MutekH tutorial for SoCLib platform

This guide explain how to run MutekH on a SoCLib hardware simulator. This is allow easy experimentation with advanced multi-processor programming.

You are **highly encouraged** to first follow the MutekH as Unix process quick start guide which introduces more basic concepts.

MutekH for SoCLib can be compiled for Mips, Arm, PowerPC processors. Other processors are available with different platforms.

The SoCLib platform

The MutekH kernel source code is fully configurable and can be tweaked to adapt hardware platform and application needs. Configuration is handled by a dedicated tool which check dependencies and other relationships between the large set of available configuration tokens.

The example below explains how to setup a SoCLib hardware simulator with 4 RISC processor (Mips, Arm or PowerPC).

Getting SoCLib

We now need to have a working SoCLib install. SoCLib installation is explained here: ?soclib:InstallationNotes

SoCLib platform description

The SoCLib source tree contains a platform dedicated to this tutorial:
?source:trunk/soclib/soclib/platform/topcells/caba-vgmn-mutekh_soclib_tutorial/

Getting the cross-compilers

SoCLib installation nodes already provides a way to get cross-compiler, if you don't already have them, MutekH holds a tool to build a complete cross-compilation toolchain:

The script is in tools/crossgen.mk:

```
 $ tools/crossgen.mk
[prints some help]
 $ tools/crossgen.mk all TARGET=mipsel-unknown-elf
```

The MutekH part

Getting the sources

```
svn co -r 1269 https://www-asim.lip6.fr/svn/mutekh/trunk/mutekh
```

Writing the example source code

Note: This example is available directly from examples/hello directory in source tree.

- Writing the source code in `hello.c`

```
#include <pthread.h>

pthread_mutex_t m;
pthread_t a, b;

void *f(void *param)
{
  while (1)
    {
      pthread_mutex_lock(&m);
      printf("(%i) %s", cpu_id(), param);
      pthread_mutex_unlock(&m);
      pthread_yield();
    }
}
int main()
{
  pthread_mutex_init(&m, NULL);
  pthread_create(&a, NULL, f, "Hello ");
  pthread_create(&b, NULL, f, "World\n");
}
```

- Writing the `Makefile`

```
objs = hello.o
```

Writing the MutekH configuration

The MutekH configuration for the hello application is in the examples/hello/config file.

This file only holds information about the application (here a simple pthread application) and relies upon files in the examples/common directory for the platform definitions.

Have a look to the BuildSystem page for more information about configuration system and configuration file format.

The ?MutekH API reference manual describes all available configuration tokens.

Platform description

The MutekH software uses hardware enumeration to get details about available hardware in the platform, so the `CONFIG_ARCH_DEVICE_TREE` token is defined in the examples/common/platforms-soclib.conf configuration file. It will let the kernel get the platform layout description from a FlattenedDeviceTree which will be built into the kernel.

The build system also compiles the correct FlattenedDeviceTree from the platform name, see examples/common/Makefile.

The used FlattenedDeviceTree source file are in examples/common/: pf_soclib_tutorial_ppc.dts, pf_soclib_tutorial_arm.dts, pf_soclib_tutorial_mips.dts.

Configuring the application along with MutekH

The MutekH kernel and the application may be built out of the source tree.

Change to the SoCLib platform directory and apply the following steps to experiment with out of tree compilation. You have to setup the following variables:

`MUTEKH_DIR`
: Path to MutekH source tree

`APP`
: Path to application source

`CONFIG`
: MutekH configuration file name

`BUILD`
: MutekH build option list (target architecture, cpu type, ?)

These variables are already set in the `config.mk` file to target the hello demo application.

Inside `config.mk`, you'll also find a `CPU` variable that determines which CPU to use in the simulator platform.

See the comments in `config.mk` for more information.

Compiling the application along with MutekH

To compile the kernel with the application, just run `make` with the path to MutekH source directory:

```
$ cd soclib/soclib/platform/topcells/caba-vgmn-mutekh_soclib_tutorial
$ make MUTEKH_DIR=/path/to/mutekh
```

This will build the MutekH kernel along with the application, and the correct simulator.

Execution

Simply run the simulator:

```
$ ./simulation.x
```

You may want to refer to other articles and documents available from the main page to go further with MutekH.

The ?SoCLib home page provides a livecd image with more advanced examples ready to compile and run. These examples are using older MutekH revisions though.

Other more advanced topics and guides are available from the Main page.