HOW TO ADD NEW PLATFORMS TO THE STH

DIDRIK LUNDBERG
DIDRIKL@KTH.SE

This is a short guide that will cover all steps of adding compatibility with additional platforms for the STH, by giving pointers to what needs to be modified. More detailed descriptions can be found inside example files. Ubuntu 15.04 was used in the writing of this guide, but things should be similar on similar systems. If you have any questions, please send an E-mail to the author. Importantly, if you wonder about how to write any of the files for your platform, please look at corresponding files for the newest platforms (Beagleboard and RPi2) for examples. The first step is always to find the documentation of the platform you want to add.

1. Introduction

Start off by the file `target`. Add the new hardware as a new `PLATFORM` configuration. Here, you will also need to decide what the name of your new platform will be - choose wisely.

2. bin

This directory contains built utilities - nothing you need to modify.

3. core

Here, the majority of the files you need to change in order to build for a new platform are located.

`/config/hw` contains the configuration file of the platform. You will need to create a new one for your platform. Details on what to change can be found inside the existing files.

`/hw/board` contains sub-directories with C files describing board memory layout and board initialization (the latter typically involves nothing). For your new platform, create a new subdirectory with these same files, and edit `board_mem.c`. There, you describe the physical address ranges of peripherals and assign memory to use for the hypervisor and the guests. Look inside the file for more detailed information on how to change it for your platform. Most of it should be automatically correctly filled in from definitions in other files, though.

`/hw/cpu/arm` contains sub-directories for various ARM processors. If you are porting to a platform with a new processor, you need to create a new sub-directory with counterparts to all the files in the other sub-directories here. Porting to a platform with a non-ARM processor will take a bit more work.

Date: August 2, 2015.

/hw/ld contains linker scripts. Confusingly, there were once different linker scripts for different platforms, but in the current version of the hypervisor, you should just use virt-hyper.ld and not create you own linker script here.

/hw/soc contains sub-directories with drivers for all the peripherals of the board. You will need to create a sub-directory for your board, and here is where you will write most of your code. You will need drivers for all peripherals that you will use - at the very least the interrupt controller (for interrupt handling) and UART (for serial output). The necessity of other drivers depend on which board you have and which guests you want to use. Look at existing boards for an idea about which drivers you need to write, and use those files as a basis for yours.

4. doc

This is simply the directory of documentation. If you want to write documentation which does not fit as comments inside the code or in a README file in your directory, you should add it here.

5. drivers

This directory contains drivers used by guests. Currently, the only driver used is for the UART of the SoC. You will need to add a new sub-directory in src with the name of your new platform, and inside that sub-directory create a C file which includes the function printf_putchar, which prints a character over the UART. Look at the drivers for previous platforms for examples.

6. guests

This directory contains the various guest operative systems. You probably do not need to change anything here, if you do not want to make your own custom guest to perform tests. There are four guests in this folder: dtest, linux, minimal, and trusted.

The guests might have small, subtle differences which make them incompatible accress certain platforms but should generally be platform-independent. For example, the dtest guest has hard-coded addresses in two places which needed to be changed for the port from Beagleboard to Raspberry Pi 2 - these are marked by TODO tags.

7. library

This directory contains various utilities. If you construct your own utilities, add them here.

8. rpi2-port

This directory contains files which are helpful when getting Raspberry Pi 2 to work with U-Boot, serial communication, and JTAG. Nothing which you need to change, but if you port it to a new platform you might want to create a similar library.

9. SIMULATION

This directory contains files related to simulation. You do not need to change anything here unless you are interested in simulation.

10. TEMPLATES

Take a look inside the `templates` directory. This directory contains three sub-directories.

`make` contains only contains general templates - you will probably not need to touch those.

`cpu` contains templates with compilation flags for compilation on various CPUs - for example ARM9 and Cortex-A7. If you want to use the hypervisor on a platform with a CPU which is not listed here, you will have to create a new .cfg file corresponding to that processor, with the correct compilation flags for that platform and the correct `TARGET_FAMILY` and `TARGET_CPU` so that the hypervisor knows which processor you are building for.

`platform` contains templates for specific platforms (for example, a platform could be a specific one-chip computer). These specify a platform CPU and not much else. You will need to create a new file here for a new platform. Look to existing configuration files for examples.

11. TEST

This folder contains some scripts for automated testing. Nothing you need to change, but some of these can be useful for rapid testing if you are making updates to the core functionality of the hypervisor.

12. UTILS

This directory, like the `library` directory, contains utilities needed when building the project.