I/O Software Layers

Daniel R. Cender

Grand Canyon University: CST-221-O500

Feb 8, 2020

This assignment will explore the details of all four software layers in an operating system’s I/O (input/output) operations.

**Interrupt Handlers**

**Description**

Interrupt handlers are callback “functions” that run once a device has performed an operation.

**Purpose**

The use of these handlers allows the operating system to delegate processing of system calls to sub-routines in the protected kernel space, but without having to constantly poll the device for new data input.

**Example**

Once a device has performed an action, the controller puts a number on the address lines specifying the device needing attention and sends a signal to interrupt the CPU (Tanenbaum & Bos, 2013). Once the CPU finds the corresponding interrupt-service handler, it sends a signal to the controller signifying that device’s port has found a handler for the interrupt (Tanenbaum & Bos, 2013). Once the handler has performed its process entirely, it will signal the device controller that it’s finished. This signal could be a literal signal, performing *up* on a semaphore, or signaling a monitor construct.

**Device Drivers**

**Description**

A device driver is a bit of device-specific code designed to control its operations.

**Purpose**

Having device drivers abstracts the need for the OS kernel to understand how to interface with every device on the market. Manufacturers just need to write a handful of drivers to support a selection of operating systems, and their products can be used in most setups after that.

**Example**

In the example given by Tenanbaum and Bos, the device driver issues commands to the device controller’s registers, where they are picked up by the CPU and handed off to an interrupt handler (2015).

**Device Independent I/O Software**

**Description**

Device-independent software performs I/O operations that are common to all drivers and to “provide a uniform interface to the user-level software” (Tenanbaum & Bos, 2015).

**Purpose**

The purpose of writing this layer of device-independent software is so the drivers do not have to implement all their operations from scratch. This simplifies the work of people responsible for writing drivers.

**Example**

The drivers must supply a set of functions the operating system requires of that particular class of device. In an example disk driver, the writers would supply functions for reading, writing, controlling power, or formatting.

**User Level I/O Routines**

**Description**

These I/O routines are linked libraries of processes called by programs at execution time. Some of them do as little as rearranging parameters, and many do real I/O operations using system calls.

**Purpose**

The user level I/O routines exist to provide an interface to the OS’s device-independent system calls and processes (Operating System - I/O Softwares, n.d.).

**Example**

A great example of a user level I/O process is the *printf* function available from the *stdio* (standard input/output) C library (Tanenbaum & Bos, 2015).

References

Gottlieb, A. (n.d.). Principles of I/O Software. Retrieved February 7, 2020, from https://cs.nyu.edu/courses/spring03/V22.0202-002/lecture-13.html

Operating System - I/O Softwares. (n.d.). Retrieved February 7, 2020, from https://www.tutorialspoint.com/operating\_system/os\_io\_software.htm

Tanenbaum, A.S. & Bos, H. (2015).*Modern Operating Systems.*Chapter 5.