Writing 2

Writing 2 CS444 Spring2018

Brian Huang

1 Introduction

General input and output is necessary for all operating systems to function. Whether the operating system is Windows, Freebsd or linux, they all need some sort of input and output to manipulate their data in some form. They manipulate this data to execute process and display information to the user. Data is usually managed through some kind of file, an example of this is in linux, where directories can be edited by any sort of text editor like vi or nano. The files can be moved, created and edited through these editors. To start talking about the more complicated topics in more detail we need to understand the definition of a block and character device. Block devices are devices that are able to randomly access fixed size chunks of data. They need to be able to be accessed in any order, an example of this is any sort of storage device such as an USB, hard drive and solid state drive. A character device are devices that are accessed in order, this is also known as data streams. These devices include things like microphones, keyboards and mice. Now that we have established these two definitions it will be easier to discuss the details of things like functionality, cryptography and scheduling.

2 Input/Output Windows and Linux

In terms of input and output Windows and Linux are drastically different. Their differences range from their style of management systems to the data structures they use to manage requests made by devices. Windows uses Input/Output request packets, also known as IRP's to manage requests. These packets are representations of any request and contain essential header data that will affect the way the request the device made is handled. The system that handles these packets is called the WIndows I/O manager. The manager is responsible for transferring the packets to the various drivers, then to its correct location. The manager does this by using a layered stack, where each driver is cataloged a packet, then it is sent to the bottom of the stack to either one or many different drivers. When the request is completed the manager notifies the application that it has initiated the packet. The request is completed when either one or all of the following conditions are met: If the request is cancelled, if it contains invalid parameters or it no longer needs to be passed down the driver stack. If one of these conditions are met the request is freed by the I/O manager or it is sent to the drive it was allocated to.

The main difference here is that instead of a manager, Linux uses a stick to manage input and outputs which is also known as block input/output. The request made with linux are similar uses a stack and file pointers to manage its requests.

3 Input/Output FreeBSD and Linux

The input output structure and management techniques the Freebsd uses are somewhat similar to that of linux. This is mainly because of how the two operating systems are connected in their lineage. Freebsd's core component is similar to Linux's where it uses files for all process and system storage. This is done by using file descriptors which are not managed by any sort of manager like windows uses. The file descriptors are instead accessed and connected by pipes and sockets. The system of using files for everything is usually hidden from the user in Linux and Freebsd and kept in the kernel. Accessing the descriptors and their data is done by using the same functions as files. The similarities of Freebsd and Linux continue as they also use the same data structure of block input and output. The only difference here is a slight difference in some of the variable the data structure contains.

4 Cryptography in Windows, FreeBSD and Linux

The developers of Windows, Freebsd and Linux are able to use any encryption techniques they want. WIndows uses Advanced Encryption Standard(AES), Rivest–Shamir–Adleman(RSA), and Digital Signature Standard(DSS) as well as a couple others. Freebsd and Linux use some of the same encryption algorithms as windows, and their implementations are mostly identical. The main difference in the implementations are naming, and data structures that are used.

5 Conclusion

Windows, Freebsd and Linux all abstract their file management system under a layer of I/O management. For Windows this is done through the Windows I/O manager and for Linux and Freebsd this is done through file descriptors. These operating systems all also uses the same cryptography techniques when encrypting their devices, where the main difference between the three is the implementation of whatever algorithm it is using.

References

[1] A. L. Mark Russinovich, David A. Solomon, "Windows internals part 1," Microsoft Press, 2012, (Accessed on 5/9/2018).

[2] R. Love, "Linux kernel development," Pearson Education, 2010, (Accessed on 5/9/2018).

[3] R. N. W. Marshall Kirk Mckusick, George V. Neville-Neil, "The design and implementation of the freebsd operating system," pearson Education, 2015, (Accessed on 5/9/2018).