# Linux硬件访问技术

**访问流程：地址映射-🡪读写存储器**

## 地址映射

在Linux系统中，无论是内核程序还是应用程序，都只能使用虚拟地址。根据CPU体系结构的不同，CPU对IO端口的编址方式有两种：  
I/O映射方式（I/O-mapped）  
　　典型地，如X86处理器为外设专门实现了一个单独的地址空间，称为"I/O地址空间"或者"I/O端口空间"，CPU通过专门的I/O指令（如X86的IN和OUT指令）来访问这一空间中的地址单元。

**动态映射** void \*ioremap(physaddr,size)

physaddr:待映射的物理地址

size ：映射的区域长度

返回值：映射后的虚拟地址。

**静态映射** 所谓的静态映射是指Linux系统根据用户事先指定的映射关系，在内核启动时，自动的将物理地址映射为虚拟地址。

寄存器读写

unsigned ioread8(void \*addr) 或unsigned ioread16(void \*addr)或unsigned ioread32(void \*addr)

unsigned ioreadb(address)或unsigned ioreadw(address)或unsigned ioreadl(address)

与读相对应的写的函数分别为

void iowrite8/16/32(u8/16/32 value,void \*addr)

void iowriteb/w/l(unsigned value,address)

**解除映射**

int iounmap(void \*start)

功能;取消参数start所指向的映射地址，解除成功返回0，解除失败返回-1.

内存映射方式（Memory-mapped）

RISC指令系统的CPU（如ARM、PowerPC等）通常只实现一个物理地址空间，外设I/O端口成为内存的一部分。此时，CPU可以象访问一个内存单元那样访问外设I/O端口，而不需要设立专门的外设I/O指令。  
　　但是，这两者在硬件实现上的差异对于软件来说是完全透明的，驱动程序开发人员可以将内存映射方式的I/O端口和外设内存统一看作是"I/O内存"资源。

**mmap系统调用**

void\* mmap ( void \* addr , size\_t len , int prot , int flags ,int fd , off\_t offset )

此函数负责把文件内容映射到进程的虚拟内存空间，通过对这段内存的读取和修改，来实现对文件的读取和修改，而不需要再调用read和write等操作。

**解除映射**

int munmap(void \*start,size\_t length)

功能;取消参数start所指向的映射内存，解除成功返回0，解除失败返回-1.

### 虚拟内存区域

虚拟内存区域是进程的虚拟地址空间中的一个同质区间，既具有相同特性的连续地址范围。一个进程的内存映像由下面几部分组成：程序代码、数据、bss和栈区域以及内存映射的区域。

一个进程的内存区域可以通过查看

/proc/pid/maps

映射一个设备是指把用户空间的一段地址关联到设备内存上。mmap设备方法所需要做的就是建立虚拟地址到物理地址的页表

int (\*mmap)(struct file \*,struct vm\_area\_struct \*)

**mmap页表的建立**

方法一：使用remap\_pfn\_range一次建立所有页表

方法二：使用nopage VMA方法每次建立一个页表。

构造页表的工作可由remap\_pfn\_range函数完成，原型如下：

int remap\_pfn\_range(struct

vm\_area\_struct \*vma, unsigned long

addr,unsigned long pfn, unsigned long

size, pgprot\_t prot)

## IO

**寄存器与内存**

寄存器与ram的主要区别在于寄存器操作有副作用（side effect）去读某个地址时可能导致该地址内容发生变化，比如很多设备的中断状态寄存器只要一读取，便会自动清零。

**内存与IO**

x86处理器中存在I/O空间的概念，I/O空间是相对于内存空间而言的，他们是批次独立的地址空间，在32位的X86系统中，I/O空间大小为64K,内存空间大小为4G。

**IO端口与IO内存**

当一个寄存器或者内存位于IO空间时，称其为IO端口。

当一个寄存器或者内存位于内存空间时，称其为IO内存。

**操作IO端口**

1. 申请

struct resource \*request\_region(unsigned long first,unsigned long n, const char \*name)

功能：这个函数告诉内核，你要使用从first开始的n个端口，name参数是设备的名字。如果申请成功，返回非零，失败返回NULL。

1. 访问

访问IO端口依赖于<asm/io.h>

unsigned inb(unsigned port) 读取字节端口

void outb(unsigned char byte,unsigned port) 写字节端口

1. 释放

void release\_region(unsigned long start,unsigned long n)