3.3.2 主设备的数据结构——CNTRL

这里要描述的数据结构包含在 device.h 中。

CNTRL 是一个巨大的数据结构，本来它的结构应该更具有层次性，但是我们在这里为了讨论的方便，将所有它涉及到的结构都放在了一起：

```
typedef    struct
  {
/*接口，即外部程序使用设备控制结构的入口部分*/
  int major                /*主设备号*/
  int use;               /* 该控制块的使用次数*/
  int mode;                /* 工作模式选择*/
  CNTRL *cntrl_p      /* 下一个设备控制块*/
  DRIVE *drive;       /* 使用该控制块的从设备*/
  char*ddt;           /* 设备驱动程序入口地址*/

/*信号量，用于控制多任务的环境下不同任务对设备的使用*/
  int access_smphr;        /*控制块使用信号量*/
  int isr_smphr;           /* 中断服务程序信号量*/

/*寄存器*/
  int address;          /* 控制器 I/O 地址*/
  int dma_base;            /* DMA 控制器的基址*/
  int dma_chan;            /* DMA 控制器通道*/
  int dma_scale;           /* DMA 控制器选择*/
  int dor;              /* DOR 寄存器 */
  int intr;             /* 中断基址*/
  int irq;              /* 控制器的中断号*/
  int state;            /* 状态寄存器*/
}CNTRL;
```

该结构是所有块设备公用的，有的设备并不需要提供其中的一些信息。以下是就该数据结构的简单说明：

（1）　接口部分

该部分的参数的作用是供外部程序使用设备控制结构，如：工作模式和设备驱动程序入口地址以及那些从设备使用该设备控制块这些参数是在编写设备驱动程序时就提供了的；主设备号是系统在启动的时候自动分配的；该控制块的使用次数是在启动的时候初始化，并在系统运行中修改。

（2）　信号量

控制块使用一个二进制信号量作为设备控制块访问的互斥信号量，保证在任一时刻只允许有一个任务使用该设备控制块。中断服务程序信号量是用于在该设备的中断服务程序中释放的信号量，利用操作系统的功能通知睡眠在该信号量上的所有进程，在由他们自己决定由哪一个进程处理当前中断。

（3）　寄存器

寄存器对外设来说无疑是非常重要的，它们的设置正确与否决定了块设备是否能够正常工作。所有的这些值在设备初始化是由用户定义，但是程序中会对它们做严格的检查，并对一些严重错误给于纠正，因为此类错误会影响硬件工作，严重的会损坏硬件。比如，不支持一次多扇区传送的硬盘控制器被强制工作在该模式之下。但是对于另一类“错误”，我们把它理解为用户需要，比如支持一次多扇区传送的硬盘控制器被强制工作在单扇区传送模式之下，这可能是因为用户的缓冲区较小，容量有限造成的。

3.3.3 从设备的数据结构——DRIVE

数据结构 DRIVE 定义如下：

```
typedef struct
  {
/*接口*/
  int minor              /*从设备号*/
  int drive;             /* 驱动器号*/
```