CSCI-3753: Operating Systems Fall 2019

Abigail Fernandes

Department of Computer Science

University of Colorado Boulder

Announcement

- Time slot registration for PA1 interview !!!
- What to prepare for the interview?
  - Laptop
  - Submitted source code
  - Knowledge … for example
    - Bootloader?
    - System call?
    - Files you modified to add your own system call?
    - Used functions
- No office hours next week

Week 3
LKM Example
Device Driver

LKM command – ***insmod***

- ***insmod*** makes an init_module system call to load the LKM into kernel memory.
- init_module system call invokes the LKM's initialization routine (named module_init) right after it loads the LKM.
- ***insmod*** passes to init_module the address of the subroutine in the LKM named module_init as its initialization routine.

LKM command – ***rmmod***

- ***rmmod*** makes an delete_module system call to unload the LKM into kernel memory.
- delete_module system call invokes the LKM's cleanup routine (named module_exit) right before it unloads the LKM.
- ***rmmod*** passes to delete_module the address of the subroutine in the LKM named module_exit as its cleanup routine.

Create, add, and remove an LKM

Step 1: Create hello.c

hello.c

```c
// Defining __KERNEL__ and MODULE allows us to access kernel-level code not usually
#undef __KERNEL__
#define __KERNEL__

#undef MODULE
#define MODULE

// Linux Kernel/LKM headers: module.h is needed by all modules and kernel.h is need
#include <linux/module.h>    // included for all kernel modules
#include <linux/kernel.h>    // included for KERN_INFO
#include <linux/init.h>        // included for __init and __exit macros

MODULE_LICENSE("GPL");

static int __init hello_init(void)
{
    printk(KERN_ALERT "Hello world!\n");
    return 0;    // Non-zero return means that the module couldn't be loaded.
}


static void __exit hello_cleanup(void)
{
    printk(KERN_ALERT "Cleaning up module.\n");
}

module_init(hello_init);
module_exit(hello_cleanup);

```

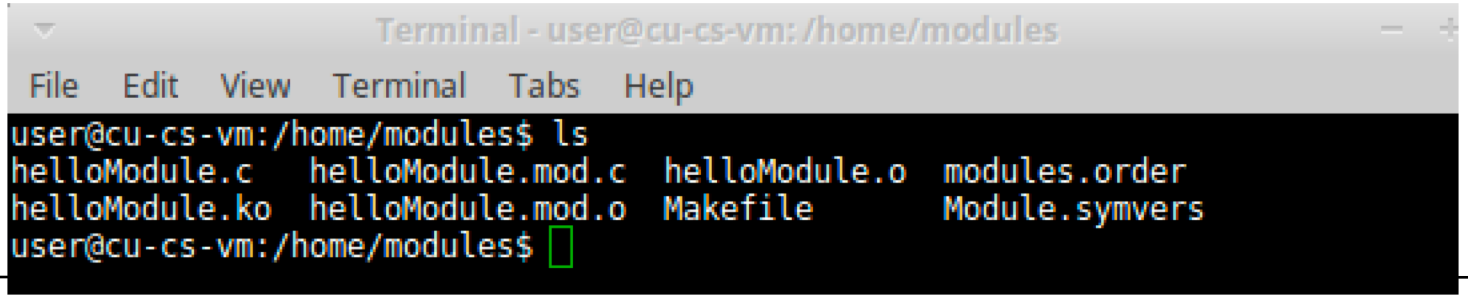
Create, add, and remove an LKM

Step 2: Create the Makefile

Add `obj-m := hello.o`

Step 3: Compile the code

sudo make -C /lib/modules/$(uname -r)/build M=$PWD modules



Create, add, and remove an LKM

Step 3: Insert the compiled module into the running kernel

sudo insmod hello.ko

```
user@cu-cs-vm:/home/modules$ lsmod | grep hello
helloModule            16384  0
user@cu-cs-vm:/home/modules$
```

Use **dmsg** to check if hello_init() was executed

```
user@cu-cs-vm:/home/modules$ dmesg | grep hello
[ 3984.350417] helloModule: loading out-of-tree module taints kernel.
[ 3984.351186] helloModule: module license 'unspecified' taints kernel.
[ 3984.358582] helloModule: module verification failed: signature and/or require
d key missing - tainting kernel
[ 3984.366265] inside hello init function
```

Create, add, and remove an LKM

Step 4: Remove the module when you're finished

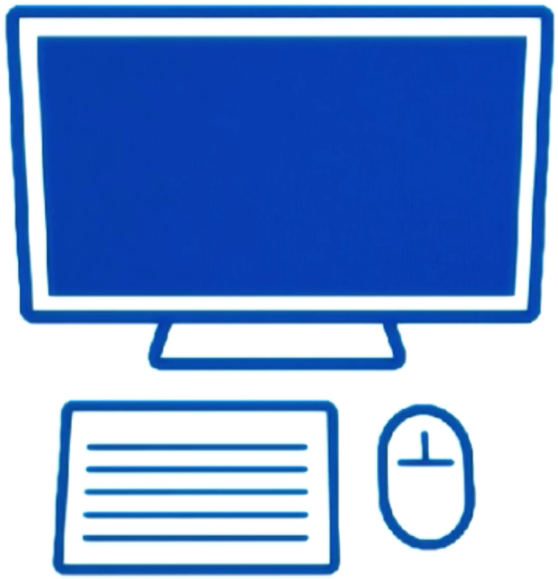
sudo rmmod hello

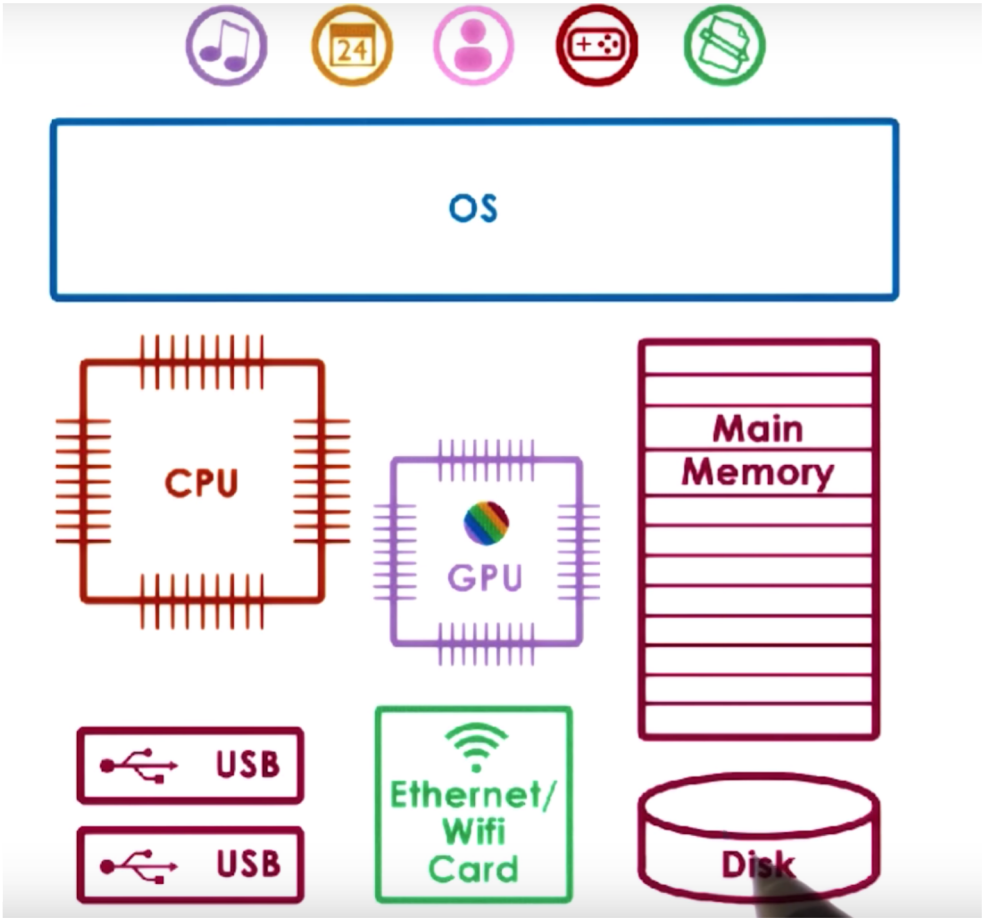
```
user@cu-cs-vm:/home/modules$ lsmod | grep hello
user@cu-cs-vm:/home/modules$
```

Use dmsg to check if hello_exit() was executed

```
user@cu-cs-vm:/home/modules$ dmesg | grep hello
[ 3984.350417] helloModule: loading out-of-tree module taints kernel.
[ 3984.351186] helloModule: module license 'unspecified' taints kernel.
[ 3984.358582] helloModule: module verification failed: signature and/or require
d key missing - tainting kernel
[ 3984.366265] inside hello_init function
[ 4507.533429] inside hello_exit function
```

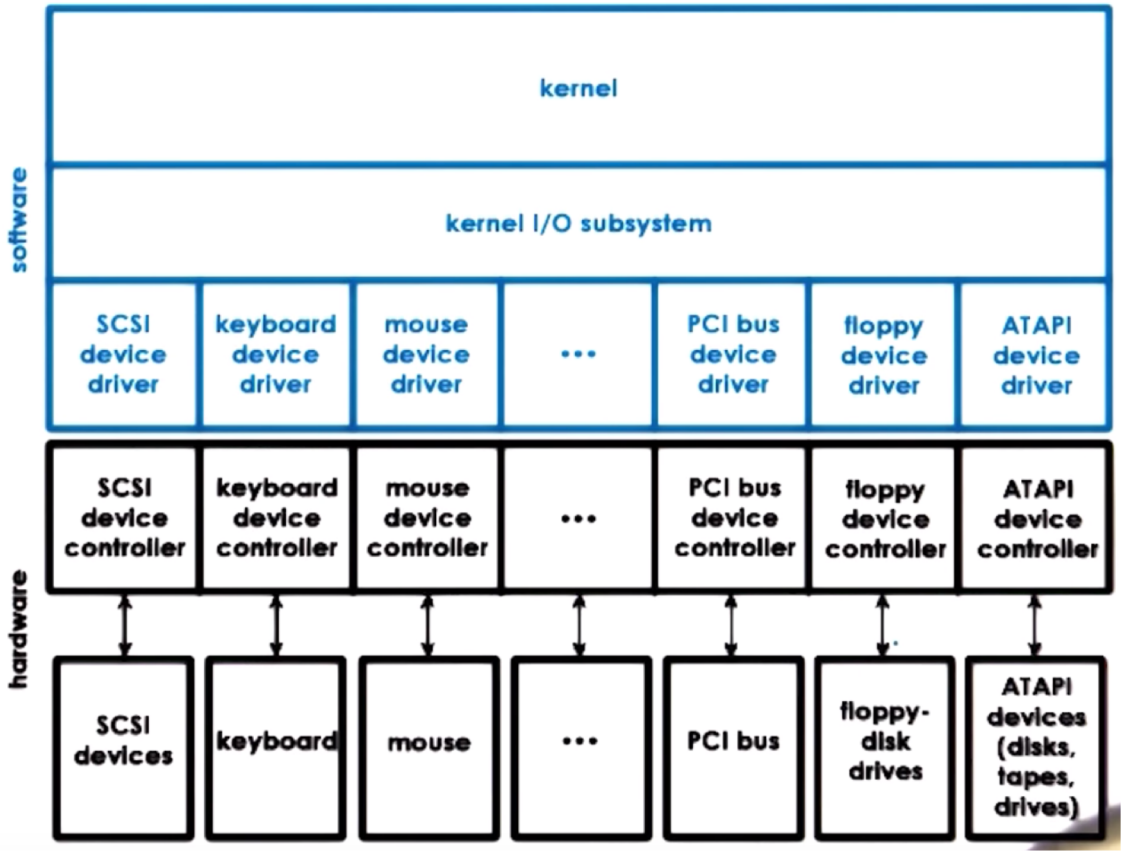
IO Devices





Device Drivers

- Device Specific
- Responsible for device access, management and control
- Provided by device manufacturers per OS/version



Types of Devices

Block (Disk)

- read/write blocks of data
- direct access to arbitrary block

Character (keyboard)

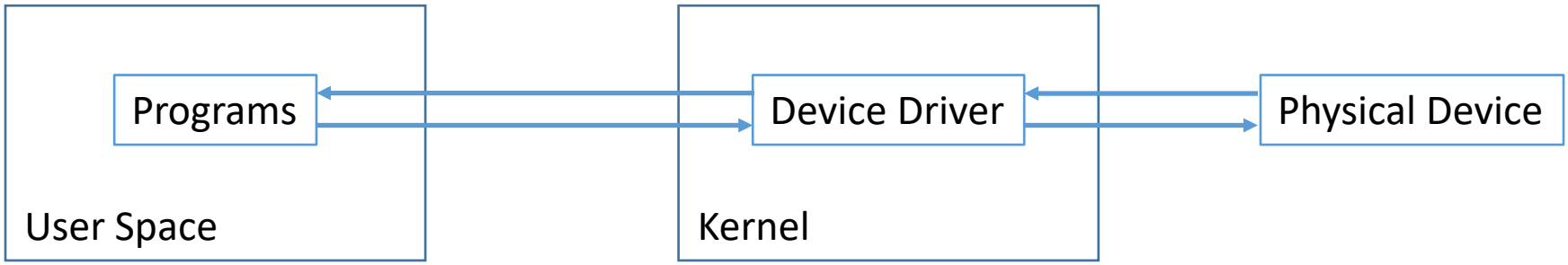
- get/put character

Network devices

Device Driver

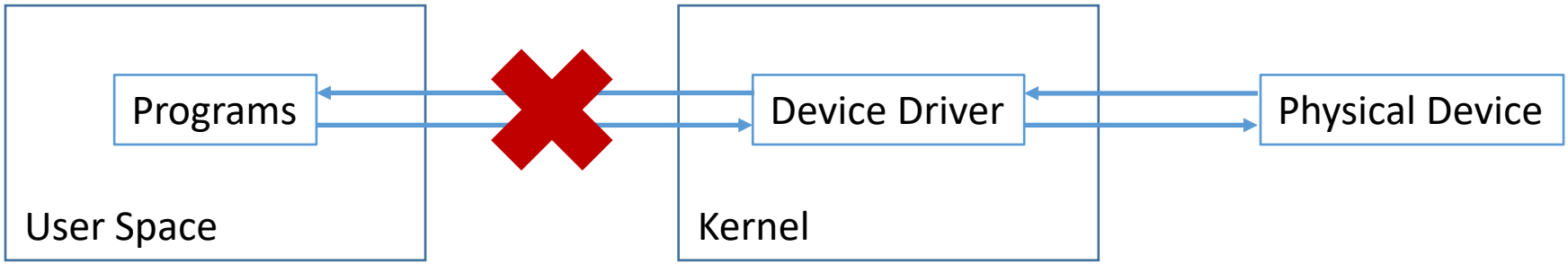
A device driver is a program that

- Is in the kernel
- Operates or controls a particular type of device that is attached to a computer.
- Transfers data to and from a user process
- Communicates with the device through the computer bus or communications subsystem to which the hardware connects.



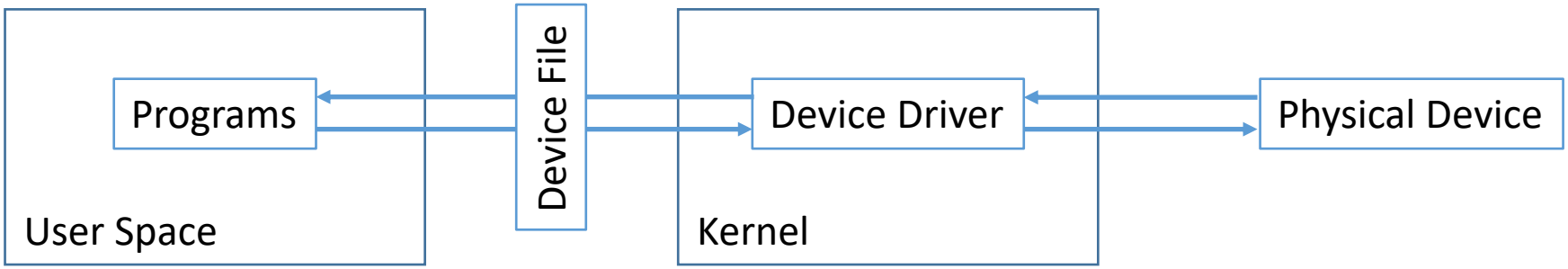
Device Driver

- When a calling program invokes a routine in the driver, the driver issues commands to the device.
- Once the device sends data back to the driver, the driver may invoke routines in the original calling program.
- A program cannot access the driver in the kernel directly.

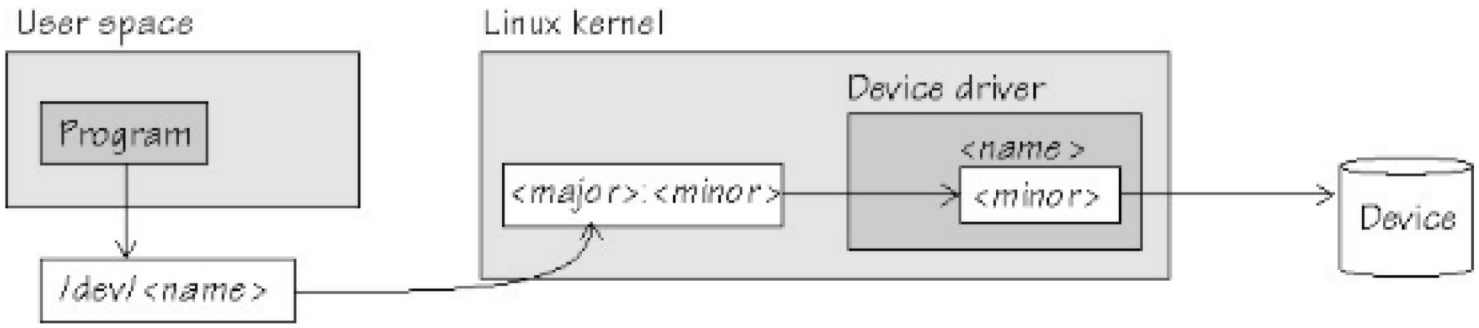


Device File

- A device file is
  - An interface between programs and the device driver.
  - Usually found under the /dev directory
  - Also called device nodes



Device files



Device File

A device file can represent

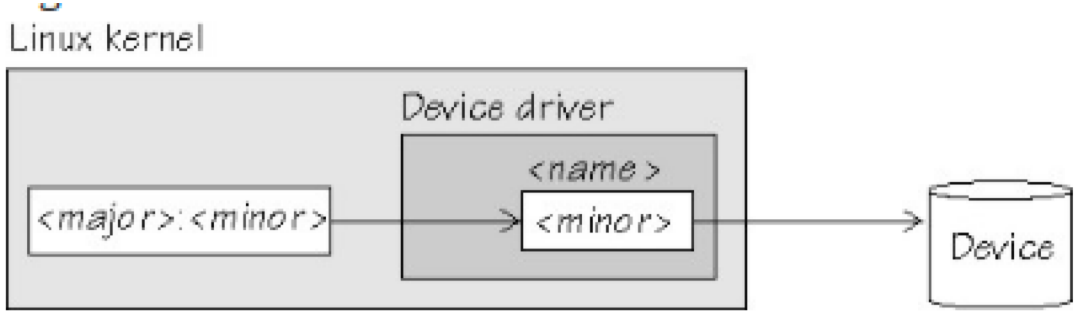
- **character devices**, which emit a stream data one character at a time (e.g., mouse)
- **block devices** which allow random access to blocks of data (e.g., hard disk)

Device File (Major and Minor numbers)

The Linux kernel represents character and block devices as pairs of numbers **<major>:<minor>.**

Major number: identifies a device driver that should be called

Minor number: identifies a particular device the driver controls.



Your turn

Run the command ls –la /dev in a Linux Environment

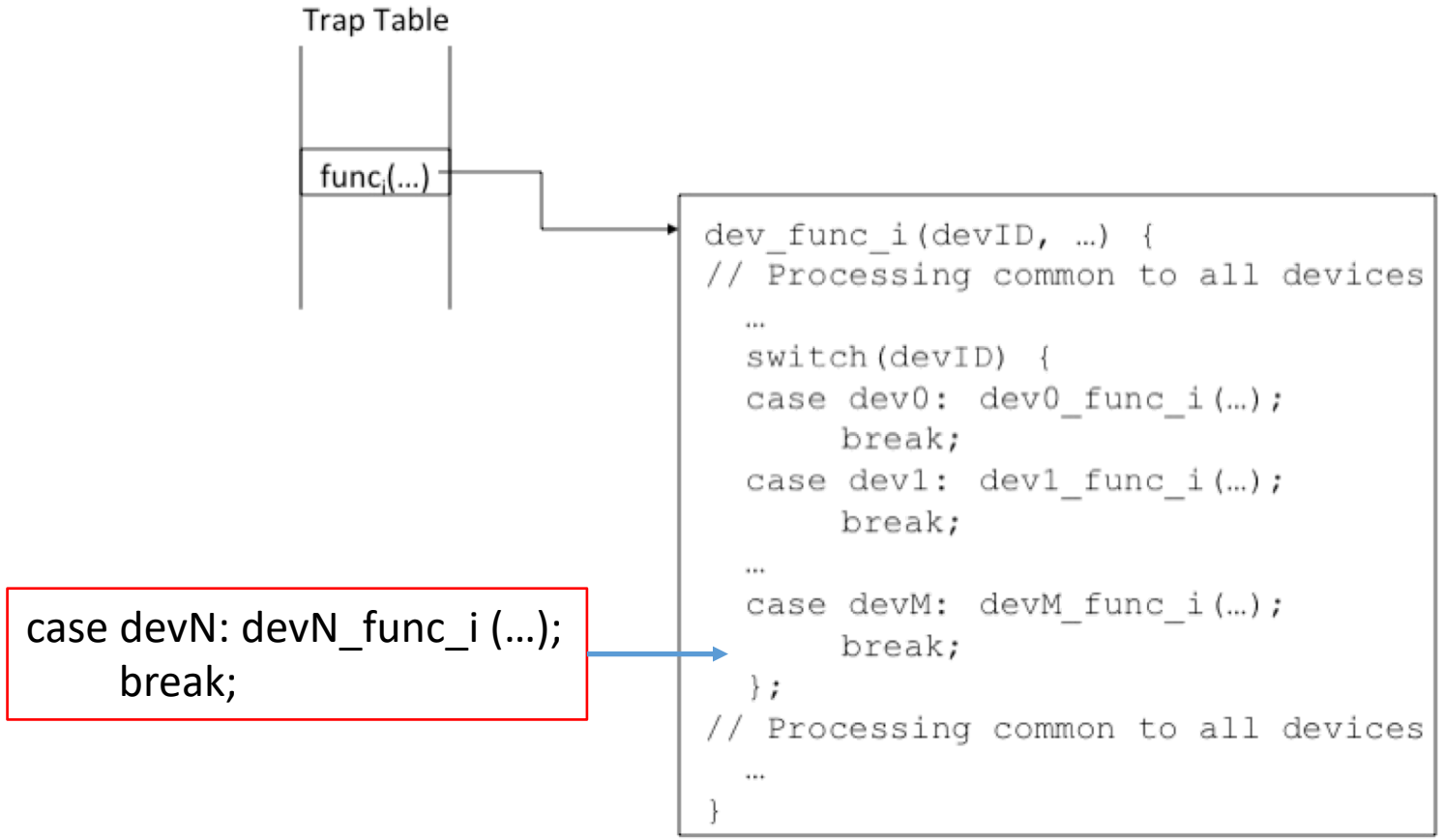
hda, sda, tty, null, zero, lp, mem, console

Major and Minor numbers

```
# ls -l /dev/hda[1-3]
brw-rw----  1 root  disk  3, 1 Jul  5  2000 /dev/hda1
brw-rw----  1 root  disk  3, 2 Jul  5  2000 /dev/hda2
brw-rw----  1 root  disk  3, 3 Jul  5  2000 /dev/hda3
```

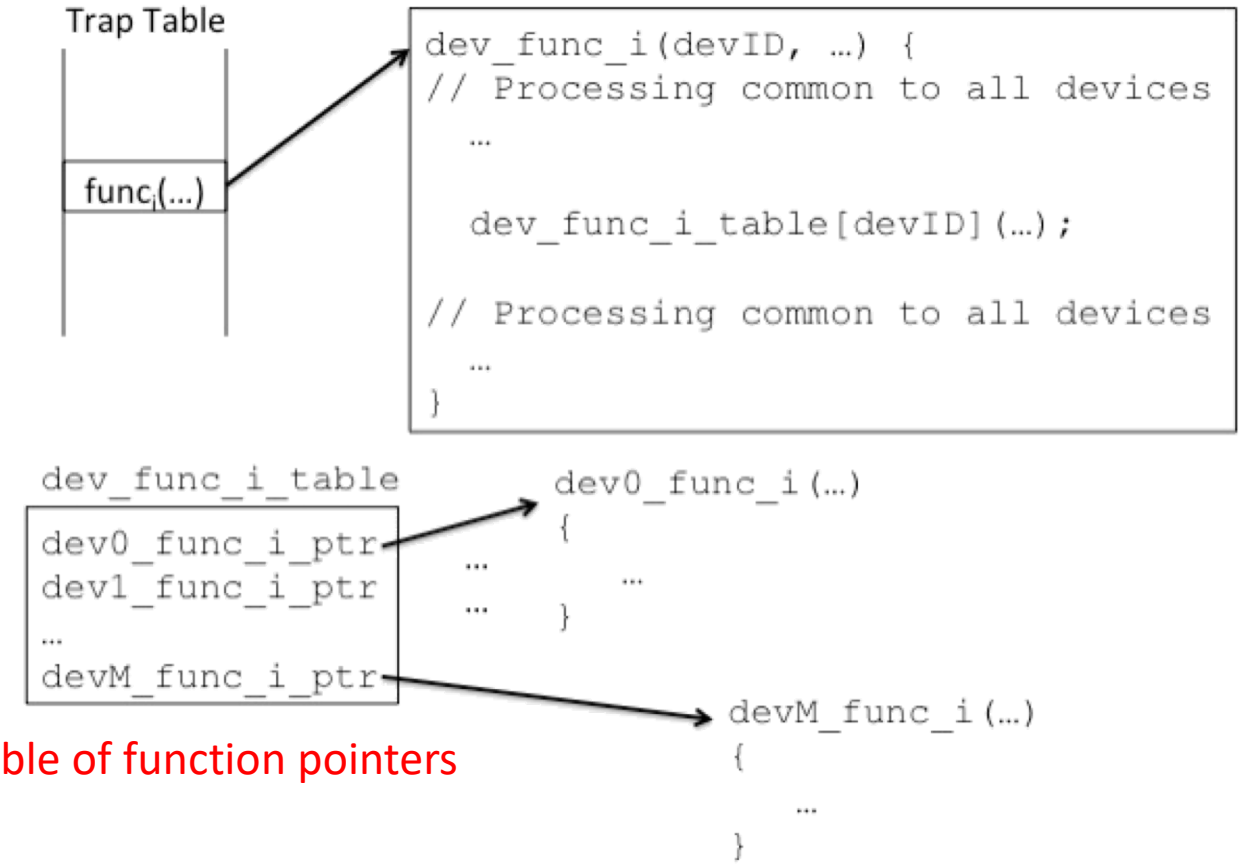
Add a New Device Driver

- Traditional device driver code in the kernel

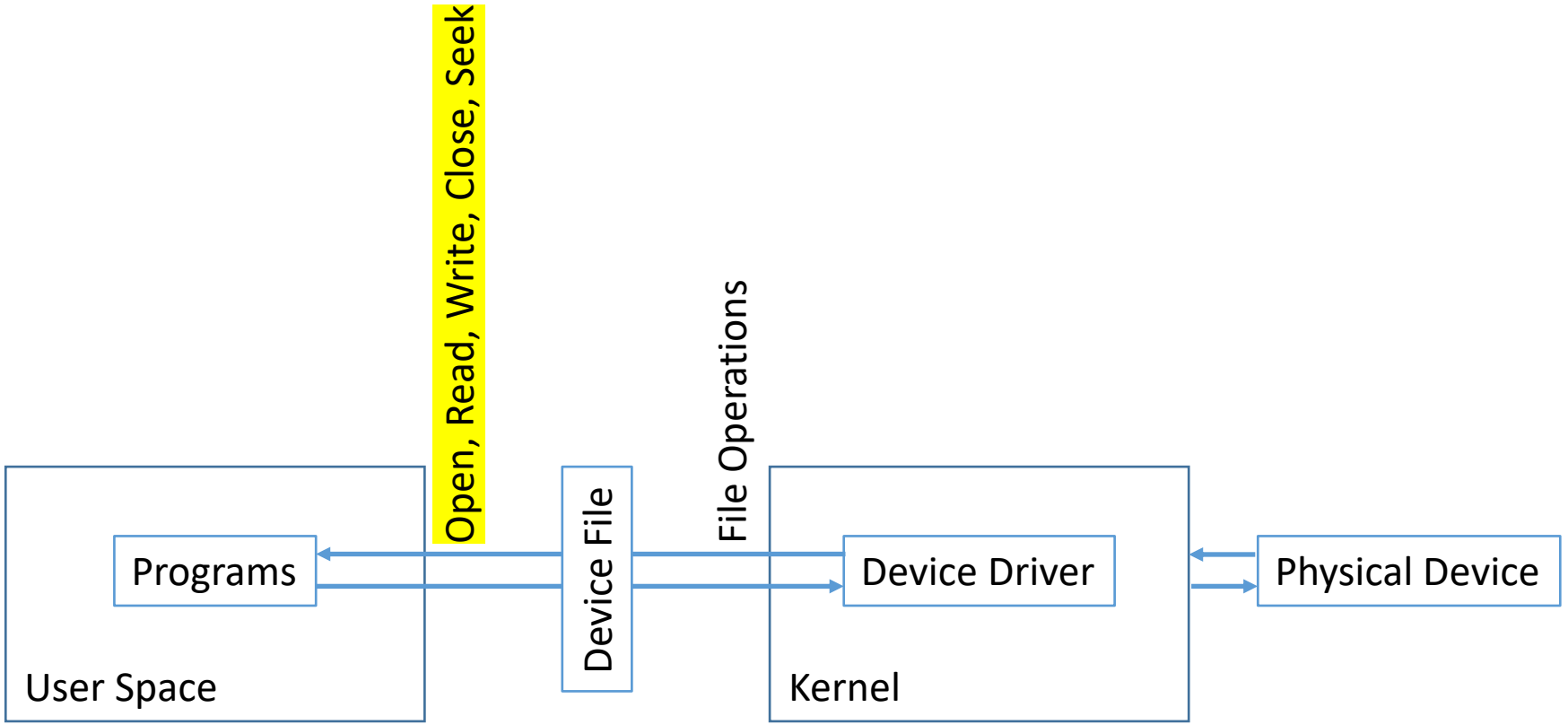


Add a New Device Driver

- For the kernel that allows LKM



PA2 – Character Device File



Week 3 – Checklist

- ❑ Write an LKM example
- ❑ Discuss device driver
- ❑ Register for a time slot for PA1 interview