Character Drivers

What to Expect?

- After this session, you would know
  - W's of Character Drivers
  - Major & Minor Numbers
  - Registering & Unregistering Character Driver
  - File Operations of a Character Driver
  - Writing a Character Driver
  - Linux Device Model
  - udev & automatic device creation

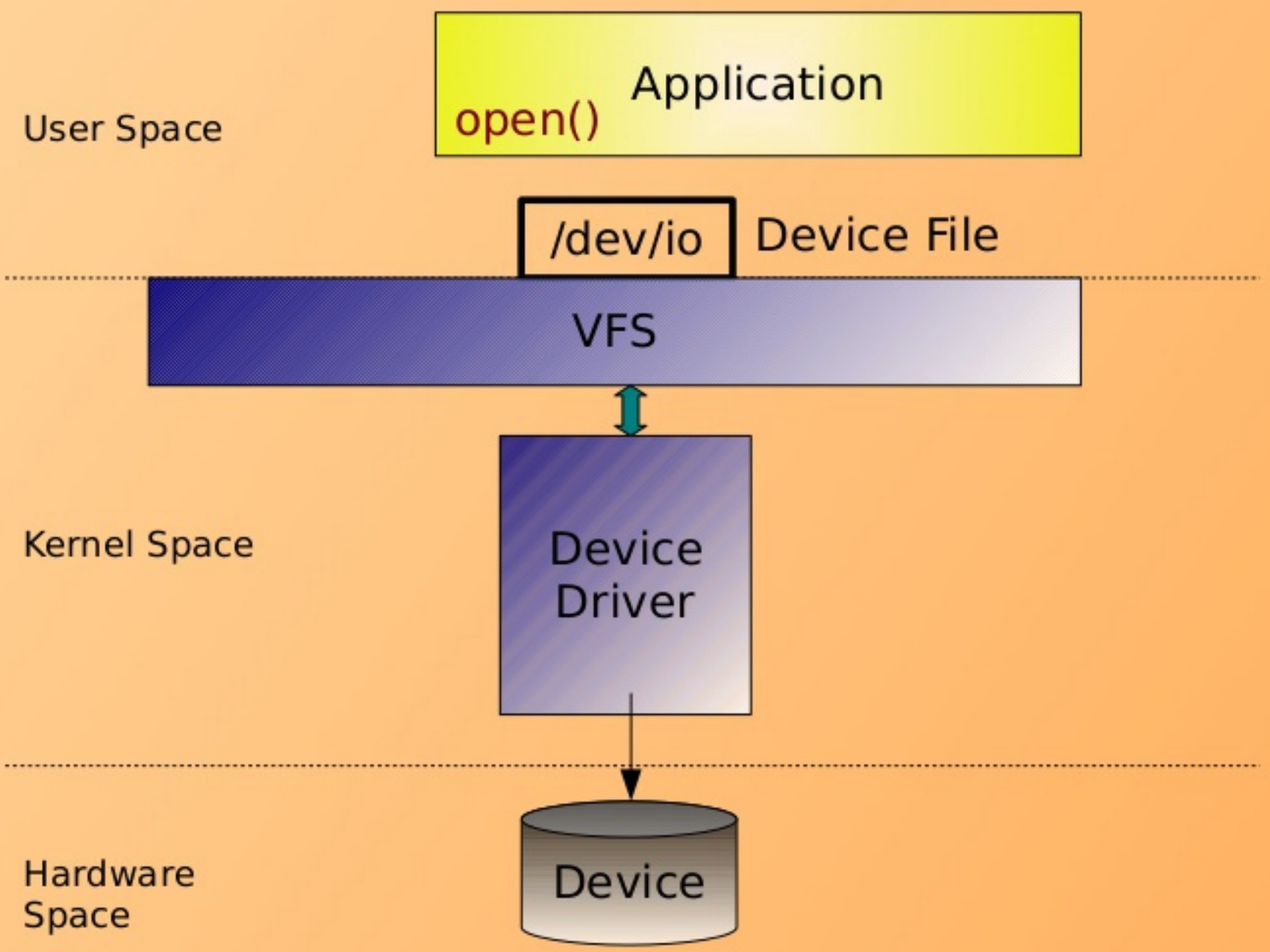
W's of Character Drivers

- What does “Character” stand for?
- Look at entries starting with 'c' after
  - ls -l /dev
- Device File Name
  - User Space specific
  - Used by Applications
- Device File Number
  - Kernel Space specific
  - Used by Kernel Internals as easy for Computation

Major & Minor Number

- ls -l /dev
- Major is to Category; Minor is to Device
- Data Structures described in Kernel C in object oriented fashion
- Type Header: <linux/types.h>
  - Type: dev_t – 12 bits for major & 20 bits for minor
- Macro Header: <linux/kdev_t.h>
  - MAJOR(dev_t dev)
  - MINOR(dev_t dev)
  - MKDEV(int major, int minor)

3 Entities in 3 Spaces



Registering & Unregistering

- Registering the Device Driver
  - int register_chrdev_region(dev_t first, unsigned int count, char *name);
  - int alloc_chrdev_region(dev_t *dev, unsigned int firstminor, unsigned int cnt, char *name);
- Unregistering the Device Driver
  - void unregister_chrdev_region(dev_t first, unsigned int count);
- Header: <linux/fs.h>
- Kernel Window: /proc/devices

The file operations

- struct file_operations
  - struct module owner = THIS_MODULE; /* <linux/module.h> */
  - int (*open)(struct inode *, struct file *);
  - int (*release)(struct inode *, struct file *);
  - ssize_t (*read)(struct file *, char __user *, size_t, loff_t *);
  - ssize_t (*write)(struct file *, const char __user *, size_t, loff_t *);
  - loff_t (*llseek)(struct file *, loff_t, int);
  - int (*unlocked_ioctl)(struct file *, unsigned int, unsigned long);
- Header: <linux/fs.h>

Initialization for Registration

- 1st way initialization
  - struct cdev *my_cdev = cdev_alloc();
  - my_cdev->owner = THIS_MODULE;
  - my_cdev->ops = &my_fops;
- 2nd way initialization
  - struct cdev my_cdev;
  - cdev_init(&my_cdev, &my_fops);
- Header: <linux/cdev.h>

Registering the file operations

- The Registration
  - int cdev_add(struct cdev *cdev, dev_t num, unsigned int count);
- The Unregistration
  - void cdev_del(struct cdev *cdev);
- Header: <linux/cdev.h>

The file & inode structures

- Important fields of struct file
  - mode_t f_mode
  - loff_t f_pos
  - unsigned int f_flags
  - struct file_operations *f_op
  - void *private_data
- Important fields of struct inode
  - unsigned int iminor(struct inode *);
  - unsigned int imajor(struct inode *);

Register/Unregister: Old Way

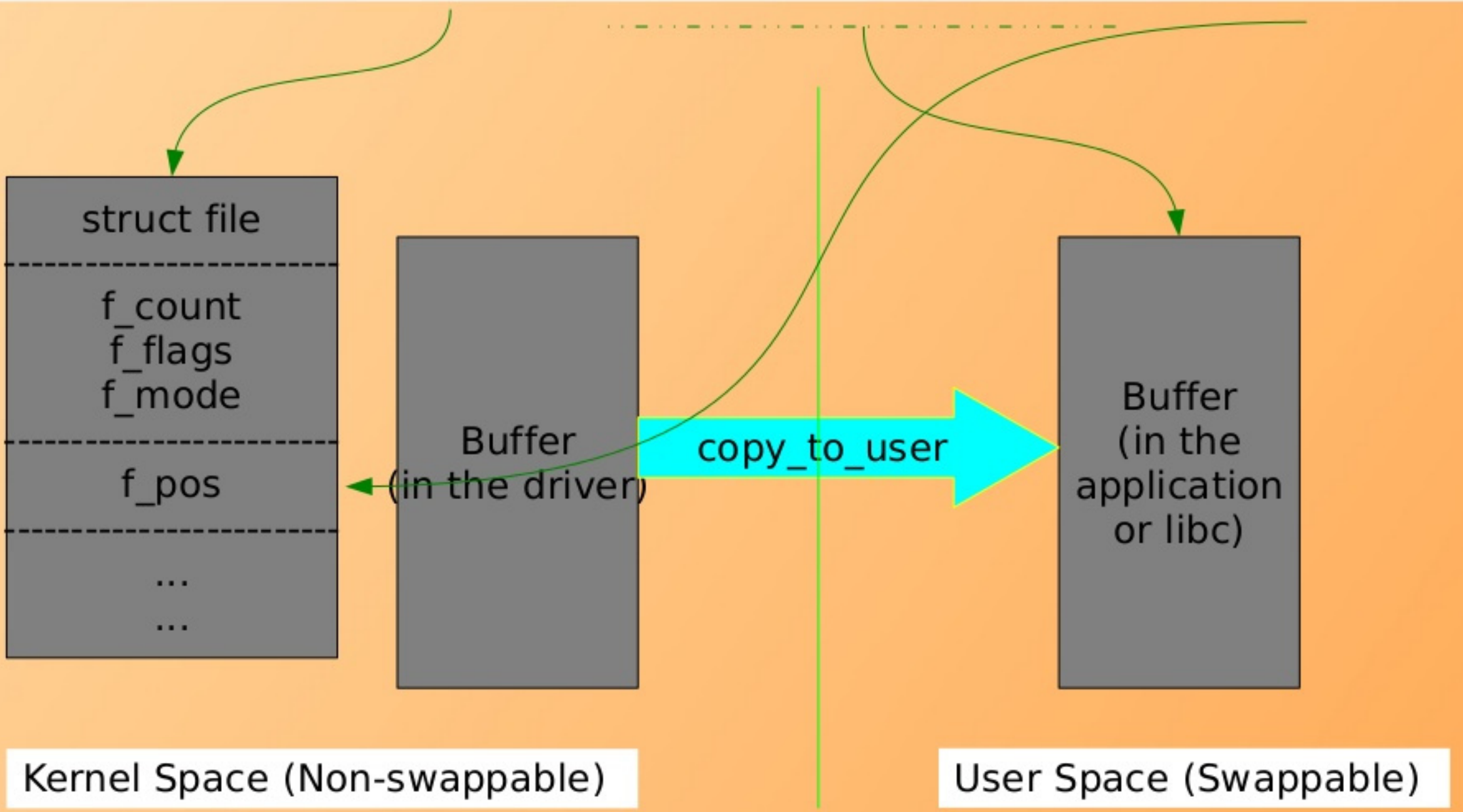
- Registering the Device Driver
  - int register_chrdev(unsigned int major, const char *name, struct file_operations *fops);
- Unregistering the Device Driver
  - int unregister_chrdev(unsigned int major, const char *name);

The /dev/null read & write

```
ssize_t my_read(struct file *f, char __user *buf, size_t cnt, loff_t *off)
{
        ...
        return read_cnt;
}
ssize_t my_write(struct file *f, char __user *buf, size_t cnt, loff_t *off)
{
        ...
        return wrote_cnt;
}
```

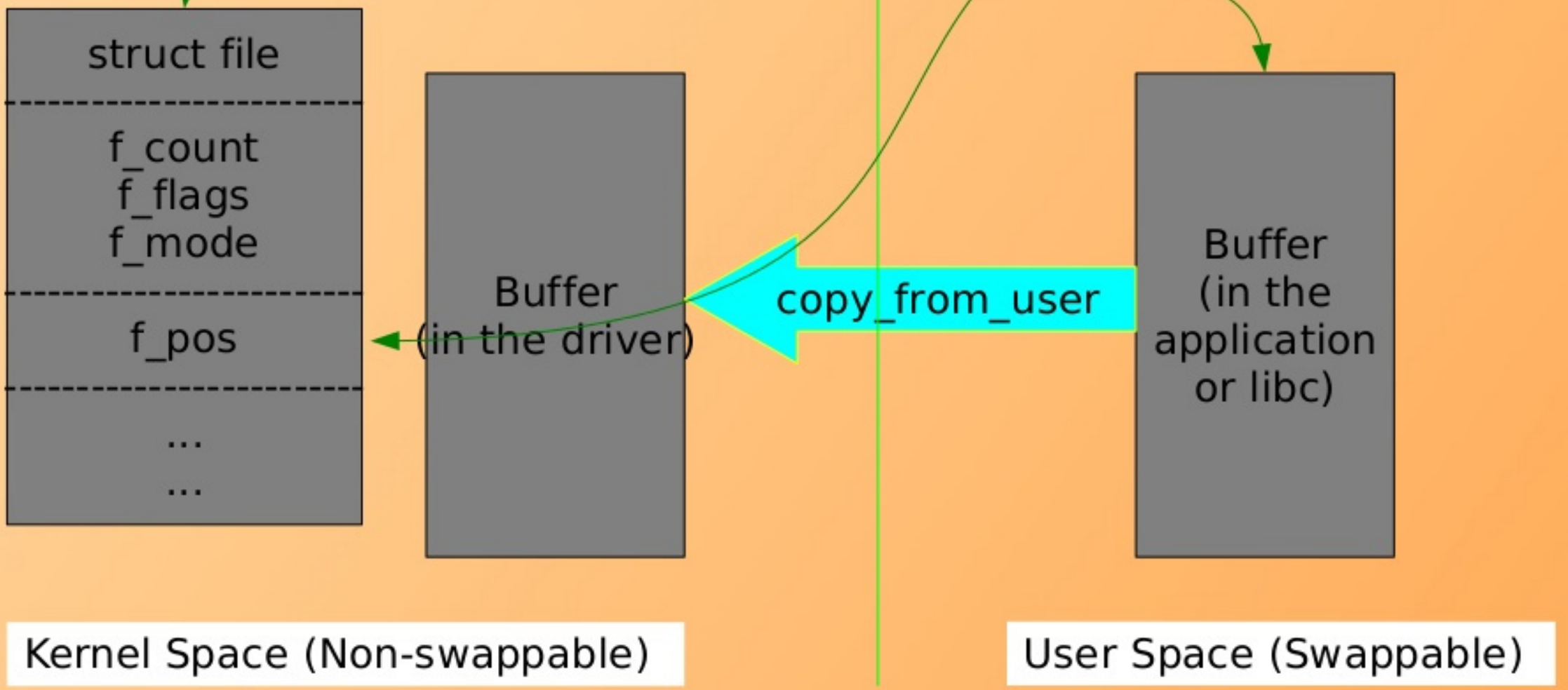
The read flow

ssize_t my_read(struct file *f, char __user *buf, size_t cnt, loff_t *off)



The write flow

ssize_t my_write(struct file *f, const char __user *buf, size_t cnt, loff_t *off)



The mem device read

```
#include <asm/uaccess.h>


ssize_t my_read(struct file *f, char __user *buf, size_t cnt, loff_t *off)
{
        ...
        if (copy_to_user(buf, from, cnt) != 0)
        {
                return -EFAULT;
        }
        ...
        return read_cnt;
}
```

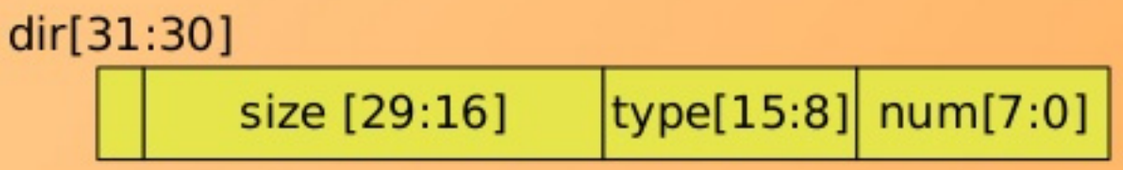
The mem device write

```
#include <asm/uaccess.h>

ssize_t my_write(struct file *f, const char __user *buf, size_t cnt, loff_t *off)
{
        ...
        if (copy_from_user(to, buf, cnt) != 0)
        {
                return -EFAULT;
        }
        ...
        return wrote_cnt;
}
```

The I/O Control API

- API
  - int (*unlocked_ioctl)(struct file *, unsigned int cmd, unsigned long arg)
- Command
  - Macros
    - _IO, _IOW, _IOR, _IOWR
  - Parameters
    - type (character) [15:8]
    - number (index) [7:0]
    - size (param type) [29:16]



- Header: <linux/ioctl.h> →...→ <asm-generic/ioctl.h>

Linux Device Model (LDM)

- struct kobject - <linux/kobject.h>
  - kref object
  - Pointer to kset, the parent object
  - kobj_type, type describing the kobject
- kobject instantiation → sysfs representation
- Parent object guides the entries under /sys/
  - bus – the physical buses
  - class – the device categories
  - device – the actual devices

udev & LDM

- Daemon: udevd
- Configuration: /etc/udev/udev.conf
- Rules: /etc/udev/rules.d/
- Utility: udevinfo [-a] [-p <device_path>]
- Receives uevent on a change in /sys
- Accordingly, updates /dev &/or
- Performs the appropriate action for
  - Hotplug
  - Microcode / Firmware Download
  - Module Autoload

Device Model & Classes

- Latest way to create dynamic devices
  - Create or Get the appropriate device category
  - Create the desired device under that category
- Class Operations
  - struct class *class_create(struct module *owner, char *name);
  - void class_destroy(struct class *cl);
- Device into & out of Class
  - struct class_device *device_create(struct class *cl, NULL, dev_t devnum, NULL, const char *fmt, ...);
  - void device_destroy(struct class *cl, dev_t devnum);

What all have we learnt?

- W's of Character Drivers
- Major & Minor Numbers
- Registering & Unregistering Character Driver
- File Operations of a Character Driver
- Writing a Character Driver
- Linux Device Model
- udev & automatic device creation

Any Queries?