Device Independent I/O (Block Special File)

- Read the following sections in the text: 5.6.2 (pp550-553), 5.6.3 (pp553-555), 5.6.4 (pp555-557)
- Refer to disk layout in figure 5-34 on p551
- Refer to the super block in Figure 5-35 on p552
- Refer to the i-node structure in Figure 5-36 on p556
- Refer to zone[ ] information in an i-n inode in Figure 5-11 on p503

Device Independent I/O (Block Special File) (cont)

- mknod - make block or character special files
- **mknod** [*OPTION*]... *NAME TYPE* [*MAJOR MINOR*]
  - Create the special file NAME of the given TYPE.
  - OPTION:
    - **-Z**, **--context**=*CONTEXT* set security context (quoted string)
    - **-m**, **--mode**=*MODE* set permission mode (as in chmod),
  - TYPE may be:
    - b create a block (buffered) special file
    - c, u create a character (unbuffered) special file
    - p create a FIFO
  - Both MAJOR and MINOR must be specified when TYPE is b, c, or u, and they must be omitted when TYPE is p.

Major Device Number and Minor Device Number

- The major device number determines which device driver process a request message is sent to
  - In Unix, the major device number determines which function is to be invoked
- The minor device number is passed in a request message to the driver process
  - In Unix, it is passed as an argument to the function to be invoked
  - It's up to the device driver process/function as to how to use the minor device number

Device Independent I/O (Block Special File) (cont)

```
struct dmap (4220 on p686)
   04220  extern struct dmap {
   04221    int _PROTOTYPE ((*dmap_opcl), (int, Dev_t, int, int) );
                //  points to gen_opcl
   04222    void _PROTOTYPE ((*dmap_io), (int, message *) );
               // request for actually I/O (points to gen_io)
   04223    int dmap_driver;
               // the device driver process to send a request message to
   04224    int dmap_flags;
   04225  } dmap[];
```

- Initialization for a RAM driver is done at pp28235-28238
- Initialization for a FLOPPY driver is (probably) done at 28204, 28209, 28210 (process number 6 ?)
  - dm->dmap_opcl = gen_opcl;
  - dm->dmap_io = gen_io;
  - dm->dmap_driver = MEM_PROC_NR; // for the RAM driver
  - dm->dmap_flags = 0; // for the RAM driver

Open Operation on Block Special File

- do_open (24550)
  - calls common_open at 24566
- In common_open (24573)
  - calls dev_open at 24640 for I_BLOCK_SPECIAL (I_CHAR_SPECIAL, too) based on i-node.i_mode information
- In dev_open (28334)
  - calls (*dp->dmap_opcl)(DEV_OPEN, dev, proc, flags) DEV_OPEN) at 28349
    - Recall dp->dmap_opcl points to "gen_opcll"
- (cont)

Open Operation on Block Special File (cont)

- In gen_opcl (28455) (note that op is DEV_OPEN)
  - identifies the dmap structure for the major device number in dp at 28466
  - calls (*dp->dmap_io)(dp->dmap_driver, &dev_mess) at 28474
    - dp->dmap_io points to "gen_io" (28575)
    - dp->dmap_driver is set to the task to send a message to (MEM_PROC_NR for the RAM driver)
    - dev_mess cantsins: op (= DEV_OPEN), minor device number, caller (the process that requested this operation) process number, and flag (=0?)
- In gen_io (28575),
  - sends a message to task_nr (=dp->dmap_driver) at 28593

Read/Write Operation on Block Special File

- do_read(25030)/do_write (25627)
  - calls read_write (25032, 25629)
- In read_write (25038)
  - calls rw_chunk at 25173) with i-node, in case of I_BLOCK_SPECIAL
- In rw_chunk (25251)
  - inode->i_zone[0] is set to dev (major and minor dev numbers) at 25278
  - then calls get_block with dev at 25303
- In get_block (22426)
  - calls rw_block (22641) with ptr to buf at 22511, where struct buf (21616) contains dev
- In rw_block (22641)
  - calls dev_io at 22661 with op = DEV_READ or DEV_WRITE (see 22660)
- (cont)

Read/Write Operation on Block Special File (cont)

- In dev_io (28406) (op = DEV_READ or DEV_WRITE, dev – major and minor device numbers)
  - identifies the dmap structure for the major device number in dp at 28420
  - calls dp->dmap_io(dp->dmap_driver, &dev_mess) at 28432, where
    - dp->dmap_io is set to "gen_io“
    - dp->dmap_driver is the process to send a request to (MEM_PROC_NR for the RAM driver)
    - dev_mess contains op (DEV_READ or DEV_WRITE), the minor device number, position, who (the requester), buffer (address), and the number of bytes
- In gen_io (28575)
  - sends a message to the process (MEM_PROC_NR for the RAM driver) at 28593

Important Data Structures (Summary)

```
struct device {    // 10857
   u64_t dv_base;   // address of the data area of the device
   u64_t dv_size;
};

typedef struct {  // 2856  client provides an array of this structure in scattered I/O
   vir_bytes iov_addr;          /* address of an I/O buffer in the client */
   vir_bytes iov_size;          /* sizeof an I/O buffer */
} iovec_t;

struct partition {  // in src/include/minix/partition.h
  u64_t base;          /* byte offset to the partition start */
  u64_t size;          /* number of bytes in the partition */
  unsigned cylinders;   /* disk geometry */
  unsigned heads;
  unsigned sectors;
};
```

sys_vircopy and sys_physcopy (p23)