CS 444 - Operating Systems II Writing Assignment 2

May 9, 2018

Prepared for

Kevin McGrath

Prepared by

Nick Wong

Abstract

I/O, or input/output is an important aspect of how a computer functions. This document will examine the differences and similarities within the implementation of I/O within the operating systems FreeBSD, Linux, and Windows.

CONTENTS

1 INTRODUCTION

Within the realm of the computer, the term I/O, or input/output, is the link which allows for basic interaction between processes in the computer, and between the computer and the user as well. I/O describes the process which happens between data transferring from one point to another. Without I/O, the output of all processes would be rendered intangible, and thus any computation within a computer would be essentially pointless.

2 I/O IN WINDOWS

The main design goals for I/O within Windows is to provide an abstraction of devices, both hardware (physical) and software (virtual or logical), to applications with[15] the following features in mind:

- Security
- Scalable applications
- Compatibility for other machines with similar architectures
- Efficiency with system resources and power
- Plug n Play to allow system to automatically search and install drivers
- Windows Management Instrumentation (WMI) support and diagnosability so that drivers /item can be managed and monitored through WMI applications and scripts. [15]

2.1 Similarities to Linux

Windows and Linux both have a construct they use to allow for device interfacing. Windows uses something called the Windows Driver Model (WDM), which allows developers to write drivers which can be scalable to different machines of the same operating system. Whereas Linux uses character and block devices as an interface for I/O.

2.2 Why these similarities and differences exist

The very first difference between these three operating systems is the way they create their process. FreeBSD/Linux creates their processes by providing two separate calls in a procedure called forking which are known as tasks. Whereas Windows creates a CreateProcess function which runs independently of the creating process. In a nutshell, FreeBSD/Linux takes an extra step in creating their processes, whereas Windows keeps its process creating function connected. A few similarities to these operating systems is their process address spaces and user mode and kernel mode.

Windows and Linux offer dynamically loadable modules. In Windows, a dynamic-link library (DLL) is used. It is a module that contains functions and data that can be used by another module. [7] In Linux, the dynamically loaded libraries are built as standard object files or standard shared libraries[3].

Both Windows and Linux use asynchronous I/O and I/O calls. Asynchronous I/O is a form of input/output processing that permits other processing to continue before the transmission has finished. Linux has various system call methods such as io_getevents(), io_submit(), io_cancel(), and io_setup()[1]. In Windows, these system calls are very similar; OPEN, CLOSE, READ, WRITE, and IOCTL.

One more similarity between Linux and Windows is their idea of power saving. Both operating systems support something called disk hibernation. Disk hibernation allows for the suspending of current work inside an OS, and saving it as a state while the computer enters a low-powered mode. Within Windows, there are two general types of disk

suspension: sleep, and hibernation mode. In sleep mode, the current state of the operating system is saved to memory (RAM), and the computer enters a low-powered mode. Upon wake, the state is reloaded relatively quickly, and processes will resume as it was before. In hibernation mode, instead of writing the state to memory, it will be written to the hard drive. This allows the computer to be completely turned off. This mode most likely resumes its previous state much slower than hibernation as it needs to read the entire state from the physical hard drive.

In Linux, there are essentially three types of disk hibernation: suspension to RAM, suspension to the disk, and a hybrid suspension. As with Windows, suspension to RAM allows for the current state of the processes to be saved to memory, and only RAM will receive power. With the suspension to disk, all process states will be saved into an area known as swap space, which could be considered a partition of the hard drive reserved for certain processes such as this, and then it powers off. Hybrid suspension saved the processes states into swap space, but instead of powering off the machine, it suspends to RAM instead. This allows for a long term disk hibernation, but if the computer were to run out of battery, its state will still be in the swap space.

2.3 Differences From Linux

As mentioned earlier in Windows, one of the main features it strives to implement is its ability to port to machines of the same architecture. Either it is not documented, or it is simply not a feature of I/O in Linux, but there doesnt seem to be anything like the portability Windows advertises. In addition to this, Windows has something called the Plug n Play system. This system specializes in automating the process between initializing the interface and adapting to hardware configuration changes for devices.

3 I/O IN FREEBSD

As with Linux, in FreeBSD, the basic model of the I/O system is a sequence of bytes that can be either accessed randomly, or sequentially. However, there are no access methods and no control blocks in a typical UNIX user process. FreeBSD uses file descriptors to access I/O streams. These streams can be represented as three basic objects: a file, a pipe, and a socket. Most processes expect three descriptors to be open: standard input, standard output, and standard error. An I/O stream from one program can be fed as input to almost any other program[16].

3.1 Similarities to Linux

Most of the implementations with respect to I/O are similar in Linux and FreeBSD as they share many aspects especially when being built. Similar to Linux, hardware devices are recognized as either character devices, or block devices.

3.2 Differences from Linux

There may be many differences within the I/O implementations of Linux and FreeBSD, however one notable one is the support of multiple filesystems in FreeBSD. FreeBSD in particular allows filesystems to be loaded dynamically when the filesystems are first referenced by the mount system call [16].

4 WHY THESE SIMILARITIES AND DIFFERENCES EXIST

It seems the Windows has attempted to make their operating system more convenient to the average user, as they have many models which automate processes such as the Plug n Play. However, Linux and FreeBSD being open source

operating systems, these operating systems seem to try to accommodate a wider variety of constructs as a lot of the drivers are written by people around the world, such is open source code. Both operating systems have their pros, but it seems that their differences largely stem from how they view the users experience. The basic ideas such as I/O itself however remain the same.

REFERENCES

[1] ”4.1.5.1. Process creation.” Linux - Process creation. N.p., n.d. Web. 28 Apr. 2017.

[2] ”About Processes and Threads.” About Processes and Threads (Windows). N.p., n.d. Web. 29 Apr. 2017. ¡https://msdn.microsoft.com/en-us/library/windows/desktop/ms681917(v=vs.85).aspx¿.

[3] Arpaci-Dusseau, Remzi, and Andrea C. Arpaci-Dusseau. Operating systems: three easy pieces. S. l.: Arpaci-Dusseau , 2015. Print.

[4] Atwood, Jeff . ”Coding Horror.” Understanding User and Kernel Mode. N.p., n.d. Web. 28 Apr. 2017. ¡https://blog.codinghorror.com/understanding-user-and-kernel-mode/¿.

[5] ”Chapter 15. The Process Address Space.” Chapter 15. The Process Address Space - Shichao’s Notes. N.p., n.d. Web. 29 Apr. 2017. ¡https://notes.shichao.io/lkd/ch15/¿.

[6] ”Completely Fair Scheduler.” Wikipedia. Wikimedia Foundation, 24 Apr. 2017. Web. 27 Apr. 2017.

[7] ”CPU Scheduling.” CPU Scheduling in OS — Operating System Tutorial — Studytonight. N.p., n.d. Web. 27 Apr. 2017. ¡http://www.studytonight.com/operating-system/cpu-scheduling¿.

[8] ”Creating Processes.” Creating Processes (Windows). N.p., n.d. Web. 29 Apr. 2017. ¡https://msdn.microsoft.com/en-us/library/windows/desktop/ms682512(v=vs.85).aspx¿.

[9] McKusick, Marshall K., Keith Bostic, Michael J. Karels, and John S. Quarterman. ”The Design and Implementation of the 4.4BSD Operating System.” 2.4. Process Management. N.p., n.d. Web. 28 Apr. 2017.

[10] McKusick, Marshall Kirk., George V. Neville-Neil, and Robert N. M. Watson. The design and implementation of the FreeBSD operating system. Upper Saddle River: Addison-Wesley/Pearson, 2015. Print.

[11] Tutorialspoint.com. ”Operating System - Processes.” Www.tutorialspoint.com. Tutorials point, n.d. Web. 28 Apr. 2017.

[12] ”Processes, Threads, and Jobs in the Windows Operating System.” Processes, Threads, and Jobs in the Windows Operating System — Microsoft Press Store. N.p., n.d. Web. 28 Apr. 2017

[13] Process Scheduling. N.p., n.d. Web. 27 Apr. 2017. ¡https://www.cs.rutgers.edu/ pxk/416/notes/07-scheduling.html¿.

[14] ”Scheduling.” Scheduling (Windows). N.p., n.d. Web. 28 Apr. 2017. ¡https://msdn.microsoft.com/en-us/library/windows/desktop/ms685096(v=vs.85).aspx¿.

[15] Russinovich, Mark E. and Solomon, David A. and Ionescu, Alex. *Windows internals*. Microsoft Press, 2012.

[16] Marshall Kirk McKusick, George V. Neville-Neil, Robert N.M. Watson. *The Design and Implementation of the FreeBSD Operating System*. Upper Saddle River, NJ, 2015.