Theory of Device Operation

Daniel R. Cender

Grand Canyon University: CST-221-O500

Feb 9, 2020

**Diagram**

**A close up of text on a white background

Description automatically generated**

*Figure 1.* Block diagram of basic I/O processing parts in a computer.

**Mouse I/O Process**

The mouse I/O process runs using the programmed I/O model, where the cost of processing an interrupt outweighs the speed and volume of data sent by the device. When the user moves their mouse, it sends a telemetry information (x and y coordinates) to the device driver. The drivers for basic I/O devices like keyboards and mice are fairly standard across most systems, given how similarly use cases are from one device to another. The driver alerts the places signals on its corresponding port of its device controller and uploads data to its corresponding data registers. If the controller is free, it will process the real I/O operation, which in this case would involve system calls and will potentially trickle down into performing an action (say, clicking a button) in a user-level program. Although a controller is required to perform system calls, drivers may interact with much of the kernel by itself (Tanenbaum & Bos, 2015). In the case of gaming mice, which sport programmable buttons, a special manufacturer driver will likely be required to interface with the controller and the device registers to perform otherwise impossible actions for a “bare-bones” mouse. After the I/O finishes, the controller runs the last segment of the driver, which will return control to the user-level code processes (Gottlieb, n.d.).

**Keyboard I/O Process**

The keyboard I/O process runs using programmed I/O setup, like the mouse. The frequency and volume of data generated by a typing human being make using a DMA overkill (Decker, n.d.). Research and reading the class material yielded a process fairly identical to the mouse I/O. When a user presses or releases keys on the keyboard, data (which key and whether it was pressed/depressed) is sent to the driver, which formats the input for interfacing with the device controller. That controller will likely just perform the true I/O operations then and there. The “bottom-part” of the driver is then alerted to the successful completion of the I/O operation and control is returned to processes in the user space (Gottlieb, n.d.).

References

Decker, D. (n.d.). Is DMA Appropriate for a Keyboard? Retrieved February 9, 2020, from https://stackoverflow.com/questions/59166606/is-dma-appropriate-for-a-keyboard

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

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