Operating Systems Course project

Year: 2016/2017

Intro

The project is meant to give you hands on experience with the different parts of the kernel. The Linux kernel is a complex piece of software, actually one of the most complex softwares available today. With around 16 Million LOC, you can easily get lost in the code. [1] Our aim is to apply what we learn in the course in a practical setting. We will use Raspberry Pis which are now available in the lab, as our main platform of development. The aim of the project is to give you an understanding of the kernel internals rather than making you write large chunks of code.

0.1 Practicalities

1. The project is to be done in groups of three.
2. The project is divided in to parts, each part covers part of the course. Plan your time well as this will not be done on the last night of submission.
3. There will be soft deadlines within the project for each part (except for the first two parts). These are not mandatory, but they serve as checkpoints for you. The teachers will give feedback on the submitted solutions after each soft deadline. Each submitted solution should include a well written report, where, well written!=long!
4. On the project submission final deadline, we expect a well written report on your results. These should be as short as possible, but should be sufficient for the teachers to understand your solutions.
5. There will be a demonstration of your project in the labs to the teachers. We will distribute a list of slots where you can demonstrate your solutions. ALL team members need to be there. The teachers might ask you questions on your solutions.

1 Description

The project is divided in modules as follows:

[1] Some Statistics here including, how many profane words used inside :): https://www.linuxcounter.net/statistics/kernel

1.1 Part 1: Learn how to build the kernel for Raspberry Pi3

Following the instructions in this tutorial, you will need to get yourself accustomed on how to build the kernel for Raspberry Pi's. You will use the instructions in this page: https://www.raspberrypi.org/documentation/linux/kernel/building.md

The build should be either on the local Raspberry Pi's (available in the labs) or using cross-compiling on an Ubuntu VM. You are highly advised to use cross-compilation method. Building locally is painfully slow! You will waste your time and will take forever to finish any tests!

1.2 Part 2: Get a working development environment

You might be a vim/EMACS person, but if not and you would like to run with good old eclipse, follow the instructions here:

http://wiki.eclipse.org/HowTo_use_the_CDT_to_navigate_Linux_kernel_source

You will need to install Eclipse CDT, and download the Linux source code for RPi from here: https://github.com/raspberrypi/linux

When doing the setup, please note the architecture of RPi3's. If you set it with a wrong architecture, you will spend painless long hours trying to debug why your code is not working while it is a problem of your compilation. In all cases, I highly advice against using eclipse for cross compiling and I urge you to use the method in module 1 :).

Now you are all set to do stuff!

Another option is to use: https://marketplace.eclipse.org/content/linux-kernel-programming-ide-link-ide