An educational kernel for the Raspberry Pi

CS310 - Presentation

Thomas Archbold

March 3, 2019

Introduction

Operating systems are some of the most pervasive pieces of software around, but also some of the most complex

The introduction of the Raspberry Pi has made computers much more accessible - allows and encourages experimenting at all levels

There are several official operating systems for the Pi - NOOBS, Raspbian, Windows IoT core, PiNET - but none provide a resource to learn about the operating system itself

Objectives

Main goal: to write a configurable operating system for the Raspberry Pi capable on booting on real hardware for educational and hobbyist use.

Specifically, this involves:

- providing different approaches for tasks such as scheduling, interprocess communication, and permanent storage
- allowing the user to configure the system at compile time to use different combinations of these approaches, and
- exposing a simple and easily extensible interface for additional features

Background material

- Pintos - Stanford's instructional x86 operating system, used to teach their CS140 course
- Baking Pi - Cambridge's tutorial on writing an operating system for the Raspberry Pi in assembly
- MINIX - Tanenbaum and Woohull's illustrative operating system for "Operating Systems: Design and Implementation"
- The little book about OS development - Helin and Renberg's "practical guide to writing your own x86 operating system"

Why is this project worthwhile?

It is more difficult to get into low-level/systems programming $\Rightarrow$ focus is on clear code to aid understanding as much as possible

The project attempts to demystify aspects of operating system development - able to see the theory in practise

Will provide an accessible platform to further tinker and experiment with operating system development, with little to lose

Useful concepts - compilation

Operating System

Freestanding environment

Cross-compiler

Linker

Useful concepts - general OS concepts

Kernel

Exception

Interrupt Request (IRQ)

Process

Concurrency

Context Switch

Synchronisation

Interprocess Communication

Tools used

Languages: C and ARM assembly

Cross-compiler toolchain: `arm-none-eabi`

Build automation: Make

Version control: git

Emulation: QEMU

Model: Raspberry Pi 1 Model B+

Project management

Design overview

Next steps

Conclusions