Character Device Drivers: An Introduction

ECE 373

Prelims

- Questions on homework or reading assignments?
- Questions on class?

Char drivers

- Driver for char devices
- Typical types of char drivers
  - Mice
  - Keyboards
  - Printers
  - Memory

- Special files in /dev marked with notation:
  - `crw-rw-rw-  1 root  wheel    5,  49 Apr 13 00:39 ptys1`

Char drivers

- Driver for char devices
- Typical types of char drivers
  - Mice
  - Keyboards
  - Printers
  - Memory

- Special files in /dev marked with notation:
  - `crw-rw-rw-  1 root  wheel    5,  49 Apr 13 00:39 ptys1`

Device nodes, deeper

- Major and minor numbers used to identify device
  - `crw-rw-rw-  1 root  wheel    5,  49 Apr 13 00:39 ptys1`
  - Major = module (tty, clock, serial ports, disk)
  - Minor = which specific instance (pty49)
- Kernel data type to store dev node is "`dev_t`"
- Defined in `include/linux/types.h`
- Unsigned 32-bit number, packed
  - 20-bits of minor, 12-bits of major
- Current list in `/proc/devices`

Manipulating dev nodes

- Macros used to extract numbers from dev_t
  - `MAJOR(dev_t dev)`
  - `MINOR(dev_t dev)`
- Found in `include/linux/kdev_t.h`
- Ensures device representation is portable
- Example 1!

Registering a new char device

- First, allocate a region
  - `int register_chrdev_region(dev_t first, unsigned int count, char *name)`
- Uses predetermined device nodes
- Not the right way to do most things anymore...
- Requires mknod to be used
  - man mknod
- Example 2!

Properly registering char device

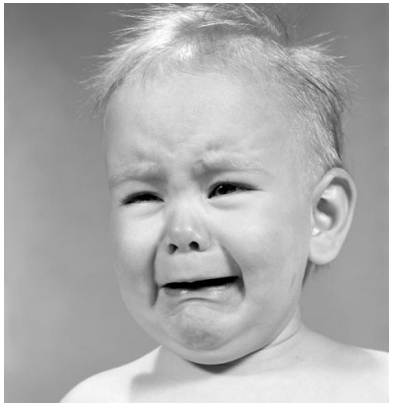
- Request a region to be allocated for you
  - `int alloc_chrdev_region(dev_t *dev,`
    `unsigned int firstminor,`
    `unsigned int count,`
    `char *name)`
- Auto-fills "`dev`" with device nodes
- "`firstminor`" typically 0, can be anything

Cleaning up after your char device

- With any dynamic stuff, it must be cleaned up
  - `void unregister_chrdev_region(dev_t first, unsigned int count)`
- Destroys internal kernel references
- Resource "leak" if not called
- Example 3!

It's there, I want to use it!

- Well, module is pretty dumb right now...
- Device needs to be connected
- Use mknod to create device '`/dev/ece`'
  - `man mknod`...
- Can you read and write it?



Why didn't it work?

- `alloc_chrdev_region()` just configures device internal to kernel
- No linkage to anything in the upper device subsystem
- System calls and driver callbacks

Beginning to hook it all up

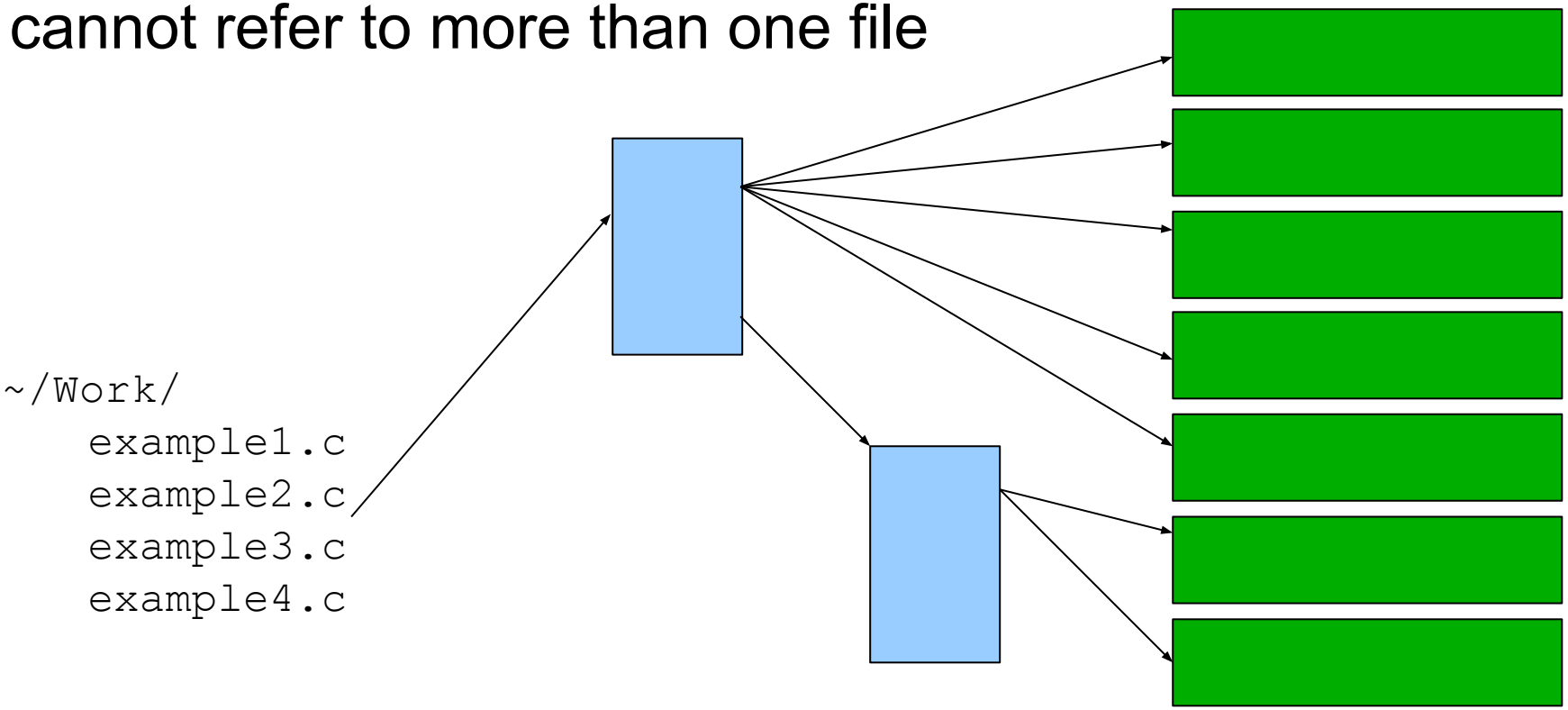
- Structure "`file_operations`" provides function pointers into system call interface
- Main linkage into `/dev` filesystem for char drivers
- Driver does not need to implement all of them
- Behaves similarly to object-oriented code

Snippet of `file_operations` API

```
struct file_operations {

    struct module *owner;

    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 *);

}
```

Files and inodes

- Internal kernel structures to manage "files"
- Opened files are managed by "`struct file`" internal to kernel
- Inode is the describer of a physical file on disk
- Many directory entries can reference single inode, inode cannot refer to more than one file



The cdev!

- Yes, another structure to worry about
- Struct that represents char devices inside kernel
- Initialized with:
  - `void cdev_init(struct cdev *cdev, struct file_operations *fops)`
- Added with:
  - `int cdev_add(struct cdev *cdev, dev_t num, unsigned int count)`
- Cleaned up with:
  - `void cdev_del(struct cdev *cdev)`

Last minute safety checks

- `file_operations` must be configured and ready to go before `cdev_add()`
- Could run into NULL pointer exceptions
- Kernel will go boom if this is misconfigured
- Example 4!

Passing data through

- System call only passes data
- Need mechanism to copy data into kernel buffers, out of kernel buffers
- Two handy-dandy functions:
  - `copy_from_user`
  - `copy_to_user`
- Example 5!

More info

- Linux Device Drivers, 3rd Edition
  - Chapter 3, Pages 42 – middle of page 57
- Essential Linux Device Drivers
  - Chapter 5, Pages 119 – top of page 129

Coming soon...

- Hardware access capabilities
- I/O ports, MMIO, PCI
- Readings:
  - LDDR Chapter 9, Chapter 12
  - ELDD Chapter 10