Using ARMSim# on Linux

The open source project, Mono, is an implementation of Microsoft's .NET framework. It can be installed and used to execute the code of the ARMSim# application. **Warning!** Mono does not currently provide all the functionality needed by the docking windows feature of ARMSim#. Docking windows therefore have to be disabled when running under Mono.

If you are an experienced Linix user, you will probably know where you can safely deviate from the following instructions. Otherwise, the safe approach is to follow the instructions below exactly.

Part 1: Install Mono

1. Using the web browser, visit the Mono downloads page at this URL:

   http://www.mono-project.com/download/

2. Select Linux as the platform and choose a download version to match your Linux platform.
3. Download and install `mono` following the instructions provided with the download.

Part 2: Copy and Edit the ARMSim# Files

1. Create a new directory named `dotnet` in your home directory on the computer.
2. Unzip the contents of `ARMSim-201.zip` into a subdirectory of `dotnet` which you can name `ARMSim-201`.
3. The subdirectory should contain the following files:

```
ARMPluginInterfaces.dll
ARMSim.exe
ARMSim.exe.config
ARMSimWindowManager.dll
DockingWindows.dll
StaticWindows2.dll
```

4. Copy the file `arm-none-eabi-32` if you are running a 32-bit Linux system or the file `arm-none-eabi-64` if you have a 64-bit system into the `ARMSim-201` subdirectory. Rename that file to `arm-none-eabi-as`; the `mv` command may be used:

```
cd ~/dotnet
mv arm-none-eabi-* arm-none-eabi-as
chmod +x arm-none-eabi-as
```

Part 3: Create a Shell Script to Invoke ARMSim#

1. Using a text editor (e.g. `vi` again) create a file named `ARMSim` which contains these lines:

```
#!/bin/sh
mono ~myusername/dotnet/ARMSim-201/ARMSim.exe \
    &> ~myusername/.ARMSimLogFile
```

   where *myusername* should be replaced with your user name on the Linux system. You can place this new file in a directory where locally installed commands are placed. A typical such directory is named `/usr/local`. This directory should be one of those listed in the `$PATH` shell variable.

2. Use the `cd` command to navigate to the folder where you placed the `ARMSim` file and mark it as executable with the `chmod` command, as follows:

```
cd /usr/local
chmod +x ARMSim
```

3. You should now have a new command available named `ARMSim`.

Part 4: An Initial Run

1. Copy the file `Angel_print_int.s` to your Linux computer.
2. Start up ArmSim# by entering the command `ARMSim` icon in a console terminal window.
3. Select the `Plugins` tab on the window opened by clicking on `File/Preferences`. Check the box for AngelSWIInstructions if it is not already checked. Click OK in the bottom right corner.
4. Click on `File/Load` and navigate to the `Angel_print_int.s` file which you copied in step 1. Open it.
5. After a brief delay, the ARMSim# display windows should be filled with an assembly source listing, the initial register contents, and more.
6. Click on the *Run* button, which is the right pointing triangle near the top-left of the ARMSim# window.

7. The output window at the bottom labelled "`stdin/stdout/stderr`" should contain the output from the assembler program. The desired output comprises four decimal numbers, one per line, which are `+255 +2000000001 +0 -255`

Notes

- If ARMSim# crashes, the error message generated by mono will be found in the file named `.ARMSimLogFile` in the home directory.
- Even when ARMSim# is exited normally, an error message reporting a `System.NullReferenceException` may be saved into the logfile. This appears to be caused by a bug in the Mono implementation of .Net and occurs when ARMSim# attempts to close its windows and exit. This exception does not occur with ARMSim# running on Microsoft Windows.