

Make による SDK ビルドについて

TWL-SDK

2008/06/19

任天堂株式会社企画開発本部 環境制作グループ

0. はじめに

本ドキュメントでは TWL-SDK のコマンドラインビルド環境における Makefile 関連の機能を説明します。

1. make コマンド
2. make ターゲット
 - a) make [build]
 - b) make install
 - c) make full
 - d) make clean
 - e) make clobber

1. make コマンド

make コマンドはアプリケーションのコンパイルなどの手順を自動化するためのツールです。
起動された make コマンドはコンパイル手順が記されたファイル(通常はカレントディレクトリ内の Makefile という名のファイル)を読み込み、その記述にしたがってコンパイラやリンカを順次呼び出します。

TWL-SDK では GNU Make 3.81 を使用します。
Cygwin または Windows のコマンドプロンプトから以下のように入力することで起動します。

```
% make
```

make では以下のようにオプションや変数の設定を行なうことができ、また make の動作を切り替えるためにターゲット名を指定することもできます。

```
% make [オプション] [変数名=設定値] [ターゲット名]
```

2. make ターゲット

本 SDK では Makefile の記述を楽にするために、ゲーム作成においてよく使われる手順についての記述をまとめたファイルを以下のディレクトリに用意しています。

ディレクトリ	\$TwlSDK/build/buildtools/
変数などの定義ファイル	commondefs
コンパイル手順定義ファイル	modulerrules

アプリケーションを作成される方はこれらのファイルをインクルードしてお使いになれます。インクルードの方法についてはサンプル類のコンパイルに使用している Makefile をご参考にしてください。

これらのファイルに定義されているターゲットを説明します。

a) ターゲット build

- コマンド % make
 % make build
- 処理 コンパイルを開始し、最終ターゲットを作成します。
- 手順 1. **SUBDIRS**, **SUBMAKES** に設定されたディレクトリそれぞれに対し make build コマンドを実行します。
 2. 必要なら作業ディレクトリを作ります。
 3. **SRCS** に指定されたファイルをコンパイル/アセンブルしオブジェクトファイルを生成します。
 4. オブジェクトファイルをリンクし **TARGET_BIN** に指定されたファイルを作成します。
 5. 必要なら生成されたファイルを他のディレクトリへインストール(コピー)します。

build は make のターゲット値を省略したときの、デフォルトのターゲット名となっているため、単純に make とだけ入力した場合にも make build 処理が行なわれます。

手順 1 で make コマンドを再起的に呼び出しているため、ツリーの一番親のディレクトリで一度 make コマンドを実行するだけでそれ以降のディレクトリの全てに対して make コマンドが実行されるようになっていきます。

作業ディレクトリは obj/ARM9-TS.HYB/Release などのように現在コンパイルしているプログラムの対象機器 (TS≡IS-TWL-DEBUGGER) や動作モード ([HYB] ハイブリッドモード/[LTD] TWL 専用モード/NITROモード時は無印) やデバッグレベル (Debug/Release/Rom) が付け加えられたものとなります。これらのコンパイルターゲットの対象機器(プラットフォーム)や動作モードやデバッグレベルは、コマンドラインオプションによる変数設定(下記参照)や環境変数の設定値で変更できます。

 % make TWL_DEBUG=TRUE ; デバッグバージョンのターゲットをビルドする。

他の変数名など詳しくは、“[ビルドスイッチ解説ページ](#)” [\\$TwlSDK/docs/SDKRules/Rule-Defines.html](#) の “SDK のビルド時のビルドスイッチ” および “SDK の Makefile 内で設定するビルドスイッチ” の項目をご覧ください。

手順 5 におけるインストール作業の指定はライブラリをコンパイルターゲットに指定し、作成したライブラリファイルを所定の位置にコピーしたりするときに使用します。

生成ファイルとしてバイナリファイルではなくライブラリファイルを指定したい場合は **TARGET_LIB** へそのファイル名を設定します。

b) ターゲット `install`

- コマンド `% make install`
- 処理 `make build` によって作成されたファイルを他のディレクトリへインストール(コピー)します。
- 手順
 1. `SUBDIRS`, `SUBMAKES` に設定されたディレクトリそれぞれに対し `make clean` コマンドを実行します。
 2. `INSTALL_TARGETS` 変数に設定されているファイルが存在した場合、そのファイルを `INSTALL_DIR` で指定されたディレクトリへコピーします。

以下のようにするとコマンドラインからインストール先を指定することができます。

```
% make INSTALL_DIR=/HOME/MYDIR
```

c) ターゲット `full`

- コマンド `% make full`
- 処理 `make build` 各コンパイルターゲット毎全てのバージョンのファイルを生成する。
- 手順
 1. 全てのコンパイルターゲット毎に `make build` を実行します。

d) ターゲット `clean`

- コマンド `% make clean`
- 処理 `make build` によって生成されたファイルを削除します。
- 手順
 1. `SUBDIRS`, `SUBMAKES` に設定されたディレクトリそれぞれに対し `make clean` コマンドを実行します。
 2. オブジェクトファイル用一時ディレクトリとバイナリファイル用一時ディレクトリなどを削除します。
 3. `LDIRT_CLEAN` に指定されたファイルを削除します。

`make install` によってコピーされたファイルは削除されません。インストールされたファイルを消すには `make clobber` をご使用ください。

e) ターゲット `clobber`

- コマンド `% make clobber`
- 処理 `make build` によって生成されたファイルを完全に削除します。

- 手順
 - 1. `SUBDIRS`, `SUBMAKES` に設定されたディレクトリそれぞれに対し `make clobber` コマンドを実行します。
 - 2. オブジェクトファイル用一時ディレクトリとバイナリファイル用一時ディレクトリを削除します。
 - 3. `make build` / `make install` でインストールされたファイルを削除します。
 - 4. `LDIRT_CLEAN` に指定されたファイルと `LDIRT_CLOBBER` で指定されたファイルを削除します。
 - 5. プリコンパイルヘッダを削除します。

`make clean` 操作に加えて、インストールされたファイルやプリコンパイルヘッダも削除します。