SDK Builds Using Make

TWL-SDK

Version 2008/06/19

The content of this document is highly confidential and should be handled accordingly.

TWL-06-0002-001-A
Released: October 20, 2008

Table of Contents

Revision History

Version	Revision Date	Description
	2008/06/19	Initial version.

1 Introduction

This document describes the features of the `makefile` in the TWL-SDK build environment.

1. `make` command
2. `make` target
    1. `make` [build]
    2. `make install`
    3. `make full`
    4. `make clean`
    5. `make clobber`

2 make Command

The `make` command is a tool that automates procedures (such as compiling applications).

A `make` command, when executed, reads a file described in the compiler procedure (usually the file called `Makefile` in the current directory) and then calls the compiler and linker following the descriptions in that file.

TWL-SDK uses GNU `make` 3.81.

Execute `make` by entering the command as shown below from the Cygwin or Windows command prompt.

```
% make
```

You can set the following options and variables with the `make` command as well as specify target names to change the operation of the `make` command.

```
% make [option] [variable name = set value] [target name]
```

3 make Target

In order to simplify the descriptions of the `makefile`, this SDK provides files, in the directory shown below, that combine descriptions for procedures often used when producing games.

Directory	`$TwlSDK/build/buildtools/`
Definition (such as variables) file	`commondefs`
Compiler procedure definition file	`modulerules`

Developers creating applications should include and use these files. Refer to the `makefile` used to compile a sample class for details on how to include the files.

Targets defined in these files are described below.

3.1 Target Build

- Command
```
% make
% make build
```
- Process Starts the compiler and creates a final target.
- Procedure
  1. Execute the `make` build command for each directory set in `SUBDIRS` and `SUBMAKES`.
  2. Create the required work directory.
  3. Compile/assemble the files specified in `SRCS` and then create an object file.
  4. Link the object file to create the file specified in `TARGET_BIN`.
  5. If necessary, install (copy) the files that were created.

Because `build` will become the default target name when you omit the target value of `make`, you can also execute the `make` build process by only typing `make`.

Because the `make` command is called more than once in step 1 of the procedure, you only need to execute the `make` command one time for the first parent directory in the tree to execute the command for all subsequent directories after the parent directory.

The work directory is a directory that will have *target devices* (`TS≒IS-TWL-DEBUGGER`), *operational modes* (“HYB” for hybrid mode, “LTD” for TWL-exclusive mode, and nothing for NITRO mode), and *debug levels* (Debug/Release/Rom) of programs currently being compiled, such as obj/ARM9-TS.HYB/Release, added to it. You can change the target devices (platforms), operational modes, and debug levels of these compiler targets by variable settings (described below) and setting values of environment variables as command line options.

```
% make TWL_DEBUG=TRUE    : Builds target of debug version.
```

For details on other variable names and other topics, see “Build Switches When Building the SDK” and “Build Switches Set Within a Makefile of the SDK” on the build switch description page `$TwlSDK/docs/SDKRules/Rule-Defines.html`.

The specification of items for the installation procedure in step 5 is done when specifying libraries in the compiler target and copying the library files that are created to a designated location.

Set the file name in `TARGET_LIB` when you want to specify a library file but not a binary file as a created file.

3.2 Target Install

- Command `% make install`
- Process Installs (copies) files created by make build to other directories.
- Procedure
    1. Execute the `make build` command for each directory set in `SUBDIRS` and `SUBMAKES`.
    2. Copy files to the directory specified by `INSTALL_DIR` when there are files specified in the `INSTALL_TARGETS` variable.

You can specify the installation destination from the command line as follows:

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

3.3 Target Full

- Command `% make full`
- Process Creates files for all versions of `make build` compiler targets.
- Procedure Execute `make build` for all compiler targets.

3.4 Target Clean

- Command `% make clean`
- Process Deletes files created by `make build`.
- Procedure
    1. Execute the `make clean` command for each directory set in `SUBDIRS` and `SUBMAKES`.
    2. Delete temporary directories for object files and temporary directories for binary files.
    3. Delete files specified in `LDIRT_CLEAN`.

Files copied by `make install` are not deleted. Use `make clobber` to delete installed files.

3.5 Target Clobber

- Command `% make clobber`
- Process Deletes files created by `make build`.
- Procedure
    1. Execute the `make clobber` command for each directory set in `SUBDIRS` and `SUBMAKES`.
    2. Delete temporary directories for object files and temporary directories for binary files.
    3. Delete files installed by `make build / make install`.
    4. Delete files created by `LDIRT_CLEAN` and files created by `LDIRT_CLOBBER`.
    5. Delete the precompiled headers.

Deletes installed files and precompiled headers in addition to the `make clean` operation.