NAME

intro — Introduction to MOBY command-line tools.

DESCRIPTION

This chapter describes user commands provided by the MOBY system.

SEEALSO

NAME

charon — an interoperability tool that allows MOBY programs to access C data and call C functions.

SYNOPSIS

charon file . . .

DESCRIPTION

The *charon* command is used to create glue code that enables MOBY programs to access C data structures and invoke C functions. The tool takes a C header file and produces a MOBY MBX file, which is a textual representation of the MOBY compiler's internal representation. The *gen-mbi* tool converts MBX files into MBI files, which are the binary representation of MOBY library code.

In more detail, if *charon* is invoked on C header file c.h, it will produce a file c.mbx. *gen-mbi* can then be invoked to produce c.mbi. A MOBY program m.mby may access the data and use the functions described by c.h by passing the associated C object file c.o to *mobyc* and by including c.mbi in the filemap given to *mobyc*.

OPTIONS

`--cpp`

Run the C pre-processor on the argument files before generating MBX code.

SEEALSO

mobyc, *gen-mbi*

NAME

gen-mbi — tool to generate MBI files from MBX files.

SYNOPSIS

gen-mbi file . . .

DESCRIPTION

The *gen-mbi* command is used to generate MBI files, which have a binary representation, from MBX files, which are ASCII. This tool is used to create low-level library modules and foreign interfaces (*e.g.*, as generated by *charon*).

OPTIONS

--cpp

Run the MBX file through the C preprocessor.

-I *dir*

This option is passed to the C preprocessor; it adds *dir* to the search path used to find include files.

-D *symbol*, **-D** *symbol=def*

This option is passed to the C preprocessor; it defines the preprocessor symbol *symb*.

SEEALSO

charon(1)

NAME

mbi-dump —

SYNOPSIS

mbi-dump

DESCRIPTION

SEEALSO

NAME

moby-idl —

SYNOPSIS

moby-idl

DESCRIPTION

SEEALSO

NAME

mobyc — a compiler for the MOBY programming language.

SYNOPSIS

mobyc file ...

DESCRIPTION

The *mobyc* command is used to compile and link MOBY programs.

OPTIONS

-?, **--help**
: Display a list of command-line options and then exit. If this option used in conjunction with the **-v** flag, a more detailed list of options is displayed (including internal debugging flags).

-t, **--threads**
: Enable the concurrency features of MOBY and link with the multithreaded versions of the MOBY libraries and runtime system.

-T, **--checkonly**

-S
: Creates an assembly file for each named source file, but does not produce object files or executables. The assembly-file name corresponds to the name of the source file, with a "`.s`" suffix substituted for the suffix of the source file.

-c
: Creates an object file for each named source file, but does not link the object files into an executable. The object-file name corresponds to the name of the source file, with a "`.o`" suffix substituted for the suffix of the source file.

-o *outfile*
: Create an executable named *outfile*. When specified with the **-S** option, the **-o** option is ignored. If neither **-o** and **-c** are not specified, a file named "`a.out`" is produced.

--filemap=*file*
: Use the *file* as the filemap instead of the default (`FILEMAP`). If this option is used in conjunction with the **--implicit-filemap** flag, then the file map is consulted first when mapping module names to files.

-i *lib*, **--use=***lib*
: Add the MOBY library "*lib*" to the libraries used by the source file.

--implicit-filemap
: Use an implicit mapping from module and signature names to file names. A module or signature "`Foo`" is assumed to be defined in the file "`Foo.mby`."

-I *path*
: Add the directory specified by *path* to the search path for MOBY libraries.

-l *library*
: Add the system library "*lib*" to the list of objects when linking.

-L *path*
: Add the directory specified by *path* to the search path for system libraries.

--main=*Module*
: Specifies that "*Module*" is the main module of the MOBY program (the default is `Main`).

--version
: Print the compiler version number and then exit.

-v
Run the compiler driver in verbose mode. This option can also be used with the **--help** flag to get a more detailed list of options.

1

The MOBY runtime system recognizes a number of command-line options that allow one to configure the execution environment and to aid in debugging the compiler and runtime. The options that control runtime parameters are:

--moby-alloc-pages=*n*
Specifies the number of pages allocated to the "*nursery*" (*i.e.*, used for small-object allocation). The default is 64 pages (512Kb).

--moby-task-pages=*n*
Specifies the number of allocation pages given to a task at one time. The default is 8 pages (64Kb).

The debugging flags allow various aspects of the runtime system to be traced. These flags are not available if the **–nodebug** linking option was specified.

--moby-debug=*file*
Specifies a file for debugging output.

--moby-debug-help
Prints a it of the debuggin options and exits.

--moby-debug-all
Enables all runtime-system tracing.

--moby-debug-gc
Enables tracing of the garbage collector.

--moby-debug-heap
Enables tracing of the heap.

--moby-debug-pcmaps
Enables tracing of the PC map registration and searching.

--moby-debug-static
Enables tracing of the static-data registration.

SEEALSO
charon(1)