Embedded Systems Programming

Basic Data Structures for Device Drivers (Module 6)

Yann-Hang Lee

Arizona State University
yhlee@asu.edu
(480) 727-7507

Summer 2014

A Memory Buffer Driver Module Example

- **As a module,**
  - *__init gmem_init(void)*
  - *__exit gmem_exit(void)*
- **In *gmem_init*, create and register a character device with file operations**
  - open, release, read, and write
  - device file is then created in file system
- **User program can use device name as a file and invoke file operations on the device**
- ***inode → cdev → gmem_dev (a super-class of cdev containging device related data) → file***

(Add a code example)

Open System Call

- **The open system maps onto an open operation registered by a driver module**
  - The kernel first does some processing (e.g., finds the *inode struct* and *file struct* for the driver
  - Invokes the open operation

*int open(const char *pathname, int flags);*

*int open(const char *pathname, int flags, mode_t mode);*

// return the new file descriptor,

*int (*open) (struct inode *, struct file *);*

File Struct

- **The file structure represents an open file.**
  - a kernel structure
  - created by the kernel on open and is passed to any function that operates on the file
  - defined in <linux/fs.h>
  - *mode_t f_mode* : readable or writable
  - *loff_t f_pos*: current reading or writing position.
  - *struct file_operations *f_op*
  - *void *private_data*

```
struct file {
        struct list_head f_list;
        struct dentry *f_dentry;
        struct vfsmount *f_vfsmnt;
        struct file_operations *f_op;
        atomic_t f_count;
        unsigned int f_flags;
        mode_t f_mode;
        loff_t f_pos;
        unsigned long f_reada, f_ramax,
                f_raend, f_ralen, f_rawin;
        struct fown_struct f_owner;
        unsigned int f_uid, f_gid;
        int f_error;
        unsigned long f_version;

        void *private_data;
};
```

inode

- **Each object in the filesystem is represented by an inode**
  - File type (executable, block special etc), permissions (read, write etc), Owner, Group, File Size,
  - File access, change and modification time, deletion time Number of links (soft/hard), extended attribute such as append only
  - Access Control List (ACLs)
  - Includes

    dev_t i_rdev;

    *struct block_device *i_bdev;*

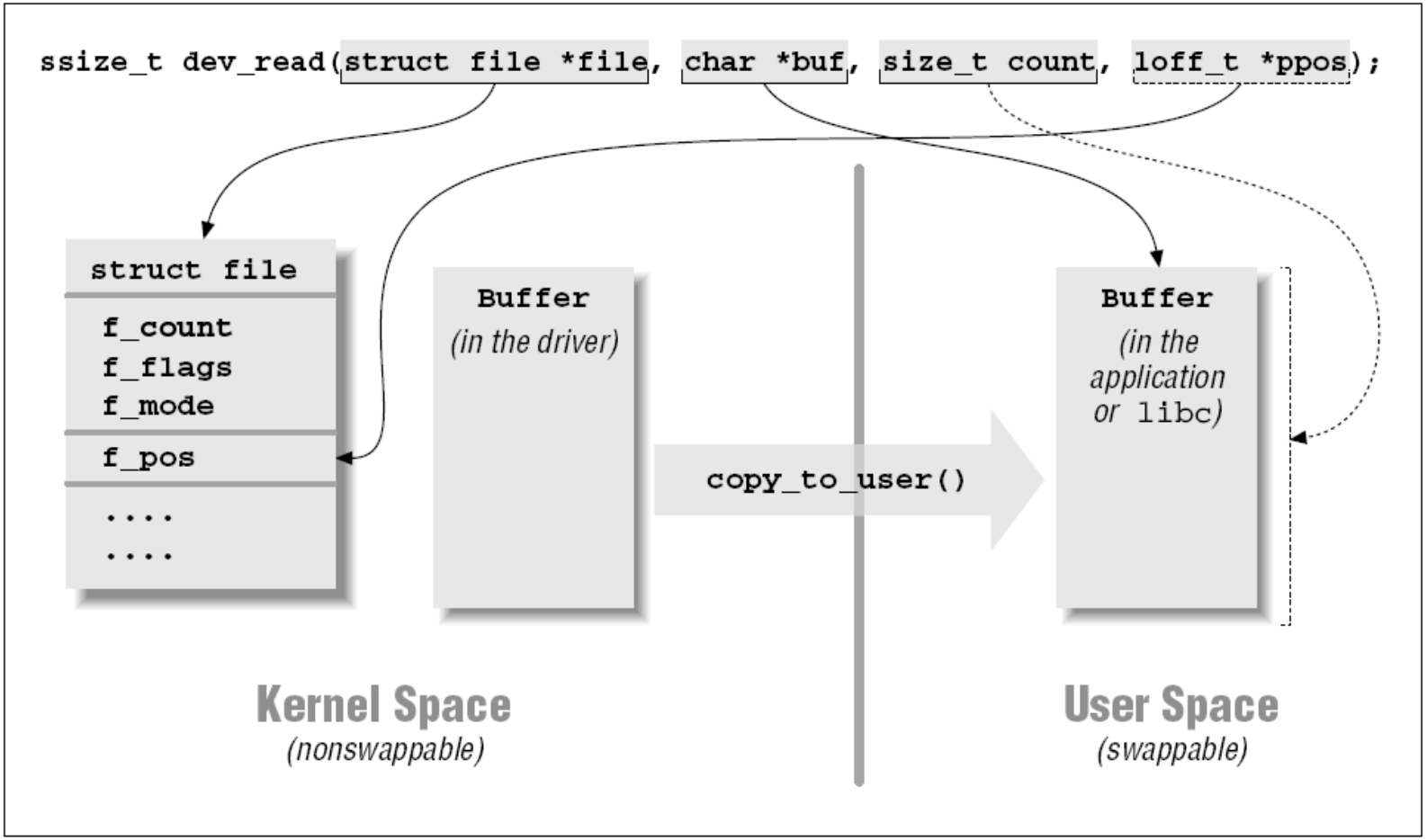
    *struct char_device *i_cdev;*

Memory Access – User and Kernel Spaces

- **You might receive some pointers that point to user space in your driver function**
  - Do not dereference it directly
  - Access it via kernel functions (<asm/uaccess.h>): *copy_to_user() and copy_from_user()*
    - They would check whether the user space pointer is valid

```
unsigned long copy_to_user(
                    void __user *to,
                    const void *from,
                    unsigned long count);
unsigned long copy_from_user(
                    void *to,
                    const void __user *from,
                    unsigned long count);
```

The Read/Write Method



Container_of

- If *strcut x* contains *y*, can we find the pointer to *x* given the pointer to *y*
- *defined in <linux/kernel.h>:*

  container_of(pointer, container_type, container_field);
- takes a pointer to a field of type container_field, within a structure of type container_type,
- returns a pointer to the containing structure.