cMent 2.2 for OS/2

An automatic project builder

User's Reference

The Masterpiece Software Company, Ltd.
One Pronghorn Park, Parks, AZ 86018
72650.1533@compuserve.com

Introduction

cMent is an automatic software project builder for the OS/2 environment. Written with VxRexx, the program is executed from the pop-up menu of a project folder containing all of the files specific to a particular software project. Its main dialogue allows you to select either a progressive rebuild; a complete rebuild with debug turned on; or a final, optimized build for production purposes. Then, based on time stamps and standard file extensions, the main program calls a number of short *.cmd* files to invoke your own preprocessors, compilers, assemblers, linkers, etc. Finally, it will automatically bind resources and create *.lib* files from *.dll* targets.

All of this is done without the need for complex make files. Such few "rules" as are needed are provided by environment variables whose values are contained in a single, global *cMent.set* file. In most cases, you will modify this file once with the paths, names and options for your tools. Occasionally, you may need to override this with a local *cMent.set* file in the project folder (for example, to specify a new set of compiler options). Very rarely, you may need to modify or add to the *.cmd* files in order to support a tool or file extension we haven't encountered yet.

The output from your tools is redirected to *Project.log*, a simple text file created in your project folder. You can edit or print this file to examine the results from the entire build. Alternatively, you can specify the *cWeed* utility to your favorite editor as a (pseudo) compiler. Then, when you "compile" from the editor, *cWeed* will extract the messages for your source file from the *Project.log* and pass them to your compiler for automatic positioning. Now just add program objects for debugging and/or profiling to the project folder and you have a complete integrated development environment (IDE) based on the *Workplace Shell*. Proprietary IDE's like *Workframe* are no longer needed, so you don't have to learn them.

The first three lines set the basic options for the linker, the Cset++ compiler and the SOM emitter. Modify or add to these as needed for your own compilers, linkers and preprocessors.

The next five lines specify the paths to the toolkit, the compiler, and a temporary files directory. The following *path*, *include* and *lib* variables include the earlier variables and should be more or less self explanatory to a programmer. Modify all of these to meet your own requirements.

The next four lines set the main *path*, *include* and *lib* strings. You shouldn't need to modify these, except to set or remove *d:\masm*.

The *source* and *target* variables control the entire preprocessing and compilation procedure. *Source* lists all the extensions which are to be recognized by *cMent* as true source files. *Target* lists the names of the *.cmd* files which are used to translate source files into their targets (for example, *ccc2exe.cmd* translates a *.ccc* source file directly into an *.exe*). If you wish to compile to *.obj* instead of the more effective *.w* intermediate code files, then you will need to replace *c2w* with *c2obj* and *cpp2w* with *cpp2obj*. If you use any other source extensions (e.g. *.cbl* or *.pli*) then you will also need to add to these lists as described in *The .cmd Files*, below.

The *csc, c, cpp, ccc, cxx* and *rc* variables are used within the *.cmd* files to build the actual command used for translation. The source file name is inserted after the /options, and before any additional names like *setargv.obj* (which is used to force expansion of wild card file names before starting your *main* function). You will need to modify these variables only if you are using a different compiler or if you wish to change the compiler options.

Similarly, the *exe, dll* and *drv* variables are used within the main program to build the command lines for creating the main project target. You will need to modify them only if you are using a dif-

Optionally, you may also include one or more source files with special extensions which *cMent* compiles directly into *.exe* files. This allows you to maintain small test programs in the same project folder as the *.dll* they are designed to test. The default setup uses *.ccc* and *.cxx* files for this purpose. You may also wish to include files with extensions which are not recognized or processed by *cMent* (e.g. *.inf* or *.cmd*) and other *Workplace Shell* items such as program objects or nested folders.

Pretty much anything else goes within the project folder, but there are some common sense restrictions on file names. For example, source files are always compiled to target *.obj* (or *.w* or *.res*) files with the same name as the source file. Include files can have any name and even trigger an automatic re-compile of any source files in which they are included (as long as they are in the same folder).

The most important rule is that the source file for the first object to be linked into the target must have the same name as the target itself. When a project is built for the first time, there is no target *.exe* or *.dll* and *cMent* may decide to build one with an incorrect name (from another source file). In this case, simply rename the target to its correct name and all future builds will be correct.

In principle, any folder can be converted to a project folder simply by adding *cMent.exe* to its pop-up menu. It is also possible to set up a program object for *cMent* and then to build *any* folder simply by dropping it on *cMent*. A project can also be built by typing the command *cMent folder-path* in any OS/2 command session. However, the easiest way for most users is to set up a template from which new projects can be dragged.

This author likes to work from a primary *work-area* folder, called *Opus*, which contains all active projects together with templates for new projects and for source files of various types. That way, a double click on *Opus* gets me right back to the project I last