





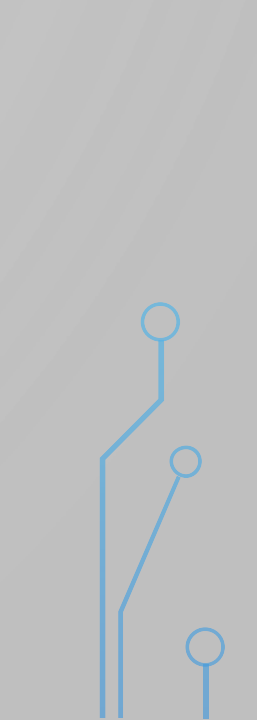
DRIVERS (OVERVIEW)

CPE / CSC 159: OPERATING SYSTEM PRAGMATICS

GREG CRIST (CRIST@CSUS.EDU)



DRIVERS

- What are drivers?
 - Software that controls, manages, or otherwise operates or interacts with a hardware device
 - Typically provides an abstraction layer between the underlying hardware and other software
 - Why are they needed?
 - Simplifies hardware interactions via abstraction
 - Facilitates hardware interactions via defined interfaces: APIs
- 
- 
- 

DRIVERS - EXAMPLES

- Examples of drivers in the real world
 - PC Video drivers
 - Different vendors have different hardware architectures, capabilities, and performance characteristics
 - Networking
 - Different physical means of establishing a network, but a common interface for programs to use (e.g. WiFi vs Ethernet)
 - Think of your OSI Networking Model – underlying hardware could be abstracted for higher layers
 - Serial communication
 - At the software level may simply be data in or data out, but different serial devices may operate differently (e.g. transmission speed, error correction, etc.)
- More examples:
 - Printers, webcams, Bluetooth modules, drawing tablets, touch screens, keyboards, mice, midi controllers, etc, etc, etc.

DRIVERS – RANGE OF COMPLEXITY

- Drivers may be simple or complex in their implementation/design
 - Typically: a high correlation with the complexity of the underlying hardware
- Simple:
 - basic hardware input/output requiring “just enough” software to function
 - May imply that hardware itself is simple (or)
 - Hardware may be complex, but interface to the system is simple
- Complex
 - significant interpretation/processing of the underlying hardware input/output
 - May imply that the hardware itself is complex (or)
 - Data requires additional processing that cannot be done (or not desired) to be performed in hardware

DRIVERS – OPERATING CONTEXT

- Drivers can (but not always) operate in different contexts
 - Kernel context (runs completely in the kernel)
 - User context (runs completely in user-space)
 - Mixed context (part kernel, part user)
- Advantages and disadvantages to both
 - Tradeoffs between performance, stability, complexity

DRIVERS – BUS AND DEVICE INTERACTIONS

- With complex computer systems composed of hardware busses and hardware devices, separate drivers may be needed for both
 - Examples: USB (Universal Serial Bus), PCIe (Peripheral Component Interconnect Express)
- Operating system kernels typically have built-in drivers to control and manage the hardware bus
- Individual devices on a bus may have separate drivers that leverage the bus operations, but are specific to the devices they are controlling
 - Example: Webcam over USB vs Video Game Controller over USB
 - Example: Video Output over PCIe vs Storage Device over PCIe

DRIVERS – DESIGN APPROACHES

- Drivers may be simple or complex
- Simple:
 - basic hardware input/output requiring “just enough” software to function
 - May imply that hardware itself is simple (or)
 - Hardware may be complex, but interface to the system is simple
- Complex
 - significant interpretation/processing of the underlying hardware input/output
 - May imply that the hardware itself is complex (or)
 - Data requires additional processing that cannot be done (or not desired) to be performed in hardware

DRIVERS – SINGLE VS MULTI-PART DESIGN

- Drivers may use a single or multi-part design
- Single-part design
 - All interactions occur in a single piece of code
- Multi-part design
 - Interactions with hardware spread across multiple pieces of code
- Considerations
 - Complex hardware interactions
 - CPU vs I/O
 - Abstraction
 - Modularization

DRIVERS – TOP HALF

- Top Half / Upper Half
 - Should be lightweight and responsive
 - Minimal hardware I/O
 - Quick to complete operations
- Generally focused on
 - Signaling (think: IPC between different software components)
 - Interrupt handling (responding to external stimulus)
- Can be triggered via interrupts or system calls from user processes
- Typically coordinates / stimulates the bottom half

DRIVERS – BOTTOM HALF

- Bottom Half / Lower Half
 - May contain more hardware I/O
 - May contain more significant processing / interpretation of data via software instructions
- Hardware I/O typically happens here
 - Input / Output operations are slow (in comparison to CPU execution) and shouldn't “slow down” the kernel operation

DRIVERS – SOFTWARE INTERFACES

- Drivers can be interacted with via interfaces defined by the operating system
 - Abstracts the hardware interfaces to the software that needs to interact with the underlying hardware
- Can leverage existing operating system components to facilitate interfacing with hardware
 - Examples: interrupt handling, system call implementations, driver tasks