# 块IO

- 基本概念
- Buffer_head
- Bio
- Request
- IO调度程序

## 基本概念

## 基本概念

- **块设备**：系统中能够随机访问固定大小数据片的设备称作块设备

  如：硬盘

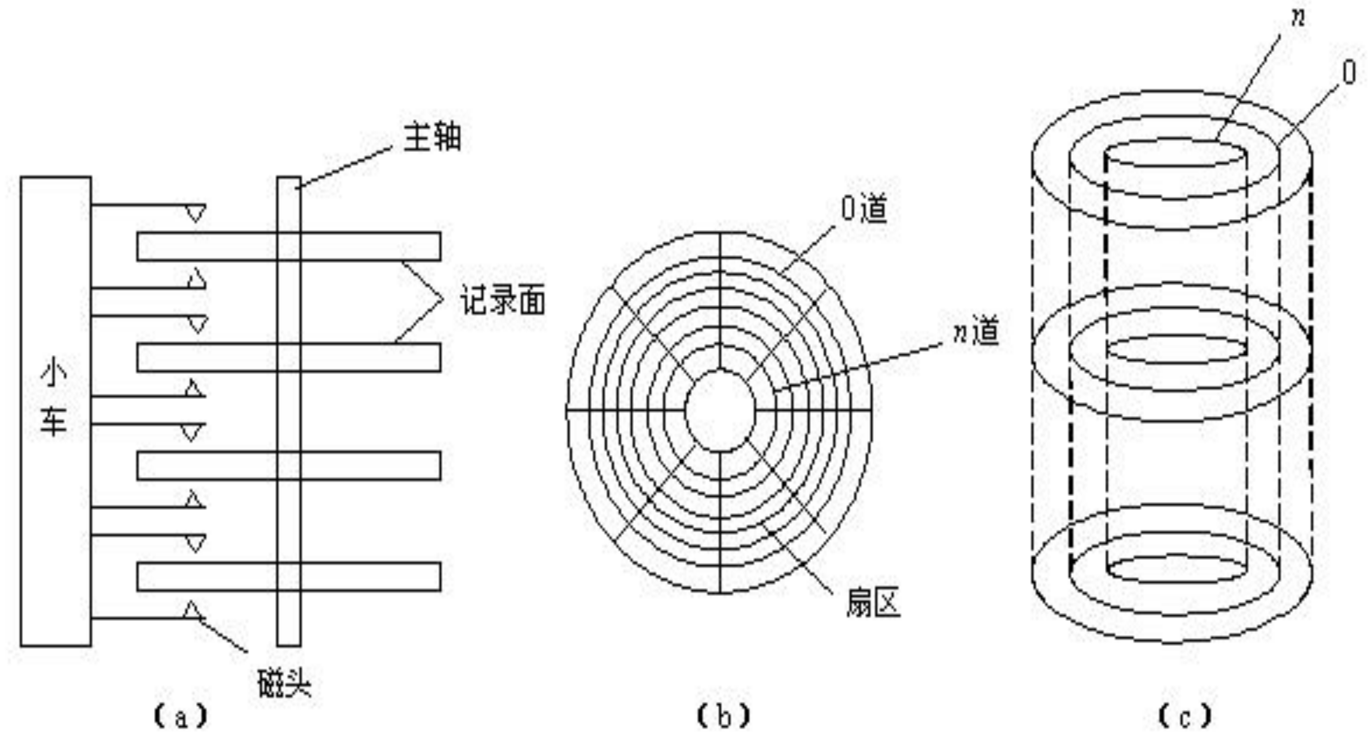
- **字符设备**：字符设备按照字符流的方式被有序访问

  如：键盘

## 基本概念

- 扇区、硬扇区、设备块
- 块、文件块、IO块
- 簇

## 基本概念

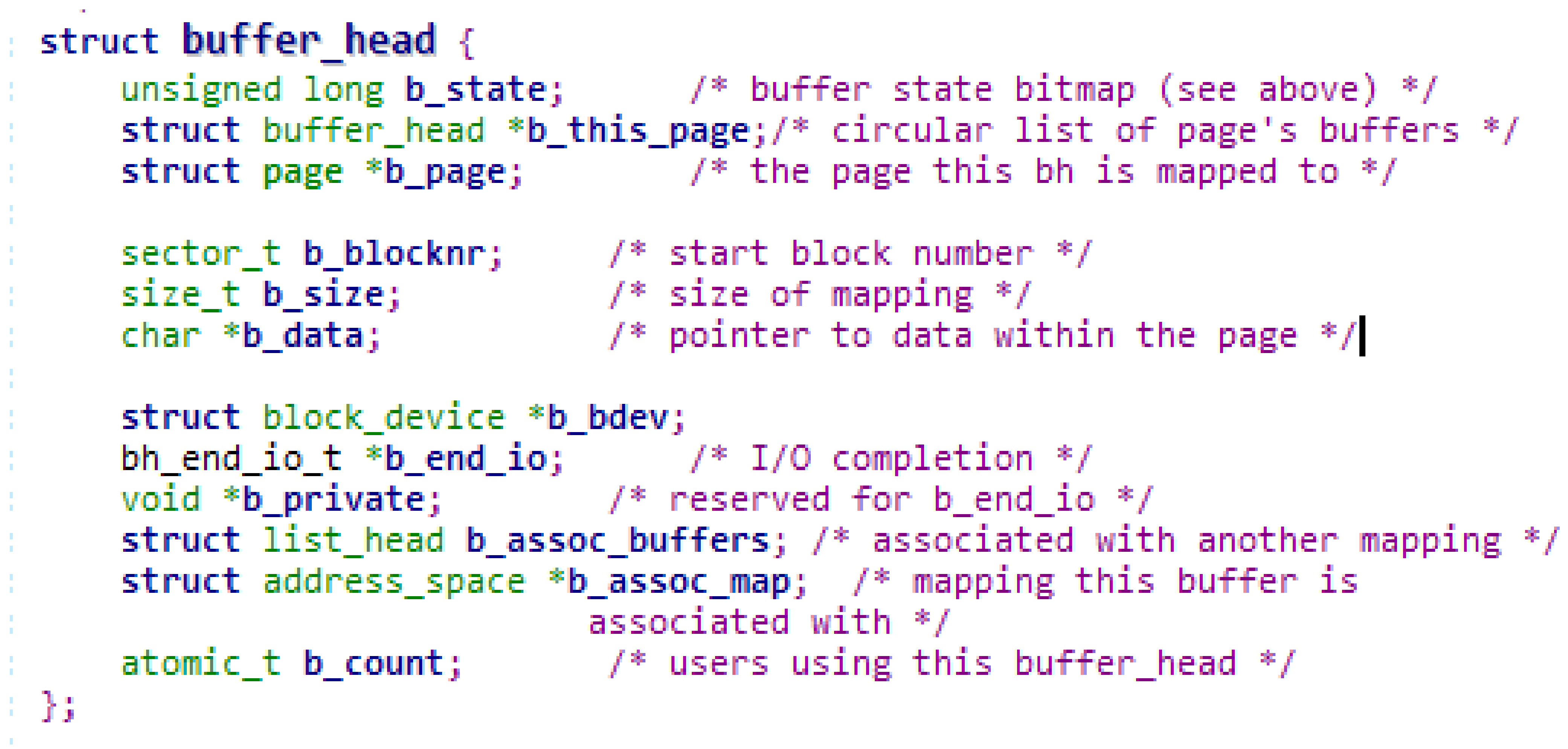


## Buffer_head

## Buffer_head

Buffer_head：描述磁盘块与物理内存缓冲区之间的映射关系

# Buffer_head



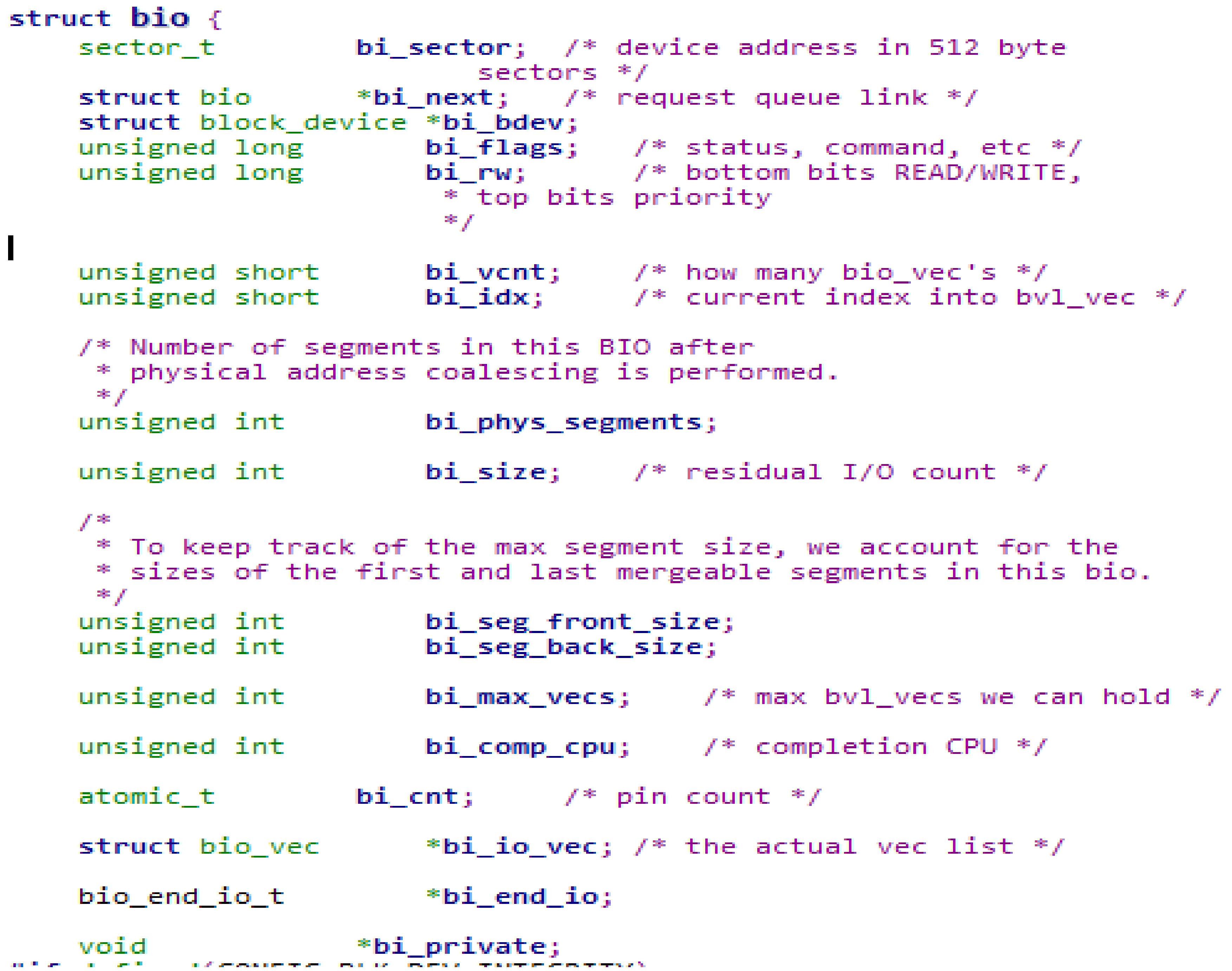
```
struct buffer_head {
    unsigned long b_state;      /* buffer state bitmap (see above) */
    struct buffer_head *b_this_page;/* circular list of page's buffers */
    struct page *b_page;        /* the page this bh is mapped to */

    sector_t b_blocknr;     /* start block number */
    size_t b_size;          /* size of mapping */
    char *b_data;           /* pointer to data within the page */

    struct block_device *b_bdev;
    bh_end_io_t *b_end_io;      /* I/O completion */
    void *b_private;        /* reserved for b_end_io */
    struct list_head b_assoc_buffers; /* associated with another mapping */
    struct address_space *b_assoc_map;  /* mapping this buffer is
                          associated with */
    atomic_t b_count;       /* users using this buffer_head */
};
```

## Bio结构体

## Bio结构体

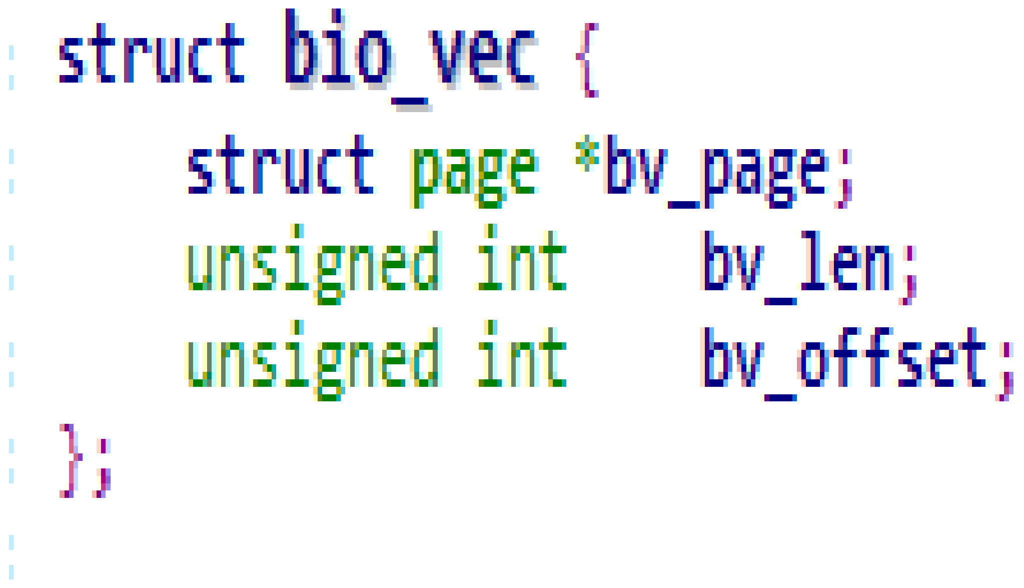
Bio结构体：内核中块IO操作的基本容器

# Bio结构体



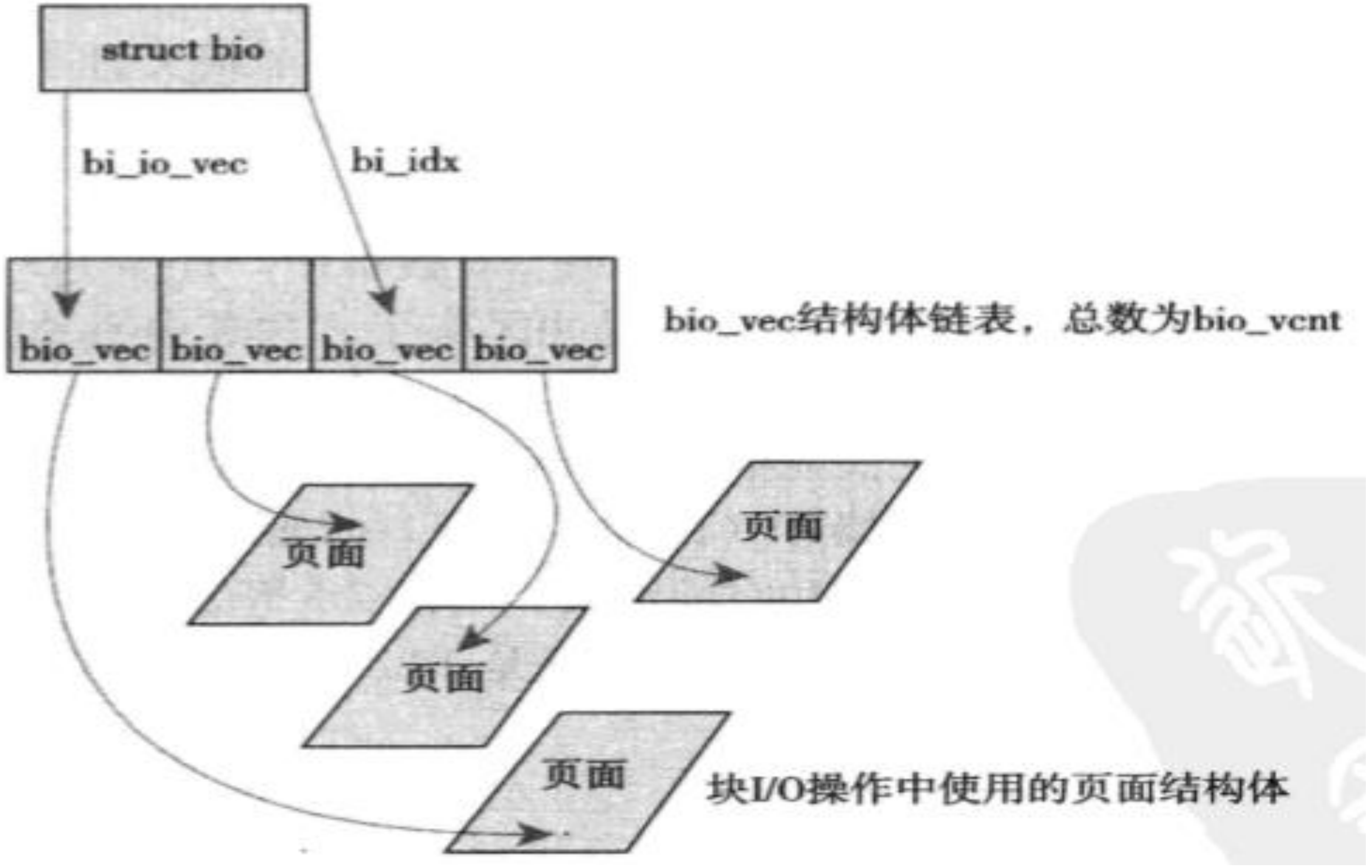
```
struct bio {
    sector_t         bi_sector;  /* device address in 512 byte
                                     sectors */
    struct bio       *bi_next;   /* request queue link */
    struct block_device *bi_bdev;
    unsigned long        bi_flags;   /* status, command, etc */
    unsigned long        bi_rw;      /* bottom bits READ/WRITE,
                          * top bits priority
                          */

    unsigned short       bi_vcnt;    /* how many bio_vec's */
    unsigned short       bi_idx;     /* current index into bvl_vec */

    /* Number of segments in this BIO after
     * physical address coalescing is performed.
     */
    unsigned int         bi_phys_segments;

    unsigned int         bi_size;    /* residual I/O count */

    /*
     * To keep track of the max segment size, we account for the
     * sizes of the first and last mergeable segments in this bio.
     */
    unsigned int         bi_seg_front_size;
    unsigned int         bi_seg_back_size;

    unsigned int         bi_max_vecs;    /* max bvl_vecs we can hold */

    unsigned int         bi_comp_cpu;    /* completion CPU */

    atomic_t         bi_cnt;     /* pin count */

    struct bio_vec       *bi_io_vec; /* the actual vec list */

    bio_end_io_t         *bi_end_io;

    void             *bi_private;
```

## Bio结构体



```c
struct bio_vec {
	struct page *bv_page;
	unsigned int	bv_len;
	unsigned int	bv_offset;
};
```

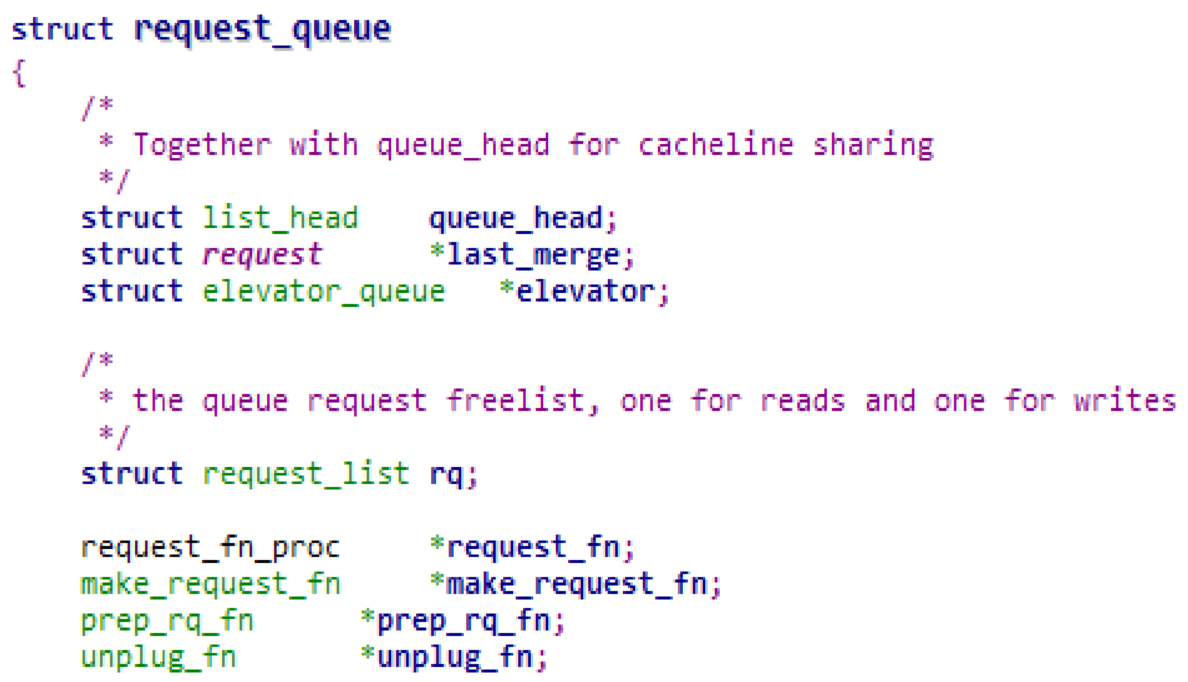
## Bio结构体



struct bio
bi_io_vec
bi_idx
bio_vec
bio_vec
bio_vec
bio_vec
bio_vec结构体链表，总数为bio_vcnt
页面
页面
页面
页面
块I/O操作中使用的页面结构体

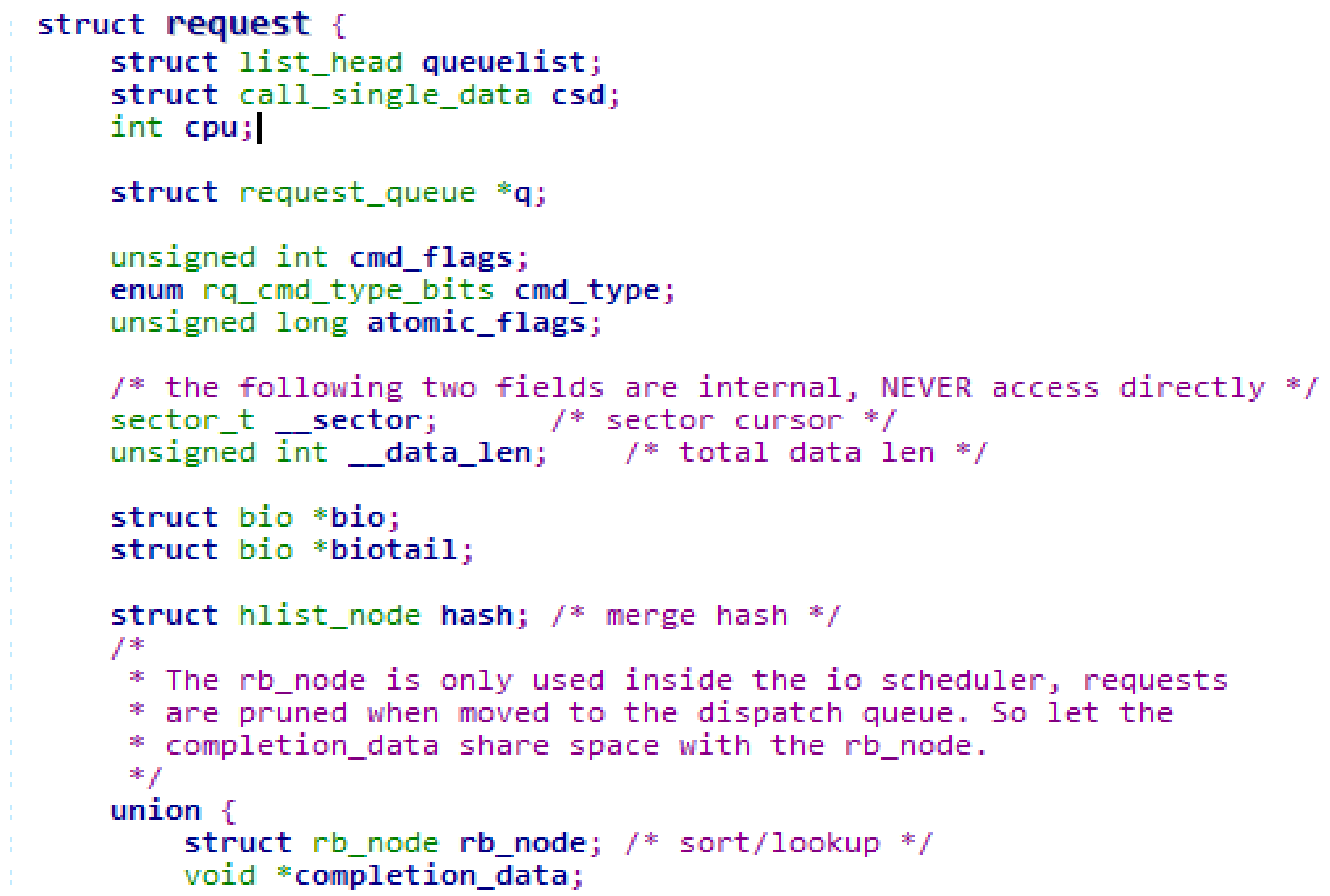
## Request

## Request



```
struct request_queue
{
    /*
     * Together with queue_head for cacheline sharing
     */
    struct list_head    queue_head;
    struct request      *last_merge;
    struct elevator_queue   *elevator;

    /*
     * the queue request freelist, one for reads and one for writes
     */
    struct request_list rq;

    request_fn_proc     *request_fn;
    make_request_fn     *make_request_fn;
    prep_rq_fn      *prep_rq_fn;
    unplug_fn       *unplug_fn;
```

## Request



```c
struct request {
    struct list_head queuelist;
    struct call_single_data csd;
    int cpu;

    struct request_queue *q;

    unsigned int cmd_flags;
    enum rq_cmd_type_bits cmd_type;
    unsigned long atomic_flags;

    /* the following two fields are internal, NEVER access directly */
    sector_t __sector;      /* sector cursor */
    unsigned int __data_len;    /* total data len */

    struct bio *bio;
    struct bio *biotail;

    struct hlist_node hash; /* merge hash */
    /*
     * The rb_node is only used inside the io scheduler, requests
     * are pruned when moved to the dispatch queue. So let the
     * completion_data share space with the rb_node.
     */
    union {
        struct rb_node rb_node; /* sort/lookup */
        void *completion_data;
```

## IO调度程序

## IO调度程序

- Linus elevator
- Deadline scheduler
- Anticipatory scheduling
- Completely Fair Queuing
- Noop scheduler