2014-06-18

Work summary

Today, I have:

- Integrated Mono HEAD as it contained updates to the ARM backend;
- Started running Mono's NUnit-based test suites on one of the Intel VMs (`thaz`). For the `corlib_test_net_4_5.dll` suite, the initial results were:

  ```
  Tests run: 9913, Failures: 3, Not run: 100
  ```

  Guesswork and investigation revealed that the failures were due to Mono running within a priviledged user account. Running from a fresh user account resulting in a single (but different) failure :

  ```
  Tests run: 9913, Failures: 1, Not run: 100
  ```

  It turns out that spurious errors are produced if `$TMPDIR` is set to something else than `/tmp`; unsetting that variable led to:

  ```
  Tests run: 9913, Failures: 0, Not run: 100
  ```

  Note: test logs have been pushed to:

  http://dd.crosstwine.com/git/?p=kitsilano/mono-tizen/make-check-logs.git

- Ran other test suites on the same platform.

  The `System.Runtime.Serialization.Formatters.Binary` suite completes successfully:

  ```
  Tests run: 10, Failures: 0, Not run: 0
  ```

  But `System_test_net_4_5.dll` doesn't:

  ```
  Tests run: 6059, Failures: 3, Not run: 119
  ```

  Note, however, that the 3 failures are all related to the same cause:

  ```
  System.Net.Sockets.SocketException : An address   \
  incompatible with the requested protocol was used
  ```

  TODO: Investigate.

- Started running NUnit-based suites on one of the ARM VMs (`teiz`). The results were much, much uglier, including an internal compiler error:

  ```
  MCS     [net_4_5] corlib_test_net_4_5.dll
  Test/Mono/DataConvertTest.cs(64,77): error CS0589: Internal \
  compiler error during parsingSystem.FormatException: Input  \
  string was not in the correct format
    at System.Double.Parse (System.String s, NumberStyles     \
    style, IFormatProvider provider) ...
  ```

  Intrumenting the Mono binary gave a hint of what was happening. Fractional numbers were being truncated, and a failure flag was set:

  ```
  4.9406564584124650e-324 => 4, FAIL
  ```

  This is due to this fragment of code in `mono/metadata/icall.c`:

  ```
  #ifdef __arm__
          if (*ptr)
                  *result = strtod (ptr, &endptr);
  #else
          if (*ptr){
                  /* mono_strtod () is not thread-safe */
                  EnterCriticalSection (&mono_strtod_mutex);
                  *result = mono_strtod (ptr, &endptr);
                  LeaveCriticalSection (&mono_strtod_mutex);
          }
  #endif
  ```

  Mono uses its own (locking!) `mono_strtod` on most platforms, but a direct `strtod` on ARM—presumably because it is faster. The problem, of course, is that while (Android) has a "dumb" libc, Tizen's uses the full glibc, which implements locale-sensitive treatment!

  Forcing `LANG=C` made the problem disappear. TODO: Rework the compile-time conditional to be Android-specific; push patch upstream;

- With the problem above cleared, NUnit loads and the `corlib_test_net_4_5.dll` suite starts running—but hangs (no disk activity, 0% CPU) after successfully running 2009 tests.

  TODO: Investigate.

- Observed the same results on `thoz`.

In parallel, I have:

- Built a new VM, `thoz`, based on `tizen-2.2_20130719.3_RD-PQ`. This image is very close or identical to what Bob's RD-PQ device is currently running.

  It exhibits the exact same problem as `teiz` (Tizen 3.0/Next) when running the test suites—which is simultaneously good and bad news;

- Uploaded an ARM tarball of the compiled Mono/Tizen 2.2/ARM, which should run (but not pass the whole test suite) on Bob's phone *as of today*:

  http://phio.crosstwine.com/tmp/thoz-2014-06-18.tar.gz

- Started investigating RPM packaging of Mono. The current plan is to start (and diverge as little as possible) from SUSE's `mono-common.spec` file;

- Figured out how to push a very de-duplicated form of all these VM images over the network. Still not ready, but getting closer.

Tomorrow, I plan to:

- Prepare a patch for the `strtod` failure identified above;
- Ask which of the `*.exe` basic tests failures are expected on the ARM architecture, if any, on the Mono development list, and start investigating them (was hoping to do this today, but it didn't happen);
- Investigate the hanging test suite on ARM platforms;
- Investigate the `SocketException` issue the Intel platform.

VMs & environments

For reference, here is the list of VMs and environments used for these tests. The VM images are regularly pushed to the following Git-Annex repository:

http://dd.crosstwine.com/git/?p=kitsilano/mono-tizen/vms.git

VMs

Name	Arch	Kernel	Base image
`tizn`	`armv7l`	`3.2.0-4-vexpress`	`tizen_20140602.4_RD-PQ`
`teiz`	`armv7l`	`3.2.0-4-vexpress`	`tizen_20140602.4_RD-PQ`

Name	Arch	Kernel	Base image
`thaz`	`i586`	`bzImage.x86`	`emulimg-2.2.x86`
`thuz`	`i586`	`bzImage.x86`	`emulimg-2.2.x86`
`thoz`	`armv7l`	`3.2.0-4-vexpress`	`tizen-2.2_20130719.3_RD-PQ`

Machines

Machine	Arch	OS
`mini`	`x86_64`	Debian Testing
`kaia`	`x86_64`	Mac OS X Mavericks