**Fredrick Onduso Ondieki**

**CST-221 Operating Systems Concepts**

**CST-221 File I/O Assignment**

**John Zupan**

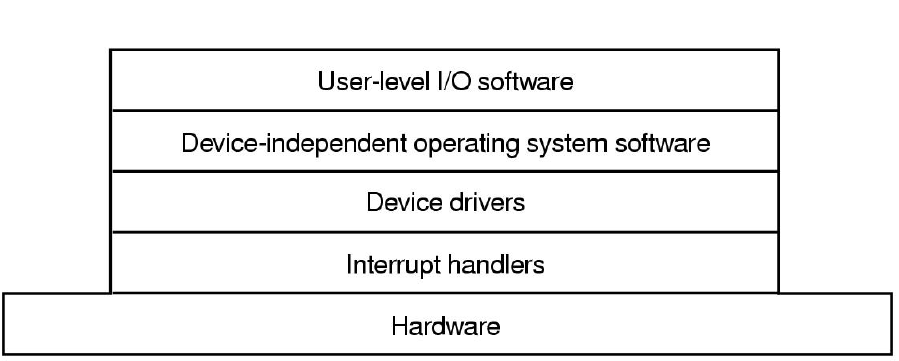
**April 2, 2019**

**CST\_221 File I/O**

In this project assignment, we are required to do research on I/O devices and how they handle hardware interruptions within the Linux operating system. To complete the assignment successfully, we are required to follow the activities directions outlined on the project assignment documentation.

**Research the four I/O Software Layers found in a typical operating system. For each Layer provide a description of the layer, the purpose of the layer, and a concrete example of how this layer is supported in the Linux operating system.**

The diagram below shows the different system independent I/O software layers. Most developers and computer scientists use. Also, each system serves a specific purpose.



**I/O Software Layers**

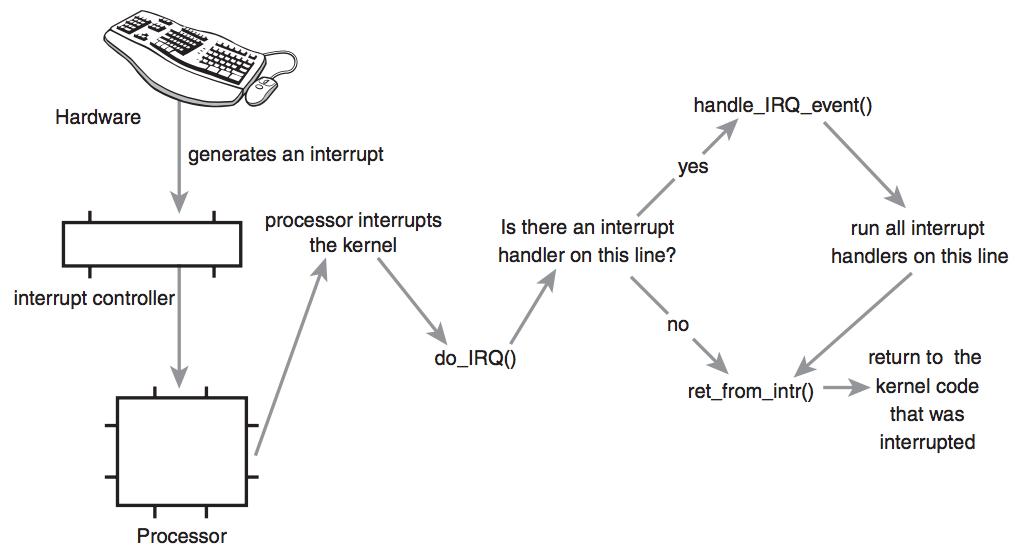
**Interrupt Handlers**

**Description**

Interrupt handlers also known as interrupt service routine (ISR) are simply hardware signals from a device to the Central Processing Unit (CPU). Interrupt handler is also a callback function used in the OS (operating system) most specifically whose task is to activate interruptions. (Chapter 8 Interrupt handlers, 2010).

**Purpose**

The purpose of having Interrupt handlers is to interrupt the normal flow and request services. Interrupt handlers are specifically hidden so that the operating system must deal with them and their functions as little as possible. The following image indicates the steps taken after an interrupt.



**Example**

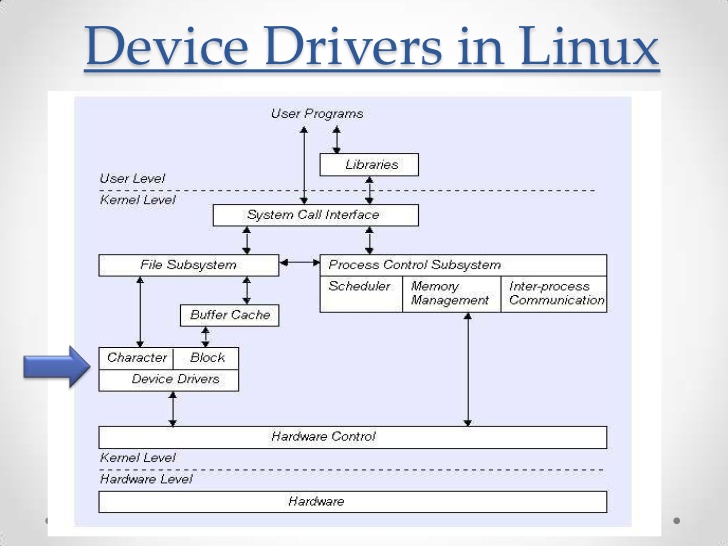
A Keyboard adapter may interrupt and send a signal to the central processing unit that a character has been typed on to the console by the user. Also, a Network adapter may also interrupt to signal the CPU that a chunk of data has been received or in another instance that the data that was being transmitted has completed its process. To add, according to Hobson, 2018 the system interrupt handler is the very innermost heart of a modern O/S. This is because in a virtual memory environment, it must \*always\* be loaded into memory and it must never move.

**Device Drivers**

**Description**

Device drivers are software programs designed to control a hardware device that is installed or attached to the computer. Specifically, device drivers are used in controlling I/O devices on the computer system. Typically, device drivers’ manufacturers write these drivers and deliver the drivers alongside with the devices. To add, device drivers perform the following functions;

* Accepts requests from devices independent software above to it
* They also interact with the device controllers to take and give I/O as well as perform required error handling functions and lastly,
* They make sure that all requests are executed successfully.



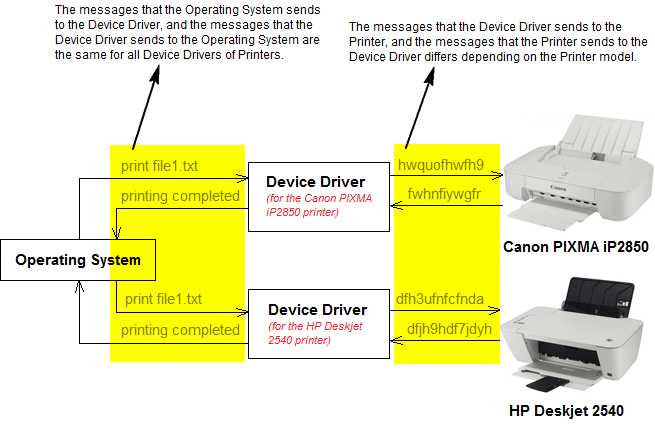
**Purpose**

Device drivers ensure that a swift functioning hardware parts to carry out the purpose for which it is created to do. Device drivers also allow devices installed on the computer to be used across different operating systems. In addition, device drivers control I/O devices by issuing commands. The commands issued are determined at the device level.

**Example**

Linux operating system allows easy coupling of device drivers to be built as well as being installed onto their systems. Suppose that we have a request to print a document file to the printer. If the device driver is idle at the request time; the driver will start to work on the request immediately.

The following image expounds further on the device drivers.



**Device-Independent I/O Software**

**Description**

Device independence refers to act or the process of making the software program or application be used in a wide range of devices regardless of where the software will be used. It does not matter the hardware used to host or run the software application. However, some I/O software manufactured are device-target specific, most of it is device independent.

**Purpose**

The device-independent I/O software achieves the uniformity on computer system interfaces. According to Nalishuwa, 2016, the device-independent system aligns the user-level software. Some functions that are done in the device-independent software include:

* Reporting errors
* Buffering
* Uniform interfacing of system devices
* Allocating and releasing the allocated devices and lastly
* Providing device independent block sizes.
* Device protection

**Example**

Everything on the Linux file system is either a file or a directory. Every device that the system uses is kept in a folder. Most importantly, all the device information is stored on the very specific folder at the time of installation. The device files stored on the folders enables the user to access the hardware associated with it. For example, if you run the script ***cat /dev/dsp > recording001,*** and record yourself to the computer via the microphone, running again the script ***cat recording001 > /dev/dsp*** the system will play back what you recorded.

**User-Space I/O software**

**Description**

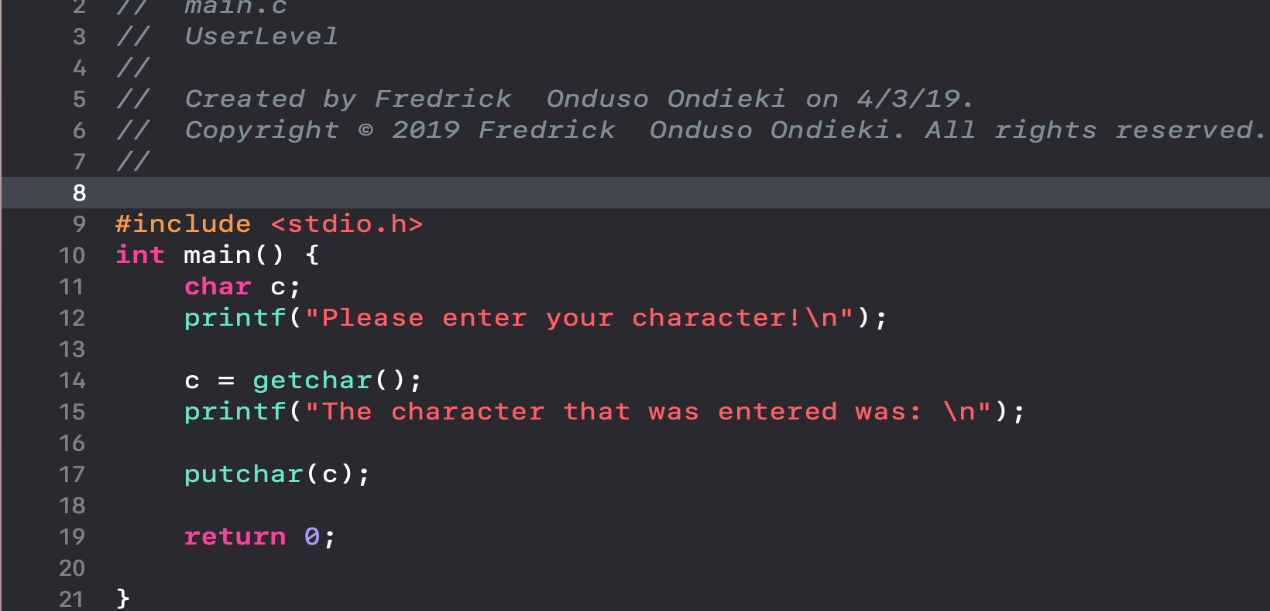
User-Space I/O software is the part that is considered as the top layer of I/O software level as described on the report above. Most of the programs on this approach are run outside the operating and the kernel systems. In addition, it consists of libraries that provide simplified interfaces to access the functionality of the kernel or to interact with the device drivers. In a general form, this layer consists of programs that are used to interpret the user input commands inclusive of all the libraries associated with commands issued or used by the user.

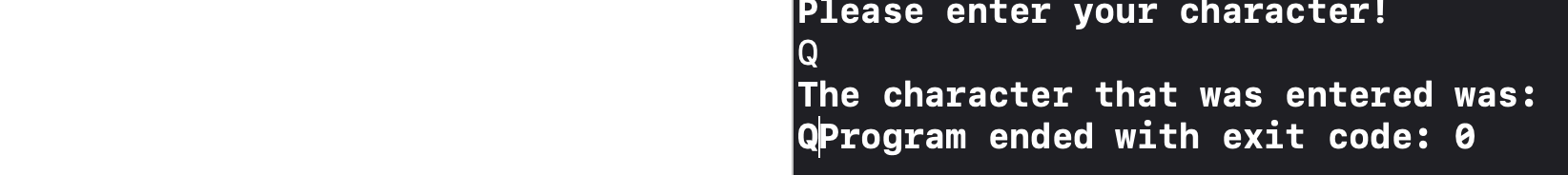
**Purpose**

The User- Space I/O Software actively and accurately route the user commands to the default place that is chosen by running program. To sum up, this is the layer that formats the commands used to make I/O calls to the subsystems.

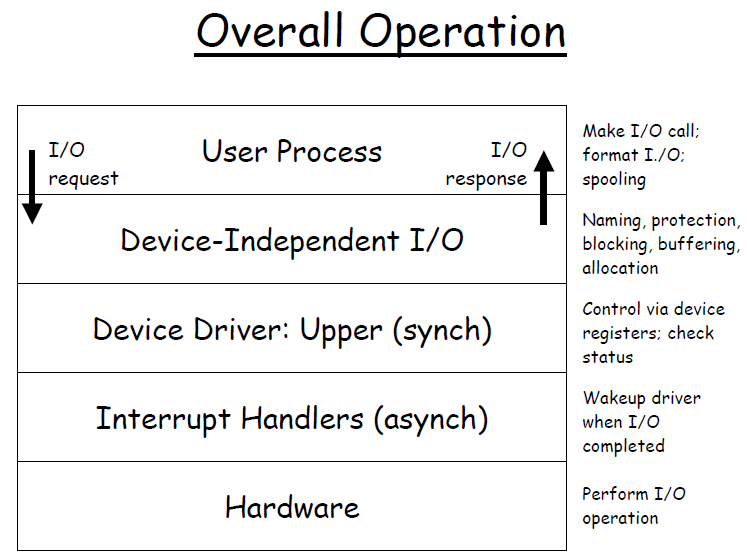
**Example**

Some User-Space I/O library include ***stdio setchar(),*** ***getchar(), printf(), and scanf()***  all which are used on the C language (Operating System - I/O Software, n.d.). For example, the program below illustrates the usage of the ***getchar()*** library. The user enters a character to the program and then it returns the character that is entered.





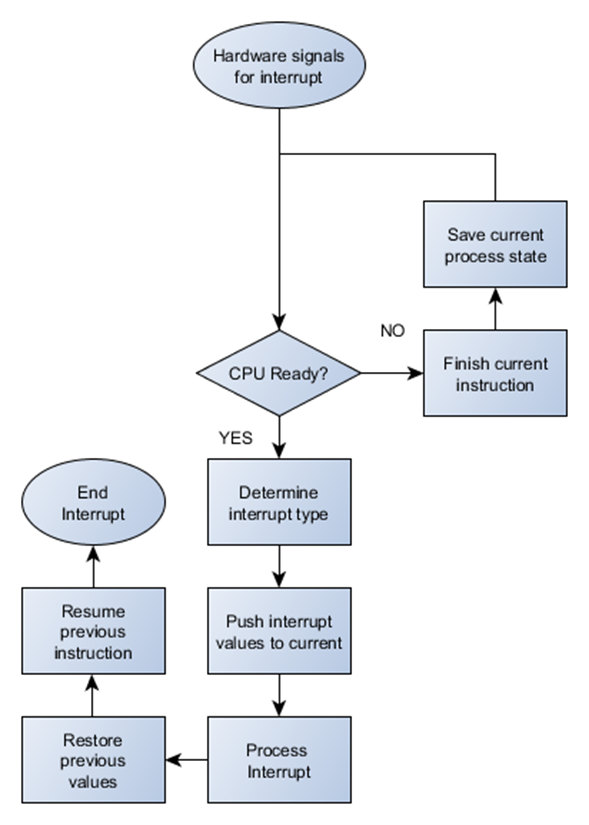
The diagram below gives the summary of I/O software layers.



**Draw a flow chart for the series of steps that are generally required to handle and process a hardware interrupt.**

Many computer systems use hardware interruptions to pass information and signals that need immediate attention. For example, a hard disk may pass information to the operating system that it has finished reading or writing data. Also, a network device such as an internet modem signaling that it is buffering the network packet data has completed successfully.

The hardware interrupts that occur on the operating system are assigned interrupt numbers. The interrupt number assigned to the devices that gave the signal help the operating system to monitor devices that created the interrupt and when it occurred. Moving forward, when an interrupt signal is received, current activities running gets stopped, and then the interrupt handler gets executed.



**Summary: Hardware interruption**

The hardware signals for interruption. The CPU is checked to confirm if it is ready to accept the signal. If the CPU is not ready, the current instruction that is running continues to execute until the process is finished and then the process is saved. Else, the interrupt type is determined and its value or referenced value made current for the process to interrupt. The previous values are then restored while the CPU resumes the previous instruction as the interruption ends.

**Research how the keyboard and mouse input I/O Devices function. For each device write a theory of operation detailing how each input device works with its various components, including its physical Hardware Interface, Device Controller, I/O registers, Interrupts, Device Drivers, and Operating System to support a fully functioning I/O system.**

**Keyboard functions**

Theory of operation

Keyboards primarily are used to enter characters of text. Other uses include performing keystrokes to open files or access various sections of the operating system. The keystrokes performed by the user can be used to play video games and manipulate the players on the game. One important aspect of the keyboard is that it can be used to pass commands to the computer and achieve more advanced things where the mouse would not be enough. For example, within a built-in command prompt, one can tell the computer to execute certain actions and even gain valuable system information. Finally, to act as a mode of the panic button. You can set some hotkeys for toggling processes, or to execute a series of actions, in case things happen to go haywire.

Those keys that we see, and press are the physical hardware part on the keyboard. Beneath the buttons, the keyboard has a matrix which is the user I/O. The key mapping on the keyboard gets the strokes performed by the user, then the input is sent to the device port. The software function passes the stroke signal and later sent to the device input which is formatted by the computer. Afterwards, the input is then applied to the running process. The common type of interrupt function that the keyboard has is ***ctrl + alt + delete.*** This function causes current applications running to pause. The process locks the system access and usage temporarily. To unlock it, the user will therefore need to press the command (***ctrl + alt + delete***) again and enter the user credentials.

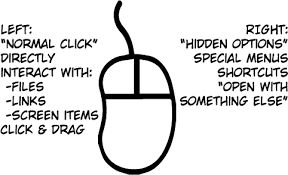
**Mouse functions**

Theory of Operation

As we have discussed above, while the keyboard is set be used for typing (Input device), the mouse is yet used as a pointing device used alongside with other examples which are joysticks, camera mouse, light pen and many others.

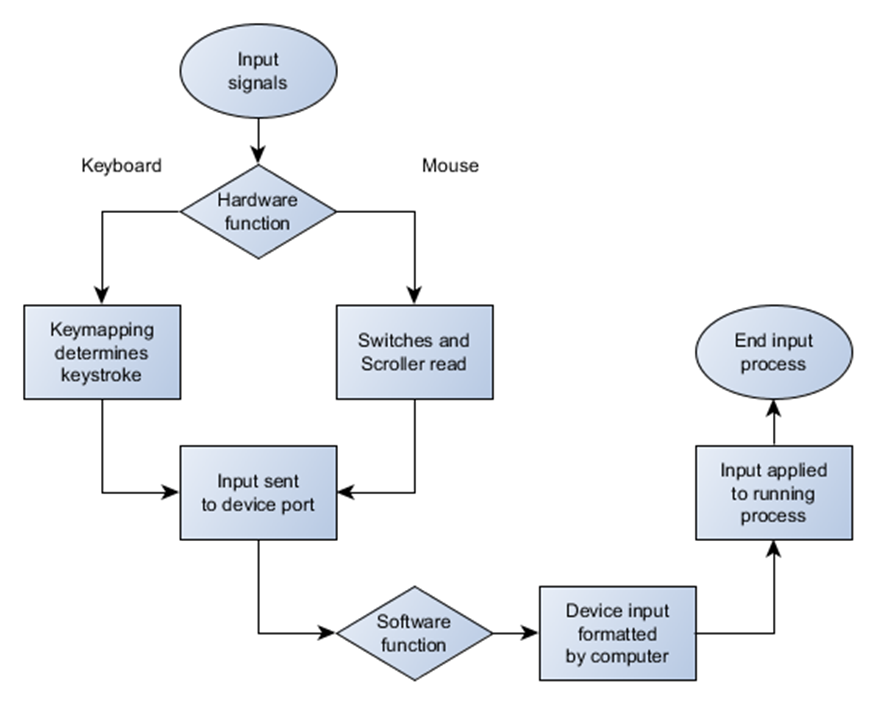
The operating system receives the signal from the which can either be from Bluetooth, Wi-Fi or cable connections. When a user is moving the mouse across the desk or table, the pattern is detected, and the system figures out the users’ direction of movement.

When we look at a typical mouse, it has parts that are part of the user input layer. These parts perform various duties. For example, the right and left buttons are used to select, double-clicking and providing a list of commands when right-clicked respectively.



When a user performs clicks on an item, the switch and the controller read the signal from the user as an input transmitting it to the software of the mouse. The computer formats the device input and then applied to the running application, then the input process is terminated.

Mouse function does not have interruption function. This is because if there is no current operation that is running, then even if the user performs numerous clicks, nothing will happen.

****

References

Chapter 8 Interrupt Handlers. (2010). Retrieved April 2, 2019, from https://docs.oracle.com/cd/E19253-01/816-4854/interrupt-15678/index.html

Chapter 3. Hardware interrupts. (n.d.). Retrieved April 6, 2019, from https://access.redhat.com/documentation/en-US/Red\_Hat\_Enterprise\_MRG/1.3/html/Realtime\_Reference\_Guide/chap-Realtime\_Reference\_Guide-Hardware\_interrupts.html

Gottlieb, A. (2003, March 31). Principles of I/O Software. Lecture 13. Retrieved April 3, 2019, from https://cs.nyu.edu/courses/spring03/V22.0202-002/lecture-13.html

Hobson, M. (2018, February 01). Why does an interrupt occur on an O.S.? Retrieved April 2, 2019, from https://www.quora.com/Why-does-an-interrupt-occur-on-an-O-S

Kumar, A. (2018, December 03). What is a Device Driver? What is its purpose? Retrieved April 2, 2019, from https://www.thewindowsclub.com/what-is-device-driver

Michael, K. (2010). The Linux programming interface: a Linux and Unix system programming handbook. Retrieved from https://ebookcentral-proquest-com.lopes.idm.oclc.org

Nalishuwa, M. (2016, August 25). Input - Output Software Layers. Retrieved April 3, 2019, from https://prezi.com/dm12v6l2lktx/input-output-software-layers/

Operating System - I/O Software. (n.d.). Retrieved April 3, 2019, from https://www.tutorialspoint.com/operating\_system/os\_io\_software.htm

Ray, J. (1999). Teach Yourself Linux in 10 Minutes. Indianapolis, IN: Pearson Education, Inc. Retrieved from https://lopes.idm.oclc.org/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=12198&site=ehost-live&scope=site

Tanenbaum, A. S. (2006). Principles of IO Software. Retrieved April 3, 2019, from https://flylib.com/books/en/3.275.1.28/1/

Tyson, J. Wilson, T. (2018). How computer keyboards work. Retrieved from https://computer.howstuffworks.com/keyboard.htm

What are the three most important uses of the keyboard? (2018, February 03). Retrieved April 7, 2019, from https://www.quora.com/What-are-the-three-most-important-uses-of-the-keyboard