Drivers

CS169 - Spring 2016
Weenix Layout

User Land!

Virtual Memory

System Calls
- open, read, write, lseek, mount, getcwd
- fork, exec, wait
- yield
- mmap, brk

VFS

S5FS

TTY

Disk Driver

Terminal Driver

Other Drivers

Process Management

Scheduler

TLB

Page Tables

Page Frames

Other Drivers

TLB

Page Tables

Page Frames
Drivers - Details

You already have threads, mutexes, and queues (condition variables)
Now you need to let programs interact with the outside world.
Read/Write to disk, Type on screen, etc.

Drivers let you do this by mediating between the OS and hardware
Drivers - Getting Started

Set ‘DRIVERS = 1’ in Config.mk
‘make clean && make’
Now you are doing drivers.
All code is in ‘kernel/drivers’
You need to implement 18 functions.
Drivers - Overview

- **Block Devices**
  - Disk Driver
    - ATA Device

- **Character Devices**
  - TTY Driver
    - Line Discipline
    - TTY
  - Memory Devices
    - /dev/zero, /dev/null, etc.
Drivers - Block Devices

These can either be real (hardware) devices or virtual (memory) devices.

You need to implement 1 block device:

- An ATA Device (hard drive)
- Reads in terms of disk blocks

Driver needs to:

- Translate reads into terms of pages
- Use queues and interrupts to allow other code to run during IO
Drivers - Block Devices

Block devices all have 2 operations

read

reads a given number of blocks, starting at a given offset, into a buffer. Returns the number of bytes read.

write

Writes a given number of blocks into the device, starting at a particular offset, from a buffer. Returns the number of bytes written.
Drivers - ATA Devices

Controlled using DMA registers
Need to provide information like sectors, blocks, etc.
DMA sends a hardware interrupt when an operation finishes
Read and write are both implemented in ‘ata_do_operation’ since they are very similar.
Drivers - ATA Devices

These are very finicky, you will likely destroy the disk several times. Use ‘./weenix -n’ to start weenix with a new disk. Use ‘./fsmaker disk0.img -i’ to look at the disk. Use ‘block’ command to see contents of blocks.
Drivers - Character Devices

These are things like ‘/dev/null’, ‘/dev/zero’, ‘/dev/tty’ etc.
These will not be actual files until VFS.

We provide a keyboard and terminal driver.
They will read keyboard input and can write strings to the screen.
Drivers - Character Devices

Similar to block devices except used in terms of characters. Generally read one line at a time.
Drivers - TTY

The VT subsystem (keyboard and terminal drivers) will call your code to handle input. You will write a line discipline to do this. Simple circular buffer to provide line-oriented read(2). Handles deletion of characters.
Drivers - Line Discipline

Keyboard interrupts call TTY which calls line discipline.
When a newline is passed, the buffer is “cooked” up to a certain point.
Anything up to that point cannot be changed.
The actual tty methods mostly just pass along data to the line discipline.
Drivers - KShell

Once you have tty’s working you will be able to use the ‘kshell’. This is a ‘sh’ like environment where you can run code from the command line. You will need to modify init to get it running.

Look at ‘kernel/test/kshell.c’ to figure out how
Drivers - Other Stuff

Be sure to continue testing your code. You might want to add your procs tests to the kshell. Ask your mentor TAs if you have problems. Start Early.