Building Stroika

Common

Stroika is a C++ class library. For the most part, it's built make. However, internally, some of these make rules use perl etc scripts.

Quick Start

For the very impatient

```
wget https://github.com/SophistSolutions/Stroika/archive/V2-Release.tar.gz
tar xf V2-Release.tar.gz
make --directory Stroika-2-Release all run-tests
```

For the more patient (hints about what to try next)

- wget https://github.com/SophistSolutions/Stroika/archive/v2.0a120.tar.gz
  or
  wget https://github.com/SophistSolutions/Stroika/archive/V2-Release.tar.gz
- tar xf V2-Release.tar.gz
- cd Stroika-2.0a120 (or whatever extracted)
- make help
  Not needed, but gives some idea of make options
- make check-tools
  Not needed, but tells you if you are missing anything critical
- make default-configurations
  Not needed, but it's a springboard for setting up the configuration you want.
  - Review/edit ConfigurationFiles/Debug.xml
  - Or try something like
    - configure MyGCC52Config -assertions enable --trace2file enable --compiler-driver 'g++-5.2'

    or
    - configure MyCPP17Test --assertions enable --trace2file enable *--cppstd-version-flag '--std=c++1z'*

    or
    - configure --help
- ls ConfigurationFiles/
  - See what configurations you've created. Edit files, add or delete.
-
- make all CONFIGURATION=*a-config*
  - make all targets for the configuration named *a-config*
- make all
  Builds everything for ALL configurations (in the folder ConfigurationFiles/)
- make run-tests
  Runs regression tests (optionally on remote machines, or with VALGRIND)

Required Tools

Required for ALL platforms

- git
    - not needed to build, but
- make
- patch
- perl
- realpath
- sed
- tar
- wget

For Windows

- Visual Studio.net 2013 (or later)
- Cygwin

  Including
    - dos2unix
    - unix2dos

For UNIX

- Compiler
    - gcc 4.9 or later OR
        - Stroika is currently tested with gcc 4.9, 5.2
    - llvm (clang++) 3.5, 3.6 or later

Optional Components

- curl

  If present, Stroika can be configured to include it and take advantage of it

- openssl

  You can use the statically linked copy in ThirdPartyProducts, or the os-intalled .so files.

Build / Configuration Design

<<<ROUGH DRAFT>>>

Alternatives

We seriously considered a number of build systems, including cmake, ant, perl scripts, qmake, etc. They all weak, with ant possibly being the best alternative, except that its heavily java oriented.

Just normal GNU make – appears to be the least bad alternative, so that's what we're doing.

STATUS

There are remnants of earlier build systems (especially perl), but they will soon be eliminated, and only about 40% of what is described here is actually implemented right now.

Configurations

The design is to make a set of configuration files – each stored in the Configurations directory. Configurations are described by a single XML file. They can be generated using make define-configuration (or TBD).

A configuration comprises BOTH target platform (e.g. windows/linux), hardware (e.g. PowerPC, x86, x64, ARM), and any other options (debug build, enable assertions, support OpenSSL, etc).

This stands in contrast to several other systems (like Visual Studio.net) that treats platform and 'configuration' as orthogonal choices.

You can call

```
make apply-configurations
```

to generate all the directories and files dependent on the defined configurations. Note – this is generally not necessary, and called automatically.

Build Results

Intermediate files (objs etc) go into **IntermediateFiles**. Final build products (libraries and executables) go into **Builds**. Subdirectories of IntermediateFiles and Builds are named by each configuration, and the data (objs, or exes etc) are layed out under {Builds,IntermediateFiles}/{ConfiguraitonName}.

Build Process

On any platform, building Stroika, and all is demo applications and regression tests is as simple as cd'ing to the top-level directory, and typing make

Special Targets

- make
  Make with no arguments runs 'make help'
- make help
  Prints the names and details of the special targets
- make all
  builds the stroika library, tests, demos, etc.
- make run-tests
  Builds Stroika, and all the regression tests, and runs the regression tests
- make project-files
  Builds project files which can be used for things like visual studio (not needed)
- make check-tools
  Checks if the tools needed to build Stroika are installed and in your path. This is done automatically, and generally not needed explicitly.

Configuration

Building Stroika requires building special configuration files. But using the above mechanism automatically builds them for you. To customize your Stroika configuration, you can manually run

- Configurations can be generated via
  - make default-configurations
    - This generates a number of configuration XML files in ConfigurationFiles/.
    - Feel free to edit this by hand, or create more with ./configure, or delete them.
- Configurations can be applied via
  - make apply-configurations
    - This generates makefiles (as appropriate for each *config.xml* file in ConfigurationFiles/), and C++ #include files

Using Visual Studio.net

Visual Studio.net project and solution files are available for the Stroika demos, top-level project files, and regression tests. Once you have built your configuration files (see above), you can use the project files to build, test, extend and develop Stroika.

Using QtCreator (on unix)

Run Library/Projects/QtCreator/CreateQtCreatorSymbolicLinks.sh to create project files at the top level of your Stroika directory. Then you can open that .creator file in qtCreator, and build and debug Stroika-based applications.

Common Errors

Tar failure

Errors about invalid parameters, and/or bad blocks can usually be fixed by installing a copy of gnu tar. We've tested 1.27.

cp: illegal option –

Install a copy of GNU cp