OCI MPC

Makefile, **P**roject and workspace **C**reator

Kevin Heifner
heifner@ociweb.com

OBJECT COMPUTING, INC.

MPC Overview

- Makefile, Project and workspace Creator generates tool-specific build files from a single (simple) project file
- Portable, extensible, flexible tool which enables you to define once, and build many!
- Not part of Boost, developed by OCI
  - initial design/implementation by Chad Elliott and Justin Michel
  - initially targeted to ACE+TAO
  - download releases: http://www.ociweb.com/products/mpc
  - checkout latest: svn://svn.dre.vanderbilt.edu/DOC/MPC/trunk
  - free for commercial and non-commercial use
  - 80 page pdf documentation
    - http://www.ociweb.com/products/mpc

MPC Overview

- Features
  - leverages native build tool chains on various platforms in a portable manner
  - generate various build tool files instead of manually creating them
  - allows developers to choose favorite build tool
  - inheritance from common base projects (reuse mechanism)
  - default values for many aspects of a project
  - simple syntax for ease of use and maintenance
  - extensibility for adding custom features or support for new build tools
  - perl based for rapid development, portability, and ease of automation

MPC Overview

- Instead of creating and maintaining separate build tool files, *generate* them
- MPC types and current output build files supported (-type)

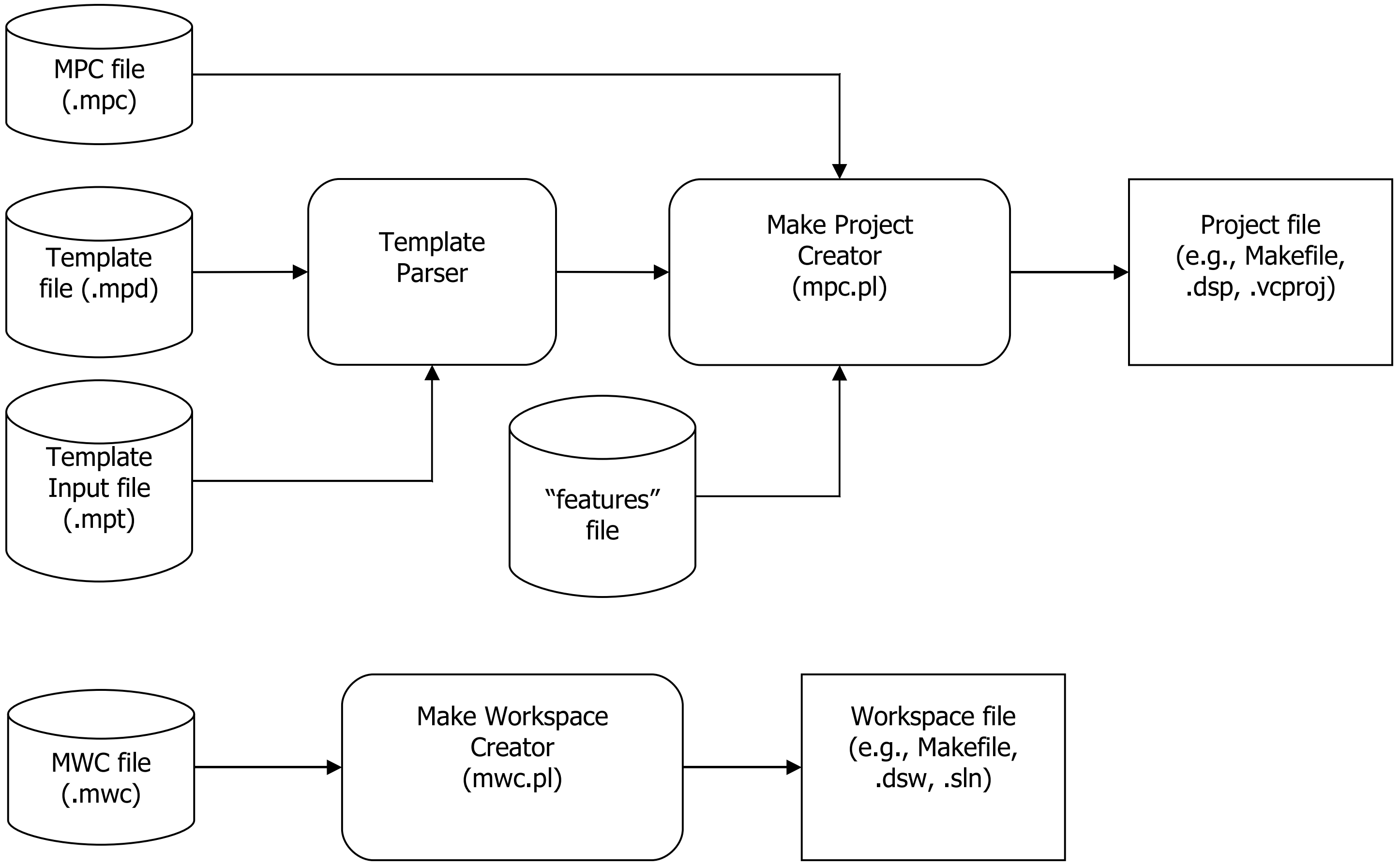
– vc6	Microsoft Visual C++ 6
– vc7, vc71	Microsoft Visual C++ 7.0, 7.1
– vc8	Microsoft Visual C++ 8 (2005)
– vc9	Microsoft Visual C++ 9 (2008)
– nmake	Microsoft NMake
– make	Generic Make
– automake	GNU Automake
– gnuace	GNU Make with ACE/TAO extensions
– bmake	Borland Make
– em3	eMbedded C++ 3 & 4
– ghs	Green Hills C++ Builder
– cc	Code Composer
– bds4	Borland Developer Studio 4 (incomplete)
– bcb2007	Borland C++ Builder 2007 (incomplete)
– sle	Visual SlickEdit (incomplete)
– wb26	Wind River Workbench 2.6 (incomplete)

(make based project types can be used with Eclipse via **`-for_eclipse`** option)

MPC Input Files

- Project Files (mpc)
  - represents build targets such as libraries and executables
  - contains such things as: include paths, library paths, source files, inter-project dependencies, etc.
- Workspace Files (mwc)
  - represents workspaces, e.g. *.sln, high level Makefile
    - made up of one or more project files (mpc)
  - contains list of mpc files, directories, or other mwc files
- Base Project Files (mpb)
  - project definitions can use inheritance (single or multiple)
  - all information is available to base projects
    - describe common project include paths, library paths, dependencies, etc
  - base projects for all Boost libraries are included as part of MPC
- Base Workspace Files (mwb)
  - same idea as base project files

MPC


MPC file (.mpc)
Template file (.mpd)
Template Input file (.mpt)
Template Parser
Make Project Creator (mpc.pl)
"features" file
Project file (e.g., Makefile, .dsp, .vcproj)
MWC file (.mwc)
Make Workspace Creator (mwc.pl)
Workspace file (e.g., Makefile, .dsw, .sln)

MPC Default Values

- Default values can be filled in for almost any element of a project, e.g.,

- project name	- header files	- documentation files
- target type/name	- inline files	- resource files
- source files	- IDL files	- precompiled headers

- Defaults can greatly simplify creation of new projects
- Specific project definitions can override the defaults

Example MPC Default Values

- Project name
  - defaults to name of mpc file minus the .mpc extension or name of current directory if no mpc file exists
  - override with **`project(my_project_name)`**
- Target type/name
  - defaults to exe with name of source file containing “**`main`**” or shared lib with name of mpc file or directory if no “**`main`**”
    - searches for “**`main`**” are case-insensitive to support macros such as **`ACE_TMAIN`** or **`BOOST_AUTO_TEST_MAIN`**
  - override with **`exename`**, **`sharedname`**, or **`staticname`**
    - **`staticname`** same as **`sharedname`** if not overridden

Example MPC Default Values

- Source files
  - defaults to all .cpp, .cxx, .cc, .c, .C files in current directory
  - override with **`Source_Files`**
- Header files
  - defaults to all .h, .hxx, .hh files in current directory
  - also interacts with **`Source_Files`** and considers IDL-compiler-generated header files
  - override with **`Header_Files`**
- IDL files
  - defaults to all .idl files in current directory
  - generated files automatically added to **`Source_Files`**, **`Header_Files`**, **`Inline_Files`**
  - override with **`IDL_Files`**

Custom Build Rules

- MPC allows the definition of custom file types and rules to process them

```
project {
  Define_Custom(FOO) {
    automatic = 0
    command = $(FOODIR)/bin/fooparser
    commandflags = -I$(FOODIR)/include
    output_option = -o
    inputext = .foo
    source_outputext = .bar
  }
  FOO_Files {
    Hello.foo
  }
}
```

Result:

```
  $(FOODIR)/bin/fooparser -I$(FOODIR)/include -o Hello.bar Hello.foo
```

Guidelines for Using MPC

- One directory per build target
  - leads to smaller, simpler project files due to defaults
    - In some very simple cases, no project file is even needed
- Follow MPC's file *naming conventions* to leverage defaults
  - e.g., .h, .cpp, .inl, _T.h, _T.cpp
- *Don't repeat yourself* – use inheritance!
  - factor common project description elements into base projects
  - leads to much smaller, simpler project files
  - future changes are localized in a few base projects
  - use existing base projects in **`$MPC_ROOT/config`**

Experiences Using MPC

- MPC is currently being used in OCI's, PrismTech, and the DOC group's nightly builds of ACE+TAO+CIAO
- MPC is an integral part of the auto-build processes
- MPC uses a powerful workspace/project metaphor that eases maintenance, but has a slight learning curve
- No automatic conversion from existing build-tool files (e.g., Makefiles or bjam) to MPC files
  - currently, users must write MPC files by hand, but they are usually very short and simple

.mwc Example

```
// file: BoostExamples.mwc
// Directory: examples
// all sub-directories
// added by default
workspace {
}
```

```
// file: MyProjects.mwc
// Directory: examples
// build a subset of projects
workspace {
  container
  conversion
  lambda
}
```

```
Directory Structure

../examples/BoostExamples.mwc
         ../container/container.mpc
         ../conversion/conversion.mpc
         ../lambda/lambda/mpc
         ../regex/regex.mpc
         ../smartptr/smartptr.mpc
         ../test/test.mpc
         ../test/unit_test_lib/ut.mpc
         ../thread/thread.mpc
```

```
> mwc.pl -features boost=1 -type vc8 MyProjects.mwc
Generating vc8 output using MyProjects.mwc
Generation Time: 1s
```

.mpc Example

```
All source files are included by default,
no need to list them explicitly
```

```
// file: thread.mpc
// Directory: examples/thread
project : boost_unit_test_framework, boost_thread {
  // specify executable name, optional
  exename = thread
  // indicate that unit_test should be run after the build
  postbuild = <%exename%> --report_level=no
}
```

```
Directory Structure

../examples/BoostExamples.mwc
          ../thread/thread.mpc
```

```
> mwc.pl -features boost=1 -type vc8
Generating vc8 output using default input
Generation Time: 1s
```

MPC Boost .mpb files

- `boost_base.mpb`
- `boost_date_time.mpb`
- `boost_filesystem.mpb`
- `boost_iostreams.mpb`
- `boost_prg_exec_monitor.mpb`
- `boost_program_options.mpb`
- `boost_python.mpb`
- `boost_regex.mpb`
- `boost_serialization.mpb`
- `boost_signals.mpb`
- `boost_test_exec_monitor.mpb`
- `boost_thread.mpb`
- `boost_unit_test_framework.mpb`
- `boost_wave.mpb`

`Used in the previous example`

More Information

- Main MPC page
  - http://www.ociweb.com/products/mpc
- Scripts to build boost with MPC (see README)
  - http://www.ociweb.com/products/boost
- 80 page pdf documentation
  - http://downloads.ociweb.com/MPC/MakeProjectCreator.pdf
- Description of features with examples
  - ../MPC/docs/README
- Description of all command line arguments
  - ../MPC/docs/USAGE
- Description of tool related options
  - ../MPC/docs/templates/*.txt
- Commercial support
  - sales@ociweb.com