NAME

mc − MOCKA, Modula-2 Compiler Karlsruhe running on sun/4 workstations

SYNOPSIS

mc [**-s** *module* | **-c** *module* | **-p** *module*]

[**-d** *dir*] ... [**-index**] [**-noindex**] [**-range**] [**-norange**]

[**-D** *directory*] [**-link** *script*] [**-edit** *script*] [**-list** *script*] [**-syslib** *dir*]

DESCRIPTION

mc is used to compile and link programs written in Modula-2.

mc -s *module* translates the DEFINITION MODULE *module* into a symbol file. This must be present when the IMPLEMENTATION MODULE *module* or a file importing *module* is compiled.

mc -c *module* translates the [IMPLEMENTATION] MODULE *module* into a code file. This must be present when a program containing *module* is created.

mc -p *module* creates an executable program for MODULE *module*.

If none of the options **-s** , **-c** , **-p** is specified, *mc* enters *session mode*.

In this mode the user need not worry about consistent module configurations and correct compilation order. Required compilations are triggered automatically. They are based on a dependency graph which is derived from the sources and updated when necessary.

Session mode also simplifies the correction of errors. If an error is detected a listing is created and the editor is invoked automatically. Errors may be corrected in the listing. After leaving the editor the listing (without the error messages) is written back to the source file.

OPTIONS

-s *module*	Create symbol file for *module*, i.e. compile the DEFINITION MODULE *module* in file *module*.md .
-c *module*	Create code file for *module*, i.e. compile the [IMPLEMENTATION] MODULE *module* in file *module*.mi .
-p *module*	Create program *module*, i.e. link code files for MODULE *module* and all (transitively) imported modules.
-d *dir*	Allow import from modules in library *dir*. This options may be repeated. Libraries (directories containing compiled modules) are inspected in the order specified. Finally the system library is inspected.
-index (**-noindex**)	Generate (don't generate) code for index checks.
-range (**-norange**)	Generate (don't generate) code for range checks.
-O	Optimize.
-g (**-nog**)	Produce (no) debugging information using the stabs format. gdb can work with this debugging information.
-gc (**-nogc**)	Produce (no) constant debugging information. gdb can use it, but dbx does currently not work with this.
-S (**-noS**)	The generated symbolic machine code is written to file *module*.s

The following options may be used to overwrite installation parameters.

-D *directory*	Specifies a directory where to place the compilation results (the files *.[dmiros]) in. This option defaults to the current directory.
-link *script*	Use *script* to invoke *ld*. When **-p** *module* is specified, *mc* collects all imported modules, checks them for consistency, creates a root module and then invokes *script module codefiles*

-edit *script* Use *script* to invoke the editor. When one of the commands **d** *module* or **i** *module* is given during session mode *script sourcefile* is called.

-list *script* Use *script* to invoke the lister. When an error is detectect during session mode *script sourcefile* is called.

-asm *script* Use *script* to invoke the assembler. The compiler produces assembler code, this script has to call the assembler to produce object code.

-syslib *dir* Use *dir* as system library.

COMMANDS

d *module* Edit DEFINITION MODULE *module*. (The *module* in a **d** or **i** command may be omitted. Then the *module* of a previous **d** or **i** command is used.)

i *module* Edit [IMPLEMENTATION] MODULE *module*.

s *module* Create symbol files for *module* and all (transitively) imported modules if they are missing or obsolete.

c *module* Create code file for *module* and symbol files for *module* and all (transitively) imported modules if they are missing or obsolete.

p *module* Create code file for *module* and code and symbol files for all (transitively) imported modules if they are missing or obsolete. Create program *module* if missing or obsolete. (*module* may be omitted. Then the *module* of a previous **p** command is used.)

<*empty*> The empty command is used to resume processing after editing a file. It is equivalent to the latest **s** , **c** or **p** command.

-flag (where *flag* stands for **index**, **noindex**, **range**, **norange** , **g** , **nog** , **S** , **noS**) has the the same meaning as the corresponding *mc* argument.

-noO (**-O**) Switch off and on optimizer. Only allowed if compiler was invoked with the -O option.

-info Shows current settings of compiler options.

q Quit.

unixcommand Commands not in the preceeding list are treated as Unix commands.

SPECIAL

Procedures written in other languages may be accessed by Modula-2 procedures. The compiler follows the type mapping and calling conventions of C. External entities must be defined in *foreign modules*. These are definition modules where the keyword DEFINITION is replaced by FOREIGN. For such a module the compiler does not insist on an implementation module. When linking a program an argument *"M.o"* for each foreign module *M* is passed to *ld*. (Hence, when an implementation of a foreign module uses a further file *N.o*, there should be an import of a corresponding (empty) foreign module *N*.) External procedures may not be assigned to procedure variables.

FILES

module.md Source file of DEFINITION MODULE *module*.

module.mi Source file of [IMPLEMENTATION] MODULE *module*.

module.d Symbol file for DEFINITION MODULE *module* (used for inter module type checking).

module.i Symbol file for [IMPLEMENATION] MODULE *module* (used for debugging).

module.m Mapping of code positions to source positions for *module* (used for debugging).

module.r Reference file for *module* (used for linking).

module.s	Assembler file for *module*.
module.o	Code file for *module*.
module	Executable program for MODULE *module*.

SEE ALSO

Programming in Modula-2 by Niklaus Wirth (Springer-Verlag Berlin, Heidelberg, New York, Tokyo; 3rd edition 1985)

BUGS

Only one *mc* process can run in the current directory. Only modules in the current directory are considered to determine the compilation order during session mode.