

WHAT IS MPC?

- Utility that promote portability of projects across different build tools & environments using generic configuration files
- Preserves the essence of a project & workspace
 - Include paths
 - Import libraries and their paths
 - Name of the target binary
- Extensible to support many different build tools and environments
 - Visual Studio, GNU Makefile, Wix, Borland
- Download
 - <https://github.com/DOCGroup/MPC>

PROJECT DEFINITION

- Captures information about how to build a project
 - Name of the target binary, import libraries and their path, source files, etc.
- Captures tool and environment specific project details
 - Import libraries with different names on Windows vs. non-Windows
 - Header files in different locations for different platforms
 - Debug vs. release version of build
- Supports three types of projects
 - executable, shared library (.dll, .so, .dylib), static library (.lib, .a)
- Has a .mpc file extension

PROJECT DEFINITION

```
project (ProjectName) {
    exename | sharedname | staticname = binaryName

    libs +=          // space delimited list of libraries
    libpaths +=      // space delimited list of directories
    includes +=      // space delimited list of directories

    Source_Files {
        // list of source files (1 per line)
    }

    Inline_Files {
        // list of inline files (1 per line)
    }

    Header_Files {
        // list of header files (1 per line)
    }

    // other groups: Documentation_Files,
    // Resource_Files
}
```

Project settings, also supports = and -= operator

Different groupings of the project files, Source_Files is the most important grouping

EXAMPLE PROJECT FILES

```
// Example 1
project (Array) {
    exename = array-jhh

    Source_Files {
        Array.cpp
        driver.cpp
    }

    Header_Files {
        Array.h
    }

    Inline_Files {
        Array.inl
    }
}
```

```
// Example 2 (equivalent to Example 1)
project (Array) {
    exename = array-jhh

    Source_Files {
        Array.cpp
        driver.cpp
    }
}
```

```
// Example 3
project (SQLiteExample) {
    exename = sqlite-example

    // use environment variables
    includes += $(SQLITE_ROOT)/include
    libpaths += $(SQLITE_ROOT)/libs

    libs += sqlite3

    Source_Files {
        example.cpp
    }
}
```

WORKSPACE DEFINITION

- Used to determine what projects are included in a given workspace

```
workspace (WorkspaceName) {  
    // list of workspaces or  
    // directories (1 per line)  
}
```

Example workspaces

```
workspace (SingleProjectWorkspace) {  
    // include only this project  
    Array.mpc  
  
    // include all projects in directory  
    ./stack  
}  
  
workspace (AllFilesWorkspace) {  
    // recursively include all .mpc files  
}
```

RUNNING MPC

- Must be run from a command prompt
- `mwc.pl -type build_type [workspace]`
 - `build_type`: `vc12`, `make`, `nmake`, etc.
- Workspace argument is optional, if blank then all projects in the current directory (including subdirectories)