Parallel Port Device Drivers: A Study in Driver Creation

Noel Vega
Languages for Embedded Systems Design, Fall, 2002

October 30, 2002

Abstract

If monitors, mice, keyboards, and other computer peripherals could talk to any given computer directly, there would be no compatibility issues between hardware and operating systems. Unfortunately, such a nirvana does not yet exist. As such, computer users are left to deal with device drivers - hard to decipher pieces of code designed to allow peripherals to "communicate" with the operating system and transfer direction to and from the computer. The complexity of device drivers, however, makes it very difficult to begin a process of automating their creation.

This paper looks into parallel port drivers in Free-BSD and Windows NT, deciphering the code to arrive at a generalized driver structure for the two operating systems and, ultimately, a pair of "pseudo" device drivers for the operating systems. By clearly understanding the structure of a device driver, it becomes a lot easier to attack the problem of automating its creation, which in turn will make it easier for users to install hardware on any machine.

1 Introduction

Computer science has been brought to the point where computers are everywhere in society. Certainly, the hardware used to perform the various tasks required of computers needs a means of communication with the computer. This becomes a great strain on hardware manufacturers, who are charged with the task of writing drivers for their hardware for each operating system (specifically, for each *version* of each operating system) they wish their hardware to be compatible with. Given the difficulty of writing device drivers, this can often lead to complications with drivers; since a new driver needs to be written for each device *and* for each OS version, there are bound to be complications with drivers. In fact, Wang, Malik, and Bergamaschi cite a Microsoft report regarding Windows XP which "shows that 61% of XP crashes are caused by driver problems." [3, 1] Clearly, a better understanding of device driver creation is called for.

While there are plenty of device drivers available, there are surprisingly few papers regarding the structure of a device driver. This paper aims to fill this gap, describing driver structures in enough detail to lend to the potential automation of their creation in the future. Such automation would clearly reduce problems with driver creation, perhaps even in an exponential fashion. At a bare minimum, it is useful - and important - to make a clear understanding of device drivers and their operation available to potential driver writers, who may in turn aid hardware manufacturers in creating drivers while the automation process is created.

2 Related Work

A field such as device driver creation, which has shown its purpose for a long time, has a rich history of related work. In hopes of furthering the understanding of the structure of device drivers, Viscarola and Mason offer a comprehensive manual on writing device drivers for Windows NT for "software engineers who have never written a device driver...those who have written drivers on other operating systems, and even...engineers who have already written a few drivers on Windows NT." [4, 1] Clearly, this is a problem even for the most seasoned driver programmer.

While a good deal of time is spent on writing device drivers, there is a growing interest in reducing the amount of time and energy spent writing device drivers. This is best accomplished either 1) through writing so-called "universal" device drivers, or 2) studying the possibility of driver automation. An example of a "universal" device driver is Thesycon's Universal Parallel Port Driver for Windows[2], which is designed to be precisely that: a driver for parallel ports designed to work for Windows NT 4.0, Windows 2000, and Windows XP. More interestingly, however, is the strive to automate driver creation. The process described by Wang, Malik, and Bergamaschi[3] aims to synthesize a platform-independent device driver from specified device behavior.

References

[1] Robert DeLine and Manuel Fähndrich. Enforcing high-level protocols in low-level software. In *PLDI*, Snowbird, Utah, 2001. ACM.

[2] G. Hildebrandt. *Universal Parallel Port Driver for Windows Reference Manual.* Thesycon System Software & Consulting, Germany, March 2002.

[3] Sharad Malik Shaojie Wang and Reinaldo A. Bergamaschi. Modeling and integration of peripheral devices in embedded systems. Technical report, Electrical Engineering Department, Princeton University and IBM T.J. Watson Research Center, 2002. Submitted to the Date 2003 conference, but not accepted as of yet.

[4] Peter G. Viscarola and W. Anthony Mason. *Windows NT Device Driver Development.* New Riders, 2001.

[5] Angel Yu. Custom windows nt 4.0 parallel port device driver: A component of a network performance measurement tool. Master's thesis, California Polytechnic State University, 1998.