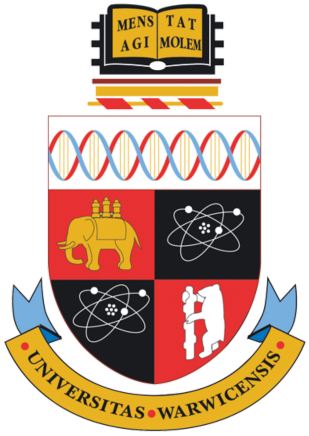

An Educational Kernel for the Raspberry Pi

Thomas Archbold
1602581
Department of Computer Science
University of Warwick

April 29, 2019

CS310 Third-year Project
supervised by Adam Chester

Abstract

Hello there

Index terms— Operating systems, kernel, Raspberry Pi

Contents

Introduction

Operating systems are some of the most pervasive pieces of software around, but due to their inherent complexity, their inner workings are often impenetrable to understand without specialist knowledge. While widespread access to a personal computer is nothing new, the introduction of the Raspberry Pi in recent years has rendered experimentation with computers much more affordable and hence readily available, inviting tinkering at all levels with less concern of economic loss. The Pi therefore provides an ideal platform for operating systems education – novice developers looking to get involved in writing such systems have access to a standardised set of hardware that is inexpensive both to maintain and replace, if and when things go wrong. In the seven years since its initial release the Raspberry Pi has several official operating systems to offer, each addressing its own issue such as ease-of-installation, Internet of Things integration, or classroom management [1], with many more unofficial ones. However, there is less in the way of those written to teach concepts of the operating system itself – this project attempts to fill this gap by providing a configurable, educational operating system for the Raspberry Pi 1 Model B+, with a focus on presenting code which is simple to understand and providing clear interfaces to encourage ease of extensibility, and hence a practical, software-driven approach to learning about operating systems.

Background

Motivation

The main goal of this project is to provide an operating system for the Raspberry Pi which serves as a tool for operating systems education. A key aspect to this is its encouragement to learn-by-doing, and as such the project aims to be both configurable at compile-time as well as open to further extension. While it would largely be possible to implement in a solely emulated environment using a virtual machine, an important aspect is that the project runs on real hardware, namely a Raspberry Pi 1 Model B+ using an HDMI connection for display and a MicroSD card for permanent storage, in order to provide the additional interaction with real-world considerations for operating system development. In order to provide the possibility of extension, the design of generic interfaces for such tasks as CPU scheduling, inter-process, and filesystems was also a key goal, as well as providing a simple interface to manage this.

An operating system draws together aspects from all over the field of computer science, whose development requires intimate knowledge of low-level concepts such as the computer's organisation and architecture, up to an understanding for the more abstract in designing how processes communicate or implementing filesystems. Thus, in additional to the functional aims of the project, the one of key personal motivations behind the project was to gain experience in low-level systems programming and interacting with real-world hardware, as well as in creating an entirely self-contained and useful piece of software.

Relevant Material

To this end, there are a handful of modern resources for getting involved with operating systems development – a particularly useful one at the time of first carrying out research for the project's proposal was `wiki.osdev.org`, which contains information about the creation of operating systems and acts as a

community for hobbyist operating system developers. However, much of the focus is on the x86 platform and past providing a brief overview of the idiosyncrasies of the Raspberry Pi as well as the code to get a barebones kernel to boot, there is little material on the specifics required to get core systems working on the platform. Cambridge's *Baking Pi* [**?**] provides more help in this regard, with Alex Chadwick's comprehensive tutorials proving an invaluable resource for information such as accessing registers and peripherals specific to the Raspberry Pi. The project can, however, be much further extended to guide through the implementation of core operating system concepts such as memory management, the process model, interprocess communication, and filesystems. Another aspect in which this series of tutorials diverges with the goal of this project is the language in which it has been implemented – while assembly is an undeniably language in which to be competent, it is not the most easily-understandable, in stark contrast to what this project hopes to achieve. The resource which aligns most tightly with the aim of this project is [**?**], whose tutorials have served as an outline to how many key features of the project have been implemented.

Finally, other notable resources which are in place to teach general operating systems development are Stanford's *Pintos* [**?**] and Tanenbaum's MINIX operating system; the former was written to accompany the university's CS140 Operating Systems course, while the latter is an illustrative operating system written alongside the book *Operating Systems: Design and Implementation* [**?**], showing how features are implemented in practice. Helin and Renberg's *The Little Book About OS Development* [**?**] also serves as a guide to writing one's own operating system. The only drawback to these three is their focus on the x86 architecture, and while they are useful resources it is in concept only, given the gap which was found to quickly form from focusing on a different processor.

Why is this project worthwhile?

References

[1] "Downloads." `https://www.raspberrypi.org/downloads/`. Page accessed: 2018-11-22.