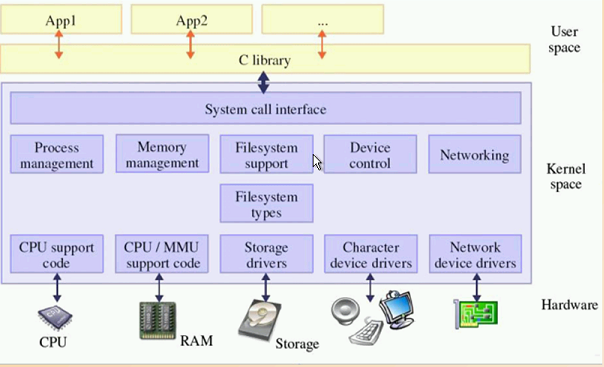
驱动程序是被动运行的，向内核注册，应用触发驱动

驱动程序向上为Linux系统提供访问硬件统一调用接口，

向下用于控制硬件：通过读写硬件寄存器达到控制硬件的目的

内核函数，驱动中用的函数

应用程序----设备文件（设备节点）-----驱动----设备



分类：

字符设备驱动：

     设备以字节流方式访问（以字节为单位读写）

     字符设备驱动实现了open，close，read，write等系统调用

     应用程序通过调用设备文件实现

块设备驱动：

     设备上的数据以块的方式存放（页）

     特别之处：操作硬件的实现方式不一样，先将数据块成块，再操作设备；数据块上的数据按照一定的格式组织：存放文件系统，实现mount（文件挂载）

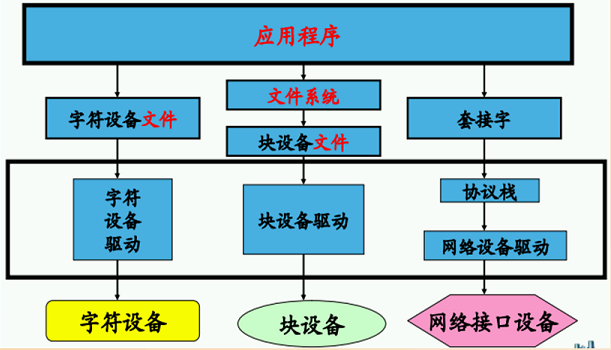
网络设备驱动：

     设备上的数据以不固定大小的帧输入与输出

     没有对应的设备文件

     系统为网络设备访问分配唯一接口（eth0）

     为应用层提供一套数据包传输函数访问接口（Socket）



编写驱动的步骤：

安装交叉编译器

获得相应的内核源代码

编写驱动程序，编译，被动执行，应用触发执行

     编译驱动两种方式，修改配置文件，或者makefile

部署方式：直接编译到内核中，或者编译成模块

用上层应用程序调用

1.cp  ...   .config

2.内核定制与裁剪 ：/char/Kconfig    menuconfig(目录式)<defaoult,默认>

3.内核源码树根目录

4.make modules

5.insmod,lsmod,/proc/modules

1、查看原理图、数据手册，了解设备的操作方法

2、在内核中找到相近的驱动程序，以它为模板开发

3、实现驱动程序的初始化，并向内核注册

4、按照内核规定的驱动框架，实现相关操作函数（如open、read、write）

5、编译驱动程序到内核中，或者编译成模块并挂载（insmod）到内核

将驱动模块源码合入内核源码

设备驱动程序应包含在drivers子目录

首先确认是否存在于设备驱动程序特性相似的目录名 存在则插入相应目录，

否则字符类型插入char目录，块类型插入block目录，网络类型插入net目录

修改内核编译选项文件

Linux内核支持使用内核编译选项包含到内核中的功能

make menuconfig读入这些内核编译选项文件来配置内核

2.6内核编译选项文件为KConfig

进入合入了驱动模块的目录，修改改目录下的KConfig，使得合入的驱动能在配置项中显示

Kconfig选项文件语法

config HELLO\_WORLD

bool “hello world”

help This is a demo make

menuconfig选中“hello world”,配置完成后在.config中加入 CONFIG\_HELLO\_WORLD=y选项

修改内核源代码中的Makefile

Makefile指定了驱动程序的编译规则，使得驱动程序能包含到内核image中

Makefile根据make menuconfig配置设定的编译条件变量，决定是要把特定源代码编译成模块还是包含到内核中，或者是清除。

进入合入了驱动模块的目录，修改改目录下的Makefile，使得合入的驱动源码能编译进内核

Makefile文件语法

Obj-$(CONFIG\_HELLO\_WORLD) += hello\_world.o

根据配置，CONFIG\_HELLO\_WORLD可以设置为“y”、“m”、“n”或者“” 根据内核编译选项的符号决定包含到内核中或者编译为模块，或者不编译。

确认合入内核的驱动在内核启动时自动运行

重新编译并启动新内核，dmesg=>确认“hello world”已打印出来 带\_\_int标志的函数被放入初始化代码段，内核会依次调用初始化代码段的 函数

主设备号与驱动程序file\_operations结构的对应关系

主设备号用来标识与设备文件相连的驱动程序。

次设备号被驱动程序用来辨别操作的是哪个设备

主设备号用来反应设备类型

次设备号用来区分同类型的设备

创建设备文件：

mknod filename type major minor

驱动数据传输模模型 :

