GuixBuildSystem

Use cases

Guix can be used for faster testing:

- It can build software that uses autotools and Android.mk Makefiles
- We might be able to use it to iterate over different Android versions, to test libsamsung-ril changes for various RIL API way faster (without even having to use it in a specific Replicant version, use repo, etc).
- It's more self contained than the whole Replicant source code

TODO:

- Import the compilation flags for various Replicant versions like -Werror -Wnounused to make sure that everything is usable in all Replicant versions.
- Finish packaging libsamsung-ril
- Add more automatic testing either in libsamsung-ipc / libsamsung-ril and or in the unofficial Guix packages.

Implementation

Guix currently uses android-make-stub to build Android software using Android.mk.

There are some comments in Guix source code that cross compilation isn't supported for the android-ndk build system.

Practically speaking it means that Guix won't be able to build any BUILD_SHARED_LIBRARY, BUILD_SHARED_LIBRARY or BUILD_STATIC_EXECUTABLE, however building the HOST_* counterpart works fine.

To workaround that they simply do replace them in the Android.mk during the build procedude:

- BUILD_SHARED_LIBRARY becomes BUILD_HOST_SHARED_LIBRARY
- BUILD_SHARED_LIBRARY becomes BUILD_HOST_STATIC_LIBRARY
- BUILD_STATIC_EXECUTABLE becomes BUILD_HOST_STATIC_EXECUTABLE

For instance we have:

```
    (arguments
     `(#:phases
       (modify-phases %standard-phases
              (delete 'bootstrap)
              (add-before 'build 'patch-host
                  (lambda _
                    ;; TODO: Cross-compile.
                    (substitute* "Android.mk"
                         (("BUILD_SHARED_LIBRARY") "BUILD_HOST_SHARED_LIBRARY")
                         ;;  (("BUILD_STATIC_LIBRARY") "BUILD_HOST_STATIC_LIBRARY")
                         ;;  (("BUILD_STATIC_EXECUTABLE") "BUILD_HOST_STATIC_EXECUTABLE")
                         )
                    #t))
              )))
```

Here the (delete 'bootstrap) is because the build is done in several phases, and here the bootstrap phase wasn't overriden by the android-ndk build system yet. As a result it tried to run autogen.sh. That might be because in libsamsung-ipc there is both an android build system and an autotools based one.

Example

- There is some work in progress to package libsamsung-ril for faster testing: https://git.replicant.us/contrib/GNUtoo/guix/tree/gnu/packages/replicant.scm?h=libsamsung-ril

Additional work

There is some work in progress by Julien Lepiller / roptat to package more Android components (including build systems like blueprint / soong) in this android-build.scm .