DEVICE DRIVER AND I/O MANAGEMENT

- Isolate device-specific code in a separate module
- Easy to add new devices
- Vendors can add devices without kernel source
- Kernel code accessing devices is standard

Block devices:
- I/O in fixed-size, randomly accessible blocks.
- May contain filesystems
- accessed via buf structures and may use buffer cache

Character devices:
- arbitrary size data transfers
- buffering is optional and internal
- kernel interprets data as byte stream
- sequential access, no seek()

Device driver does not need to control a HW device. Pseudo-devices
mem: physical memory
kmem: kernel memory
null: sink
zero: generate 0
pseudo-terminals: terminal based IPC

**Device driver switches:**

```c
struct bdevsw {
    int (*d_open)();
    int (*d_close)();
    int (*d_strategy)();
    int (*d_size)();
    int (*d_ioctl)();
    int (*d_xhalt)();
    char *d_name;
    int *d_flag;
    ....
```
struct cdevsw {
    int (*d_open)();
    int (*d_close)();
    int (*d_read)();
    int (*d_write)();
    int (*d_ioctl)();
    int (*d_segmap)();
    int (*d_xpoll)();
    int (*d_xhalt)();
    char *d_name;
    int *d_flag;
}

....
}
cdevsw[];

d_open called when device is opened
d_close called when device is closed by the last referer
d_strategy common interface for read+write. Device driver can change
the ordering of requests for optimization only for block devices
d_size size of the device (i.e. Partition)
d_read read requests
d_write write requests
d_ioctl generic entry point for device configuration and control
d_segmap for memory mapped character devices, mapping device memory
to process address space (for mmap() system call) (i.e. Frame buffer)
d_xpoll check for I/O available for non-blocked I/O.
d_xhalt shutdown the device

Major and minor numbers:

All devices should uniquely be identified by these to numbers. Major number denotes
the class of device driver as the module and minor number is the device instance on
that module.

(*bdevsw[getmajor(device)].d_open)(device, ..... );

where device is a combined variable including major and minor numbers.

Device Driver Interface/Driver Kernel Interface

Common interface for developing device drivers. SVR4 in 1992. Divided into
sections:
1. data definitions needed to include
2. driver entry point routines, interrupt handling and initialization etc.
   init(), start(), open(),read(),write(),......
3. kernel routines to invoke by the driver
sleep(), kmem_alloc(), kmem_free(), allocate stream, send message etc.
4. kernel data structures to use by the driver
   buf, cred, stream data structures
5. kernel #define statements to use

Parts of the interface

Driver-kernel: Entry points and kernel support routines
Driver-HW: machine dependent. Driver and the device
Driver-boot: drivers incorporation into kernel

General recommendations:
• Should not directly access system data structures other than (4)
• Should only access fields of (4) that are in the specification. Others may change
• Should not use arrays of (4). Sizes of structures may change.
• Should only clear/set bitmask fields. Not directly assign.
• Should use (3) functions to modify (4) whenever possible
• Should declare all global variables and private routines as private.

Dynamic loading/unloading.