buildlink3: Implementation

Johnny Lam
jlam@NetBSD.org

Outline

- wrapper scripts
- bsd.buildlink3.mk

wrapper scripts

- `wrapper.sh` – main driver script that sources sub-scripts
- `marshall` – deals with consecutive arguments that must be treated specially
- `buffer` – expands some single arguments into multiple consecutive arguments
- `cache` – caches the result of argument transformations
- `logic` – transforms the arguments
- `buildcmd` – appends the transformed arguments to the command line
- `reorderlibs` – optionally changes the order that libraries on the linker command line

wrapper.sh

- This is the file that's copied into the buildlink directory as `${CC}`, `${CXX}`, `${LD}`, etc.
  - The executables to replace with a wrapper script are named in `_BLNK_WRAPPEES`.
  - We also symlink each executable to common names, e.g. `cc`, `c++`, `gcc`, `g++`, `CC`, etc.
- Sets global variables then calls the rest of the scripts to do the real work.
- Writes out the commands executed to a work log, `${WRKLOG}`, for debugging purposes.
  - Should add way to log to stderr so we can just capture all of the output of a build into a single file.

libtool.sh, libtool-*

- `wrapper.sh` replacement for libtool that calls a few libtool-specific sub-scripts.
- libtool`-fix-la`
  - Modifies `dependency_libs` and `relink_command` in uninstalled `*.la` files so that when linking against the libtool archive or when installing it, the libraries in work directory are used.
  - Removes redundant or useless options to make smaller, cleaner libtool archives.
    - No more `-lm -lm -lm -lm -lm -lm -lm -lm -lm -lm -lm` as used to be found in KDE libtool archives – the extra libraries are optimized away.

libtool.sh, libtool-* (cont.)

- `libtool-post-cache`, `libtool-post-logic`
  - Workaround authors that don't follow the libtool documentation and link against uninstalled libtool archives with `-L../path/to/src/dir -lfoo`.
    - Replace with `../path/to/src/dir/libfoo.la`
    - Works by remember all local directories passed via `-L`, and checking those directories for `libfoo.la` when `-lfoo` is encountered.
    - This is needed to work properly with the way `libtool-fix-la` modifies the uninstalled `*.la` files.

marshall

- Merges consecutive arguments together, e.g. `-I /dir`, `-Wl,-R -Wl,/dir`.
- Skips over arguments that shouldn't be processed by the `logic` script.
  - Darwin's GCC uses some special options that are similar in nature to setting the rpath, and those named paths need to be skipped.

buffer

- Grab arguments off the command line (through marshall) and modify the number and types of options passed along.
  - `-R/path1:/path2:/path3` must be split up into `-R/path1 -R/path2 -R/path3` or else the sed script that does the transformations will break.
- There's actually a fairly clever stack implementation in this script.

gen-transform.sh

- This script generates two sed scripts and the `reorderlibs` shell script.
  - `.transform.sed`
    - Used by the `logic` script to transform arguments.
  - `.untransform.sed`
    - Used by `bsd.buildlink3.mk` to unbuildlinkify files before installation.
  - `reorderlibs`
    - Changes the order of `-l` options on the command line, e.g. ensure `-lcrypt` comes before `-lcrypto` when both are present.
- {_BLNK,BUILDLINK}_TRANSFORM contain the commmands that specify the contents of the sed scripts.
  - Order of commands is very important!

bsd.buildlink3.mk

- We compute a lot of variables' values using shell commands, so we save their values using BUILDLINK_VARS into a file that, if it exists, is sourced at the start of bsd.buildlink3.mk.
  - We don't use MAKEFLAGS since it's easy to overflow the command line with all of the variables and values computed for buildlink3.
  - This technique can be generalized and put into bsd.pkg.mk.
- Work flow (top to bottom)
  - bsd.builtin.mk
  - Dependency reduction
  - Set CFLAGS, CPPFLAGS, LDFLAGS, etc.
  - Generate the wrapper scripts
  - Populate the buildlink directory

_BLNK_PACKAGES, _BLNK_DEPENDS

- These are the key variables used in `.for` loops that control most of bsd.buildlink3.mk.
- `_BLNK_PACKAGES`
  - Lists all direct and indirect dependencies for the package being built.
  - Built up via the `BUILDLINK_PACKAGES` variable in each `buildlink3.mk` file.
  - Ordered so that at any point in the list, the packages listed after that point don't depend on packages listed before that point.
  - Used to determine what flags to add to CFLAGS, etc. and which files to symlink into the buildlink directory.
- `_BLNK_DEPENDS`
  - Lists only the direct dependencies for the package being built.
  - Built up via the `BUILDLINK_DEPENDS` variable in each `buildlink3.mk` file, which is guarded by `BUILDLINK_DEPTH` to prevent recursive dependencies.
  - Ordered in the same way as `_BLNK_PACKAGES`
  - Used to generate appropriate `DEPENDS+=...` and `BUILD_DEPENDS+=...`

BUILDLINK_..., _BLNK_...

- BUILDLINK_... are public variables
- _BLNK_... are variables private to buildlink3 implementation
- For each private variable, there is usually a public variable with a similar name
  - e.g. _BLNK_PASSTHRU_DIRS & BUILDLINK_PASSTHRU_DIRS
  - The private variable extends and cleans up the value of the public one

bsd.builtin.mk

- Check for built-in software that satisfy dependencies.
  - Include `builtin.mk` files for every package listed in `_BLNK_PACKAGES`.
  - `builtin.mk` files may include `buildlink3.mk` files, so the value of `_BLNK_PACKAGES` may be different between before and after this file is included (subtle but important!)
- `PREFER_{PKGSRC,NATIVE}` are set from `/etc/mk.conf`
  - The most specific package listing has the greatest precedence, and in case of a tie, `PREFER_PKGSRC` wins (subtle but important!)
    - e.g. `PREFER_PKGSRC=yes`, `PREFER_NATIVE=getopt` means that we use the pkgsrc software for everything except if the native getopt satisfies a getopt dependency
- USE_BUILTIN.<pkg>
  - Set by bsd.builtin.mk from the values of `PREFER_{PKGSRC,NATIVE}`
  - Used within `<pkg>/builtin.mk` to determine whether to allow the built-in software to satisfy the dependency or not

Dependency reduction

- Tries to get rid of redundant dependencies so that the list of dependency requirements stored in the package meta-files is simpler.
  - `foo>=1.0`, `foo>=1.1nb3`, `foo>=1.3` can be optimized away to just `foo>=1.3`.
  - Can only handle >= dependency requirements for now.
  - Should be a SMOP to handle more complicated dependencies.
    - Hard to handle C-shell-style glob patterns.

Set CFLAGS, CPPFLAGS, LDFLAGS, etc.

- These are the values used by GNU configure scripts to fill in values in `*.in` files.
- Should refer to the true installed locations of the headers and libraries
  - In the pkgviews case, the true locations are within the depot directories of each of the dependencies

Generate the wrapper scripts

- Build up `_BLNK_TRANSFORM` list passed along as commands to `gen-transform.sh`.
- We need to preserve rpaths from transformation by the `logic` script.
  - _BLNK_PASSTHRU_RPATHDIRS contains the list of directory paths allowed as an rpath. Important: anything not listed here is removed.
  - Mangle the paths so the meat of the transformation script won't alter them, then de-mangle the paths at the end.
    - `mangle`, `sub-mangle`, `rpath`, `sub-rpath`
    - Cumbersome: need to find a simpler way
- There are different ways to piece together a wrapper script
  - Private cache consulted before the common cache
  - Custom logic scripts executed after the main logic script to refine the argument transformations

Populate the buildlink directory

- Symlink headers, libraries, and `*.pc` pkgconfig data files into the buildlink directory.
- `_BLNK_FILES_CMD.<pkg>`
  - Shell command that lists the files to be symlinked.
  - Defaults to extracting the `+CONTENTS` file of the installed package via `pkg_info(1)` and grepping for the appropriate files.
  - Can tweak this variable to different degrees using `BUILDLINK_FILES.<pkg>` and `BUILDLINK_FILES_CMD.<pkg>`.
  - Important: remember that built-in headers and libraries aren't symlinked into the buildlink directory. If you need it to appear there, then you need to create your own target to make it happen.
- Libtool archives
  - Create new `*.la` files in the buildlink directory by replacing all references to outside directories with ones into the buildlink directory.
    - This makes `libtool(1)` think the real libraries are actually in the buildlink directory and won't go searching elsewhere for libtool archives.

Things that I glossed over

- Other wrapper script pieces (but they're very straightforward)
- `x11-links/buildlink3.mk` and how `${X11BASE}` is handled
  - This is undergoing some changes at the moment – reed@NetBSD.org and xtraeme@NetBSD.org are doing testing.
- How buildlink3 makes package views work, e.g. `${DEPOTBASE}` handling.
  - Tune in tomorrow

Summary

- Now you know everything. *Go forth and multiply.*