

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 started

Of course, you need a working SoCLib install. SoCLib installation is explained here: [?soclib:InstallationNotes](#)

Moreover, you'll need the MutekH source tree and its prerequisites. See [InstallationNotes](#)

The MutekH part

Getting the sources

Even if it is available in newer revisions, this tutorial has been tested and is expected to work well at revision 1269, please try with this one if you have troubles with the last revision.

```
svn co -r 1269 https://www.mutekh.org/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

```

Writing the example source code

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

Compiling application along with the simulator

To compile the kernel with the application, just run `make` with the path to MutekH source directory from the SoCLib platform directory. This single `make` invocation feature is specific to the this SoCLib platform.

```
$ 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 suited simulator.

Separate compilation

You may also compile MutekH and application first and then the standalone SoCLib simulator:

```
$ cd mutekh/
$ make CONF=examples/hello/config BUILD=soclib-arm:pf-tutorial
$ cd soclib/soclib/platform/topcells/caba-vgmn-mutekh_soclib_tutorial
$ make system.x
```

Execution

Simply run the simulator with optional MutekH binary file:

```
$ ./system.x
$ ./system.x path/to/mutekh/hello-soclib-arm.out
```

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](#).