**build-package-data.mk**

define build-package-data

$(call trace, build-package-data($1,$2,$3))

$(call profStart, build-package-data($1,$2,$3))

# args:

# $1 = dir

# $2 = distdir

# $3 = GHC stage to use (0 == bootstrapping compiler)

buildPackageData :: Package -> TodoItem -> Rules ()

buildPackageData pkg @ (Package name path todo) (stage, dist, settings) =

$1\_$2\_CONFIGURE\_OPTS += --disable-library-for-ghci

ifeq "$$(filter v,$$($1\_$2\_WAYS))" "v"

$1\_$2\_CONFIGURE\_OPTS += --enable-library-vanilla

ifeq "$$(GhcWithInterpreter)" "YES"

ifneq "$$(DYNAMIC\_GHC\_PROGRAMS)" "YES"

$1\_$2\_CONFIGURE\_OPTS += --enable-library-for-ghci

endif

endif

else

$1\_$2\_CONFIGURE\_OPTS += --disable-library-vanilla

endif

ifeq "$$(filter p,$$($1\_$2\_WAYS))" "p"

$1\_$2\_CONFIGURE\_OPTS += --enable-library-profiling

else

$1\_$2\_CONFIGURE\_OPTS += --disable-library-profiling

endif

ifeq "$$(filter dyn,$$($1\_$2\_WAYS))" "dyn"

$1\_$2\_CONFIGURE\_OPTS += --enable-shared

else

$1\_$2\_CONFIGURE\_OPTS += --disable-shared

endif

let argEnable x suffix = arg [(if x then "--enable-" else "--disable-") ++ suffix]

in mconcat

[ argEnable False "library-for-ghci"

, argEnable (vanilla `elem` ways) "library-vanilla"

, when (GhcWithInterpreter && not DynamicGhcPrograms && vanilla `elem` ways) $

argEnable True "library-for-ghci"

, argEnable (profiling `elem` ways) "library-profiling"

, argEnable (dynamic `elem` ways) "shared"

]

ifeq "$$(HSCOLOUR\_SRCS)" "YES"

$1\_$2\_CONFIGURE\_OPTS += --with-hscolour="$$(HSCOLOUR\_CMD)"

endif

when HsColourSrcs $ with HsColour

# We filter out -Werror from SRC\_CC\_OPTS, because when configure tests

# for a feature it may not generate warning-free C code, and thus may

# think that the feature doesn't exist if -Werror is on.

$1\_$2\_CONFIGURE\_CFLAGS = $$(filter-out -Werror,$$(SRC\_CC\_OPTS)) $$(CONF\_CC\_OPTS\_STAGE$3) $$($1\_CC\_OPTS) $$($1\_$2\_CC\_OPTS) $$(SRC\_CC\_WARNING\_OPTS)

cflags = mconcat

[ commonCcArgs `filterOut` ["-Werror"]

, argOption $ ConfCcArgs stage

, customCcArgs settings

, commonCcWarninigArgs

]

$1\_$2\_CONFIGURE\_LDFLAGS = $$(SRC\_LD\_OPTS) $$(CONF\_GCC\_LINKER\_OPTS\_STAGE$3) $$($1\_LD\_OPTS) $$($1\_$2\_LD\_OPTS)

ldflags = mconcat

[ commonLdArgs

, argOption $ ConfGccLinkerArgs stage

, customLdArgs settings

]

$1\_$2\_CONFIGURE\_CPPFLAGS = $$(SRC\_CPP\_OPTS) $$(CONF\_CPP\_OPTS\_STAGE$3) $$($1\_CPP\_OPTS) $$($1\_$2\_CPP\_OPTS)

cppflags = mconcat

[ commonCppArgs

, argOption $ ConfCppArgs stage

, customCppArgs settings

]

$1\_$2\_CONFIGURE\_OPTS += --configure-option=CFLAGS="$$($1\_$2\_CONFIGURE\_CFLAGS)"

$1\_$2\_CONFIGURE\_OPTS += --configure-option=LDFLAGS="$$($1\_$2\_CONFIGURE\_LDFLAGS)"

$1\_$2\_CONFIGURE\_OPTS += --configure-option=CPPFLAGS="$$($1\_$2\_CONFIGURE\_CPPFLAGS)"

# Also pass these as gcc-options, because Cabal uses them to check for

# the existence of foreign libraries.

$1\_$2\_CONFIGURE\_OPTS += --gcc-options="$$($1\_$2\_CONFIGURE\_CFLAGS) $$($1\_$2\_CONFIGURE\_LDFLAGS)"

argConf key args = joinArgs $ arg ["--configure-option=", key, "="] <> joinArgsWithSpaces args

[ argConf "CFLAGS" cflags

, argConf "LDFLAGS" ldflags

, argConf "CPPFLAGS" cppflags

, joinArgs $ mconcat [arg ["--gcc-options="], cflags, arg [" "], ldflags

ifneq "$$(ICONV\_INCLUDE\_DIRS)" ""

$1\_$2\_CONFIGURE\_OPTS += --configure-option=--with-iconv-includes="$$(ICONV\_INCLUDE\_DIRS)"

endif

ifneq "$$(ICONV\_LIB\_DIRS)" ""

$1\_$2\_CONFIGURE\_OPTS += --configure-option=--with-iconv-libraries="$$(ICONV\_LIB\_DIRS)"

endif

ifneq "$$(GMP\_INCLUDE\_DIRS)" ""

$1\_$2\_CONFIGURE\_OPTS += --configure-option=--with-gmp-includes="$$(GMP\_INCLUDE\_DIRS)"

endif

ifneq "$$(GMP\_LIB\_DIRS)" ""

$1\_$2\_CONFIGURE\_OPTS += --configure-option=--with-gmp-libraries="$$(GMP\_LIB\_DIRS)"

endif

argConfWith key opt = do

value <- option opt

when (value /= "") $ argConf ("--with-" ++ key) $ arg [value]

, argConfWith "iconv-includes" IconvIncludeDirs

, argConfWith "iconv-libraries" IconvLibDirs

, argConfWith "gmp-includes" GmpIncludeDirs

, argConfWith "gmp-libraries" GmpLibDirs

ifeq "$$(CrossCompiling)" "YES"

$1\_$2\_CONFIGURE\_OPTS += --configure-option=--host=$(TargetPlatformFull)

endif

ifeq "$3" "0"

$1\_$2\_CONFIGURE\_OPTS += $$(BOOT\_PKG\_CONSTRAINTS)

endif

$1\_$2\_CONFIGURE\_OPTS += --with-gcc="$$(CC\_STAGE$3)"

ifneq "$3" "0"

# There is no LD\_STAGE0, Cabal will figure it out

$1\_$2\_CONFIGURE\_OPTS += --with-ld="$$(LD\_STAGE$3)"

endif

$1\_$2\_CONFIGURE\_OPTS += --configure-option=--with-cc="$$(CC\_STAGE$3)"

$1\_$2\_CONFIGURE\_OPTS += --with-ar="$$(AR\_STAGE$3)"

$1\_$2\_CONFIGURE\_OPTS += $$(if $$(ALEX),--with-alex="$$(ALEX)")

$1\_$2\_CONFIGURE\_OPTS += $$(if $$(HAPPY),--with-happy="$$(HAPPY)")

, when CrossCompiling $ argConf "--host" $ argOption $ TargetPlatformFull

, argConf "--with-cc" $ argPath Gcc

, when (stage == Stage0) $ bootPkgConstraints

, with Gcc

, when (stage /= Stage0) $ with Ld

, with Ar

, with Alex

, with Happy

ifneq "$$(BINDIST)" "YES"

ifneq "$$(NO\_GENERATED\_MAKEFILE\_RULES)" "YES"

-- Not needed with Shake (?)

$1/$2/inplace-pkg-config : $1/$2/package-data.mk

$1/$2/build/autogen/cabal\_macros.h : $1/$2/package-data.mk

((path </> dist) </>) <$>

[ "package-data.mk",

"haddock-prologue.txt",

"inplace-pkg-config",

"setup-config",

"build" </> "autogen" </> "cabal\_macros.h"

] &%> \\_ -> do

# This rule configures the package, generates the package-data.mk file

# for our build system, and registers the package for use in-place in

# the build tree.

$1/$2/package-data.mk : $$$$(ghc-cabal\_INPLACE) $$($1\_$2\_GHC\_PKG\_DEP) $1/$$($1\_PACKAGE).cabal $$(wildcard $1/configure) $$(LAX\_DEPS\_FOLLOW) $$$$($1\_$2\_HC\_CONFIG\_DEP)

need [path </> name <.> "cabal"]

when (doesFileExist $ path </> "configure.ac") $ need [path </> "configure"]

-- the rest of the dependencies are implied by **run** and **with** commands

# Checking packages built with the bootstrapping compiler would

# generally be a waste of time. Either we will rebuild them with

# stage1/stage2, or we don't really care about them.

ifneq "$3" "0"

ifneq "$$($1\_NO\_CHECK)" "YES"

"$$(ghc-cabal\_INPLACE)" check $1

endif

endif

-- Skipping as advised by Simon Marlow

"$$(ghc-cabal\_INPLACE)" configure $1 $2 "$$($1\_$2\_dll0\_MODULES)" --with-ghc="$$($1\_$2\_HC\_CONFIG)" --with-ghc-pkg="$$($1\_$2\_GHC\_PKG)" $$($1\_CONFIGURE\_OPTS) $$($1\_$2\_CONFIGURE\_OPTS)

run GhcCabal cabalArgs

cabalArgs = mconcat $

[ arg ["configure", path, dist]

, joinArgsWithSpaces $ customDllArgs settings

, with $ Ghc stage

, with $ GhcPkg stage

, customConfArgs settings

-- also see $1\_$2\_CONFIGURE\_OPTS comments above

ifeq "$$($1\_$2\_PROG)" ""

ifneq "$$($1\_$2\_REGISTER\_PACKAGE)" "NO"

$$(call cmd,$1\_$2\_GHC\_PKG) update --force $$($1\_$2\_GHC\_PKG\_OPTS) $1/$2/inplace-pkg-config

endif

endif

endif

endif

when (registerPackage settings) $ run (GhcPkg stage) ghcPkgArgs

ghcPkgArgs = mconcat $

[ arg ["update", "--force"]

, when (stage == Stage0) $ arg ["--package-db=libraries/bootstrapping.conf"]

, arg [path </> dist </> "inplace-pkg-config"]

]

PACKAGE\_DATA\_MKS += $1/$2/package-data.mk

$(call profEnd, build-package-data($1,$2,$3))

endef

-- Not needed with Shake (?)