## NAME

maureen − Mirror-library Auto Reflection/Registering Engine

## SYNOPSIS

**maureen** [ *options* ]

## DESCRIPTION

*MAuReEn* is a tool generating Mirror reflection library's registering code from a set of C++ input files and a set of input/output pattern pairs.

## OPTIONS

**−h --help**

Prints a help screen.

**−t --toolset <toolset-name>**

Use the toolset specified by **toolset-name**.

**−i --input <input-pattern>**

Input pattern, which is a part of a input/output pattern pair. This means that any **--input** option must be directly followed by an **--output** option.
**input-pattern**: 'root|path|file|suffix|kind|scope|name'
see, **INPUT PATTERNS** and **EXAMPLES** below.

**−o --output <output-pattern>**

Output pattern, which is a part of a input/output pattern pair. This means that any **--output** option must be directly preceded by an **--input** option.
**output-pattern**: 'root|path|file|suffix'
see, **OUTPUT PATTERNS** and **EXAMPLES** below.

**−m --marefile <filepath>**

Include statements from the specified **MAREfile**.

**−I --include-path <direpath>**

Add include file search path. This option applies to input patterns that have the root part specified as **%**.

**−D --define <preprocessor-symbol-definition>**

Define a preprocessor symbol to be used when preprocessing the input sources.

**−p --macro-prefix <PREFIX>**

Prepend PREFIX to all registering macros in output.

**−b --boost-macro-prefix**

Prepend BOOST_ prefix to all registering macros in output.

## INPUT PATTERNS

Input patterns specify which C++ input source or header files and which constructs from these files should be processed into the resulting registering code. Every input pattern must have a matching output pattern (see below). The input patters can be specified on the command line when invoking *maureen*, or in one of the **MAREfiles** specified with the **-m** command line options. An input pattern has the following format:

'root|path|file|suffix|kind|scope|name'

where the | (pipe) symbol acts as the separator and the individual components have the following meaning:

**root** The root path to the sources to be processed. The special value % means that the input files will be searched at the paths specified by the **-I** options.
There is an important concept related to the root component of the pattern: the **base directory**. In case the input pattern was specified on the command line then the base directory is the current working directory from which *maureen* has been invoked. If the pattern was specified in a **MAREfile**, then the base directory is the directory where the **MAREfile** is located. An empty value specifies that the **base directory** is to be used as root. **Examples:** /usr/include, /home/login/include, C:\Devel\MyProject\headers, %.

**path** The path to the sources relative to the root. The Special value % means that any path applies including an empty path. This way any file in the root directory or its subdirectories can be specified. **Examples:** src, test, %.

**file** The file name pattern without the suffix, The special value **%** acts as a wildcard for a string of arbitrary length. **Examples:** foo, foo%, %.

**suffix** The file suffix (extension) pattern. The special value **%** acts as a wildcard for a string of arbitrary length. **Examples:** h, hpp, h%, %.

**kind** The kind of programming construct to be processed. The special value **%** can be used to process everything. **Examples:** namespace, class, enum, %.

**scope** The pattern for the scopes to be processed. The special value **%** represents any named scope (i.e. not the global scope), an empty value represents the global scope only. **Examples:** std, boost, foo%, %.

**name** The pattern specifying the names of the language constructs (specified by the **kind** component) to be processed. The special value **%** can be used to process everything. **Examples:** tm, my_class, my_project, %.

**Examples or full input patterns:**

1: '%|%|%|%||%|%'

Process files of any type in the directories specified by the **-I** options and their subdirectories, register everything (namespaces, classes, enums, etc.), in the global scope.

2: '%|%|%|%|%|%|%'

Process files of any type in the directories specified by the **-I** options and their subdirectories, register everything (namespaces, classes, enums, etc.), in every namespace (excluding the global scope).

3: '%|%|%|hpp|%|%|%'

Process **hpp** files in the directories specified by the **-I** options and their subdirectories, register everything (namespaces, classes, enums, etc.), in any named namespace.

4: '|%|%|hpp|%|%|%'

Process **hpp** files in the base directory and its subdirectories, register everything, in any named namespace.

5: '||%|hpp|%||%'

Process **hpp** files in the base directory but not its subdirectories, register everything, in the global scope.

6: '||%|hpp|%|%|%'

Process **hpp** files in the base directory but not its subdirectories, register everything, in any named namespace.

7: '||%|hpp|namespace||%'

Process **hpp** files in the base directory but not its subdirectories, register only **namespace**s from the global scope.

8: '||%|hpp|namespace|%|%'

Process **hpp** files in the base directory but not its subdirectories, register only nested **namespace**s.

9: '|%|%|inc|%||%'

Process **inc** files in the base directory and its subdirectories, register everything, from the global scope, but not from any other namespace.

10: '|%|%|hpp|%|test|%'

Process **hpp** files in the base directory and its subdirectories, register everything, from the **test** namespace, but not its nested namespaces.

11: '|%|%|hpp|%|foo%|%'

Process **hpp** files in the base directory and its subdirectories, register everything, from the **foo** namespace and its nested namespaces.

12: '|%|%|hpp|class|%|foo'

Process **hpp** files in the base directory and its subdirectories, register only classes named **foo** from any named namespace.

13: '%||time|h|%|std|tm'

Process the **time.h** file in the input search directories but not in their subdirectories, register anything named **tm** from the **std** namespace.

14: '%||time||struct|std|tm'

Process the **time** file in the input search directories but not in their subdirectories, register only **struct**s named **tm** from the **std** namespace.

## OUTPUT PATTERNS

Output patterns specify the names and locations of the output files containing the resulting Mirror's registering code. Every output pattern is linked to an input pattern The output patters can be specified on the command line when invoking *maureen*, or in one of the **MAREfiles** specified with the **-m** command line options.

As with the input pattern the output pattern has several components which can contain literal strings (names, paths), but there are also several variables which can reference the values of the components from the related input pattern. The variables are geneally expanded by the $(*variable-name*) expression although several of the variable-expansion expressions can also have an extended syntax:

**root** References the root from the input pattern

**path** References the path of the input file

**file** References the basename of the input file

**suffix** References the suffix of the input file

**kind** References the kind of the processed (registered) language construct (namespace, class, enum, etc.)

**scope** References the scope of the processed (registered) language construct. The separating **::** (double-colon) can be replaced in the expansion by another separator with this expression syntax $(*scope:*<**separating character**>), for example for a type **foo::bar::baz::foobar** the expression $(*scope:/*) expands into **'foo/bar/baz'** and $(*scope:-*) expands into **'foo-bar-baz'**.

**name** References the name of the processed (registered) language construct.

An output pattern has the following format:

'root|path|file|suffix'

where the **|** (pipe) symbol acts as the separator and the individual components have the following meaning:

**root** The directory path where the output files should be placed. As with the input pattern there is the concept of the **base directory**: In case the output pattern was specified on the command line then the base directory is the current working directory from which *maureen* has been invoked. If the pattern was specified in a **MAREfile**, then the base directory is the directory where the **MAREfile** is located. The path specified as root is relative to the **base directory**, unless the value is an absolute path. **Examples:** /opt/my_project/meta-data, ./meta-data, ../output/meta-data, $(root).

**path** The path to the output files relative to the root. If an empty value is specified, then the output files are placed directly into the root directory. **Examples:** foo, test, $(path).

**name** The basename of the output file. **Examples:** $(name), $(scope:-)-$(name), meta-$(name), meta-data.

**suffix** The suffix of the output file. **Examples:** hpp, metahpp, meta$(kind).

**Examples or full output patterns:**

1: '../output/meta-data|$(path)|$(file)|$(suffix)'

Place the output files to the **../output/meta-data** directory at the same paths that the input files had, relative to the root specified in the input pattern.

2: '../output/meta-data|$(scope:/)|$(name)|hpp'

Place the output files to the **../output/meta-data** directory, into subdirectories for every namespace, recursively for nested namespaces, the files named by the individual registered language constructs, with the **hpp** extension.

3: 'meta-data||$(scope:-)-$(name)|hpp'

Place the output files directly into the **meta-data** directory, the files named by the full-nested-names of the individual registered language constructs (the double-colons replaced by -), with the **hpp** extension.

4: '|$(scope:_)|$(name)|meta-$(kind)'

Place the output to the base directory, into subdirectories for every namespace (the double-colons in the nested namespace named replaced by an underscore), the files named by the basic names of the individual registered constructs, with extensions like **meta-struct** for struct's, **meta-class** for classes, etc.

## MAREFILES

Besides specifying the input-to-output processing options on the command line, they can be also be specified inside special files, called **MAREfile**s (from Mirror Auto Reflection Engine files). When invoking *maureen*, one or multiple MAREfiles can be specified explicitly with the **-m** option, or if there is a file with the name **MAREfile** in the current working directory from which *maureen* was invoked, then it is used implicitly (unless there are any marefiles specified explicitly; in such case the **MAREfile** is not used).

The marefiles can contain three types of lines; comments, commands and input/output pattern pairs.

Comment lines must begin with the **#** sign and are ignored by the processor. Comments cannot be used on the same line with commands or input/output pattern pairs.

**Examples:**
# This is a comment
# ERROR - invalid comment
@ 'define|TEST' # ERROR - invalid comment

Command lines must begin with the **@** (at) symbol and have the following format (note that the command with the potential parameters is enclosed in apostrophes):

@ '<*command*>[|*parameter1*[|*parameter2*[|...]]]'

**Examples:**
@ 'define|_DEBUG'
@ 'define|__BEGIN_STD_NAMESPACE|namespace std {'
@ 'define|__END_STD_NAMESPACE|}'

The input/output pattern pair simply contains an input and an output pattern, separated by a space on a single line (again, note that both patterns are enclosed in apostrophes):

'<*input-pattern*>' '<*output-pattern*>'

The format of the patterns is the same as specified in the **INPUT PATTERNS** and **OUTPUT PATTERNS** sections above.

**Examples:**
'|%|%|hpp|namespace||%' '../out/meta-data||$(name)|$(suffix)'
'|%|%|hpp|%|%|%' '../out/meta-data||$(scope:_)|$(suffix)'

## EXAMPLES

This section shows several simple examples including the sources, MAREfiles and *maureen* command-line invocations.

Let's suppose there is the following directory tree:

```
+-+-[root_dir/]
```

```
  |
  +-+-[include/]
  | |
  | +---[fubar.hpp]
  | +---[person.hpp]
  | `---[tetrahedron.hpp]
  |
  +-+-[maureen/]
  | |
  | +---[MAREfile.1]
  | +---[MAREfile.2]
  | `---[MAREfile.3]
  |
  `---[MAREfile]
```

where the header files contain the following definitions:

```
// --- fubar.hpp ---

namespace foo {
namespace bar {
namespace baz {

struct qux { };

} // namespace baz
} // namespace bar
} // namespace foo

// --- person.hpp ---

namespace test {

struct person_pod
{
      std::string given_name;
      std::string middle_name;
      std::string family_name;
      std::tm birth_date;
      double weight;
      double height;
};

} // namespace test

// --- tetrahedron.hpp ---

namespace test {

struct vector
{
      double x,y,z;

      vector(double _x, double _y, double _z);
```

```
        vector(double _w);
        vector(void);

        double length(void) const;
        friend vector operator + (const vector& a, const vector& b);
        friend vector operator - (const vector& a, const vector& b);
        // cross produnt
        friend vector operator % (const vector& a, const vector& b);
        // dot product
        friend double operator * (const vector& a, const vector& b);
};

struct triangle
{
        vector a, b, c;

        triangle(const vector& _a, const vector& _b, const vector& _c);
        triangle(void);

        double area(void) const;
};

struct tetrahedron
{
        triangle base;
        vector apex;

        tetrahedron(const triangle& _base, const vector& _apex);

        tetrahedron(
                const vector& a,
                const vector& b,
                const vector& c,
                const vector& d
        );

        const vector& a(void);
        const vector& b(void);
        const vector& c(void);
        const vector& d(void);

        void reset_apex(const vector& _apex);
};

} // namespace test
```

The MAREfiles have the following contents:

```
# Marefile
# Process everything in the global scope
'|%|%|hpp|%||%' 'out/metadata||$(name)|$(suffix)'
# Process everything in the nested namespaces
'|%|%|hpp|%|%|%' 'out/metadata|$(scope:/)|$(name)|$(suffix)'
```

```
# maureen/MAREfile.1
# Process everything in the global scope
'|%|%|hpp|%||%' '../out/metadata||$(name)|$(suffix)'
# Process everything in the nested namespaces
'|%|%|hpp|%|%|%' '../out/metadata|$(scope:/)|$(name)|$(suffix)'

# maureen/MAREfile.2
# process namespaces in the global scope
'|%|%|hpp|namespace||%' '../out/metadata||$(name)|$(suffix)'
# process everything in the namespaces
'|%|%|hpp|%|%|%' '../out/metadata||$(scope:_)|$(suffix)'
```

Invoking *maureen* from the **root_dir** without any arguments, which causes it to use the **MAREfile** in the root dir, produces the following directory tree:

```
+-+-[root_dir/]
  |
  +-+-[out/]
  | |
  | '-+-[metadata/]
  |   |
  |   +-+-[foo/]
  |   | |
  |   | +-+-[bar/]
  |   | | |
  |   | | +-+-[baz/]
  |   | | | |
  |   | | | '---[qux.hpp]
  |   | | |
  |   | | '---[baz.hpp]
  |   | |
  |   | '---[bar.hpp]
  |   |
  |   +-+-[test/]
  |   | |
  |   | +---[person_pod.hpp]
  |   | +---[tetrahedron.hpp]
  |   | +---[triangle.hpp]
  |   | '---[vector.hpp]
  |   |
  |   +---[foo.hpp]
  |   '---[test.hpp]
  |
  '---[...]
```

The directory **out/metadata** is the output root. The files **foo.hpp** and **test.hpp** contain code registering the namespace **foo** and **test** respectivelly, with Mirror. The directory **test/** contains registering code for types defined in the **test** namespace. The files inside are named by the individual classes declared therein and contain registering code for the respective classes.

The directory **foo** and its nested subdirectories **bar** and **baz** contain headers for registering the constructs defined in the namespaces **foo**, **foo::bar** and **foo::bar::baz**, respectivelly. The file **qux.hpp** for example

contains registering code for **struct foo::bar::baz::qux**, declared in the **root_dir/include/fubar.hpp** input header file.

The same output as above, could be achieved by the following invocations of *maureen*:

```
user@host:.../root_dir$ maureen \
    --input  '|%|%|hpp|%||%' \
    --output 'out/metadata||$(name)|$(suffix)' \
    --input  '|%|%|hpp|%|%|%' \
    --output 'out/metadata|$(scope:/)|$(name)|$(suffix)'
user@host:.../root_dir$ maureen --marefile maureen/MAREfile.1
```

Invoking *maureen* with the **-m maureen/MAREfile.2** option produces the following output tree:

```
+-+-[root_dir/]
  |
  +-+-[out/]
  | |
  | '-+-[metadata/]
  |   |
  |   +---[foo.hpp]
  |   +---[foo_bar.hpp]
  |   +---[foo_bar_baz.hpp]
  |   '---[test.hpp]
  |
  '---[...]
```

Again, the directory **out/metadata** is the output root. The files **foo.hpp**, **foo_bar.hpp**, **foo_bar_baz.hpp** and **test.hpp** contain code registering everything declared in the **foo**, **foo::bar**, **foo::bar::baz** and **test** namespaces, respectivelly.

## AUTHOR

Matus Chochlik, Matus.Chochlik@fri.uniza.sk

## COPYRIGHT



## AUTHOR

Matus Chochlik, Matus.Chochlik@fri.uniza.sk

## COPYRIGHT

Copyright (c) 2008, 2009, 2010, 2011 Matus Chochlik

Permission is granted to copy, distribute and/or modify this document under the terms of the Boost Software License, Version 1.0. (See a copy at http://www.boost.org/LICENSE_1_0.txt)