Lab One

Marinel Tinnirello

Miranda.Tinnirello@Marist.edu

August 26, 2020

1 Problem One

Q: What are the advantages and disadvantages of using the same system call interface for manipulating both files and devices?

A system call is a program requesting a service from an operating system's kernel. Among its various tasks, an operating system must manage files and devices. There exists an argument, then, that both file and device management could share the same system call interface.

1.1 Advantages

A number of functions are relatively similar (read/write and get/set attributes), so keeping the same interface allows for minimal restructuring between file or device calls. Operating systems tend to abstract the physical properties of any given unit, which makes the notion of treating devices like files, or at least special cases, to be plausible. By adding hardware-specific code to any given device's drivers on top of however the file interface is set up allows for a much cleaner API.

1.2 Disadvantages

Treating devices as though they are files implies a loss of functionalities and performance. Restricting devices to using the pre-established file APIs would mean that there would be issues with some functionalities such as input/output. This could be rectified if some kind of input/output control functions were included to act as a device-specific protocol.

2 Problem Two

Q: Would it be possible for the user to develop a new command interpreter using the system call interface provide by the operating system? How?

A command interpreter, or command line interface, allows for a user to directly enter something to be done by the system. There shouldn't be an issue with a user implementing their own CLI using system calls.

2.1 How to achieve this

Some operating systems implement CLIs by other programs. However, in our operating system, we implement the interpreter in the kernel itself. While this wouldn't make the task impossible, it would be a tedious task for the user to recreate an interpreter on the user-level, but they're welcome to try.

Depending on what the user wants to do, the user would be implementing their CLI via a shell. If the user wants to change commands already built into our operating system, they need to have access to the kernel, which could be an issue. If the user wants to change the names of some programs themselves, there's no need to access the kernel, which makes the task much easier.