

特殊的宏

| 名称 | 用法 | 实际意义 |
|------------------------------|---|-------------------|
| PDX(va) | $(((((u_long)(va)) \gg 22) \& 0x03FF))$ | 获得31-22位，即一级页表项偏移 |
| PTX(va) | $(((((u_long)(va)) \gg 12) \& 0x03FF))$ | 获得21-12位，即二级页表项偏移 |
| PTE_ADDR(pte) | $((u_long)(pte) \& \sim 0xFFF)$ | 低12位清零，即为物理地址 |
| PPN(va) | $(((((u_long)(va)) \gg 12))$ | 获得高20位，即为物理页号 |
| VPN(va) | $(((((u_long)(va)) \gg 12))$ | 获得高20位，即为物理页号 |
| PADDR(kva) | | 将虚拟地址转换成物理地址 |
| KADDR(pa) | | 将物理地址转换为虚拟地址 |
| page2ppn page2pa | pp - pages page2ppn(pp) << PGSHIFT | 页表和页框号的相互转换 |
| pa2page page2kva va2pa | &pages[PPN(pa)] KADDR(page2pa(pp)) 比较复杂 | 物理地址、页表、虚拟地址的相互转换 |
| | | |

理解一下：

辨析一下：

- 将x取地址：说明需要的是x的值
- 强制类型转换：说明是把x当作地址传进去

我现在需要搞懂参数的关系

在 Gxemul 中，console设备被映射到 0x10000000，simulated IDE disk被映射到 0x13000000，实时时钟（Real-Time Clock）被映射到0x15000000，等等



去年代码

```
int time_read()
{
    u_int time=0;
    user_panic(syscall_read_dev(&time,0x15000000,4));
    user_panic(syscall_read_dev(&time,0x15000010,4));
    return time;
}

void raid0_write(u_int secno, void *src, u_int nsecs)
{
    for(u_int i=secno;i-secno<nsecs;i++)
    {
        if(i%2==0)
        {
            ide_write(1,i/2,src+(i-secno)*0x200,1);
            //ide_write(u_int diskno, u_int secno, void *src, u_int nsecs)
        }
        else
        {

```

```

        ide_write(2,i/2,src+(i-secno)*0x200,1);
    }
}

void raid0_read(u_int secno, void *dst, u_int nsecs)
{
    for(u_int i=secno;i-secno<nsecs;i++)
    {
        if(i%2==0)
        {
            ide_read(1,i/2,dst+(i-secno)*0x200,1);
            //ide_read(u_int diskno, u_int secno, void *dst, u_int nsecs)
        }
        else
        {
            ide_read(2,i/2,dst+(i-secno)*0x200,1);
        }
    }
}

```

```

* * -----*
* | device | start addr | length |
* * -----+-----+-----*
* | console | 0x10000000 | 0x20 |
* | IDE | 0x13000000 | 0x4200 |
* | rtc | 0x15000000 | 0x200 |
* * -----*
*/
// 理解一下，即：物理设备地址（PHYSADDR）映射到内存虚拟地址的偏移量
#define PHYSADDR_OFFSET ((signed int)0xA0000000)
#define DEV_CONS_PUTGETCHAR 0x0000
#define DEV_CONS_HALT 0x0010
// 理解一下，即：设备（dev）控制台（console）的地址
#define DEV_CONS_ADDRESS 0x10000000
#define DEV_CONS_LENGTH 0x0000000000000020
// kseg1偏移量（PHYSADDR_OFFSET） + 设备（dev）控制台（console）的地址 + 要输出字符串的偏移
#define PUTCHAR_ADDRESS (PHYSADDR_OFFSET + \
DEV_CONS_ADDRESS + DEV_CONS_PUTGETCHAR)
// kseg1偏移量（PHYSADDR_OFFSET） + 设备（dev）控制台（console）的地址 + 要停止（halt）输出的偏移
#define HALT_ADDRESS (PHYSADDR_OFFSET + \
DEV_CONS_ADDRESS + DEV_CONS_HALT)
void printcharc(char ch)
{
    *((volatile unsigned char *) PUTCHAR_ADDRESS) = ch;
}
void halt(void)
{
    *((volatile unsigned char *) HALT_ADDRESS) = 0;
}
void printstr(char *s)
{
    while (*s)
        printcharc(*s++);
}

```

```

void ide_read(u_int diskno, u_int secno, void *dst, u_int nsecs)
{
    // 0x200: the size of a sector: 512 bytes.
    int offset_begin = secno * 0x200;
    int offset_end = offset_begin + nsecs * 0x200;
    int offset = 0;
    u_int zero = 0;
    u_int cur_offset = 0;
    while (offset_begin + offset < offset_end) {
        // error occurred, then panic.
        // select the IDE id to read
        /* 再次复习C语言，这里就是把diskno的值写入了0xb3000010
        这里是使用&引用的方式，把 diskno 的地址传进去了，
        这样在sys_write_dev()函数里调用bcopy的时候，就会解引用，把 diskno 的值写入
        0xb3000010 */
        if (syscall_write_dev((u_int)&diskno, 0x13000010, 4) < 0)
            user_panic("ide_read panic");
        // offset
        cur_offset = offset_begin + offset;
        /* 把距离磁盘镜像基地址的偏移量 cur_offset 的值写入到 0x13000000 */
        if (syscall_write_dev((u_int)&cur_offset, 0x13000000, 4) < 0)
            user_panic("ide_read panic");
        // start read
        /* 向 0x13000020 写入0来开启读操作 */
        if (syscall_write_dev((u_int)&zero, 0x13000020, 4) < 0)
            user_panic("ide_read panic");
        // get status of last operation(read)
        u_int succ = 0;
        /* 从0x13000030中读出本次读的结果状态，并且写入到 succ */
        if (syscall_read_dev((u_int)&succ, 0x13000030, 4) < 0)
            user_panic("ide_read panic");
        if (!succ)
            user_panic("ide_read panic");
        // IDE读操作结束以后，数据已经被加载到了缓存区，所以最终要从缓存区读出数据到目标地址
        if (syscall_read_dev((u_int)(dst + offset), 0x13004000, 0x200) < 0)
            user_panic("ide_read panic");
        offset += 0x200; // 读取下一个扇区
    }
}

```

```

void ide_write(u_int diskno, u_int secno, void *src, u_int nsecs)
{
    int offset_begin = secno * 0x200;
    int offset_end = offset_begin + nsecs * 0x200;
    int offset = 0;
    u_int one = 1;
    u_int cur_offset = 0;
    // DO NOT DELETE WRITEF !!!
    // writef("diskno: %d\n", diskno);
    while (offset_begin + offset < offset_end)
    {
        // copy data from source array to disk buffer.
        // if error occur, then panic.
        /* 写磁盘需要先将要写入对应 sector 的 512 bytes 的数据放入设备缓冲 0x13004000 中
        */
        if (syscall_write_dev((u_int)(src + offset), 0x13004000, 0x200) < 0)

```

```

user_panic("ide_write panic");
// select the IDE id to write
if (syscall_write_dev((u_int)&diskno, 0x13000010, 4) < 0)
user_panic("ide_write panic");
// offset
cur_offset = offset_begin + offset;
if (syscall_write_dev((u_int)&cur_offset, 0x13000000, 4) < 0)
user_panic("ide_write panic");
// start write
/* 向 0x13000020 写入0来开启读操作 */
if (syscall_write_dev((u_int)&one, 0x13000020, 4) < 0)
user_panic("ide_write panic");
// get status of last operation(write)
u_int succ = 0;
/* 从0x13000030中读出本次写的结果状态, 并且写入到 succ */
if (syscall_read_dev((u_int)&succ, 0x13000030, 4) < 0)
user_panic("ide_write panic");
if (!succ)
user_panic("ide_write panic");
offset += 0x200; // 写入下一个扇区
}
}

```