 1.3 FrameBuffer FrameBuffer 是出现在 2.2.xx 内核当中的一种驱动程序接口。这种接口将显示设备抽象为帧缓冲区。用户可以将它看成是显示内存的一个映像，将其映射到进程地址空间之后，就可以直接进行读写操作，而写操作可以立即反应在屏幕上。该驱动程序的设备文件一般是 /dev/fb0、/dev/fb1 等等。比如，假设现在的显示模式是 1024x768-8 位色，则可以通过如下的命令清空屏幕： $ dd if=/dev/zero of=/dev/fb0 bs=1024 count=768在应用程序中，一般通过将 FrameBuffer 设备映射到进程地址空间的方式使用，比如下面的程序就打开 /dev/fb0 设备，并通过 mmap 系统调用进行地址映射，随后用 memset 将屏幕清空（这里假设显示模式是 1024x768-8 位色模式，线性内存模式）： int fb. unsigned char* fb_mem. fb = open (“ /dev/fb0 ” , O_RDWR). fb_mem = mmap (NULL, 1024*768, PROT_READ|PROT_WRITE,MAP_SHARED,fb,0). memset (fb_mem, 0, 1024*768).FrameBuffer 设备还提供了若干 ioctl 命令，通过这些命令，可以获得显示设备的一些固定信息（比如显示内存大小）、与显示模式相关的可变信息（比如分辨率、象素结构、每扫描线的字节宽度），以及伪彩色模式下的调色板信息等等。 通过 FrameBuffer 设备，还可以获得当前内核所支持的加速显示卡的类型（通过固定信息得到），这种类型通常是和特定显示芯片相关的。比如目前最新的内核（2.4.9）中，就包含有对 S3、Matrox、nVidia、3Dfx 等等流

行显示芯片的加速支持。在获得了加速芯片类型之后，应用程序就可以将 PCI 设备的内存I/O（memio）映射到进程的地址空间。这些 memio 一般是用来控制显示卡的寄存器，通过对这些寄存器的操作，应用程序就可以控制特定显卡的加速功能。 PCI 设备可以将自己的控制寄存器映射到物理内存空间，而后，对这些控制寄存器的访问，给变成了对物理内存的访问。因此，这些寄存器又被称为“ memio ”。一旦被映射到物理内存，Linux 的普通进程就可以通过 mmap 将这些内存 I/O 映射到进程地址空间，这样就可以直接访问这些寄存器了。 当然，因为不同的显示芯片具有不同的加速能力，对memio 的使用和定义也各自不同，这时，就需要针对加速芯片的不同类型来编写实现不同的加速功能。比如大多数芯片都提供了对矩形填充的硬件加速支持，但不同的芯片实现方式不同，这时，就需要针对不同的芯片类型编写不同的用来完成填充矩形的函数。 说到这里，读者可能已经意识到 FrameBuffer 只是一个提供显示内存和显示芯片寄存器从物理内存映射到进程地址空间中的设备。所以，对于应用程序而言，如果希望在 FrameBuffer 之上进行图形编程，还需要完成其他许多工作。举个例子来讲，FrameBuffer 就像一张画布，使用什么样子的画笔，如何画画，还需要你自己动手完成。