


```

-B      Build all modules
-C<x>   Code generation options:
  -C3      Turn on ieee error checking for constants
  -Ca<x>   Select ABI; see fpc -i or fpc -ia for possible values
  -Cb      Generate code for a big-endian variant of the target architecture
  -Cc<x>   Set default calling convention to <x>
  -CD      Create also dynamic library (not supported)
  -Ce      Compilation with emulated floating point opcodes
  -Cf<x>   Select fpu instruction set to use; see fpc -i or fpc -if for possible values
  -CF<x>   Minimal floating point constant precision (default, 32, 64)
  -Cg      Generate PIC code
  -Ch<n>[,m] <n> bytes min heap size (between 1023 and 67107840) and optionally
  -Ci      IO-checking
  -Cn      Omit linking stage
  -Co      Check overflow of integer operations
  -CO      Check for possible overflow of integer operations
  -Cp<x>   Select instruction set; see fpc -i or fpc -ic for possible values
  -CP<x>=<y> packing settings
    -CPPACKSET=<y> <y> set allocation: 0, 1 or DEFAULT or NORMAL, 2, 4 and 8
    -CPPACKENUM=<y> <y> enum packing: 0, 1, 2 and 4 or DEFAULT or NORMAL
    -CPPACKRECORD=<y> <y> record packing: 0 or DEFAULT or NORMAL, 1, 2, 4, 8, 16
  -Cr      Range checking
  -CR      Verify object method call validity
  -Cs<n>   Set stack checking size to <n>
  -Ct      Stack checking (for testing only, see manual)
  -CT<x>   Target-specific code generation options
    -CTcld      Emit a CLD instruction before using the x86 str
  -CX      Create also smartlinked library
-d<x>   Defines the symbol <x>
-D      Generate a DEF file
  -Dd<x>   Set description to <x>
  -Dv<x>   Set DLL version to <x>
-e<x>   Set path to executable
-E      Same as -Cn
-fPIC   Same as -Cg
-F<x>   Set file names and paths:
  -Fa<x>[,y] (for a program) load units <x> and [y] before uses is parsed
  -Fc<x>   Set input codepage to <x>
  -FC<x>   Set RC compiler binary name to <x>
  -Fd      Disable the compiler's internal directory cache
  -FD<x>   Set the directory where to search for compiler utilities
  -Fe<x>   Redirect error output to <x>
  -Ff<x>   Add <x> to framework path (Darwin only)
  -FE<x>   Set exe/unit output path to <x>
  -Fi<x>   Add <x> to include path
  -Fl<x>   Add <x> to library path
  -FL<x>   Use <x> as dynamic linker
  -Fm<x>   Load unicode conversion table from <x>.txt in the compiler dir
  -FM<x>   Set the directory where to search for unicode binary files
  -FN<x>   Add <x> to list of default unit scopes (namespaces)
  -Fo<x>   Add <x> to object path
  -Fr<x>   Load error message file <x>
  -FR<x>   Set resource (.res) linker to <x>
  -Fu<x>   Add <x> to unit path

```

APPENDIX A. ALPHABETICAL LISTING OF COMMAND LINE OPTIONS

-FU<x> Set unit output path to <x>, overrides -FE
-FW<x> Store generated whole-program optimization feedback in <x>
-Fw<x> Load previously stored whole-program optimization feedback from <x>
-g Generate debug information (default format for target)
-gc Generate checks for pointers (experimental, only available on some targets)
-gh Use heaptrace unit (for memory leak/corruption debugging)
-gl Use line info unit (show more info with backtraces)
-gm Generate Microsoft CodeView debug information (experimental)
-go<x> Set debug information options
 -godwarfsets Enable DWARF 'set' type debug information (breaks gdb < 6.5)
 -gostabsabsincludes Store absolute/full include file paths in Stabs
 -godwarfmethodclassprefix Prefix method names in DWARF with class name
 -godwarfcpp Simulate C++ debug information in DWARF
 -godwarfomflinnum Generate line number information in OMF LINNUM records in
-gp Preserve case in stabs symbol names
-gs Generate Stabs debug information
-gt Trash local variables (to detect uninitialized uses; multiple 't' characters
-gv Generates programs traceable with Valgrind
-gw Generate DWARFv2 debug information (same as -gw2)
-gw2 Generate DWARFv2 debug information
-gw3 Generate DWARFv3 debug information
-gw4 Generate DWARFv4 debug information (experimental)
-i Information
 -iD Return compiler date
 -iSO Return compiler OS
 -iSP Return compiler host processor
 -iTO Return target OS
 -iTP Return target processor
 -iV Return short compiler version
 -iW Return full compiler version
 -ia Return list of supported ABI targets
 -ic Return list of supported CPU instruction sets
 -if Return list of supported FPU instruction sets
 -ii Return list of supported inline assembler modes
 -io Return list of supported optimizations
 -ir Return list of recognized compiler and RTL features
 -it Return list of supported targets
 -iu Return list of supported microcontroller types
 -iw Return list of supported whole program optimizations
-I<x> Add <x> to include path
-k<x> Pass <x> to the linker
-l Write logo
-M<x> Set language mode to <x>
 -Mfpc Free Pascal dialect (default)
 -Mobjfpc FPC mode with Object Pascal support
 -Mdelphi Delphi 7 compatibility mode
 -Mtp TP/BP 7.0 compatibility mode
 -Mmacpas Macintosh Pascal dialects compatibility mode
 -Miso ISO 7185 mode
 -Mextendedpascal ISO 10206 mode
 -Mdelphiunicode Delphi 2009 and later compatibility mode
-n Do not read the default config files
-o<x> Change the name of the executable produced to <x>
-O<x> Optimizations:


```

-Tdarwin    Darwin/Mac OS X
-Tembedded  Embedded
-Temx       OS/2 via EMX (including EMX/RSX extender)
-Tfreebsd   FreeBSD
-Tgo32v2    Version 2 of DJ Delorie DOS extender
-Thaiku     Haiku
-Tiphonesim iPhoneSimulator from iOS SDK 3.2+ (older versions: -Tdarwin)
-Tlinux     Linux
-Tnativent  Native NT API (experimental)
-Tnetbsd    NetBSD
-Tnetware   Novell Netware Module (clib)
-Tnetwlibc  Novell Netware Module (libc)
-Topenbsd   OpenBSD
-Tos2       OS/2 / eComStation
-Tsymbian   Symbian OS
-Tsolaris   Solaris
-Twatcom    Watcom compatible DOS extender
-Twdosx     WDOSX DOS extender
-Twin32     Windows 32 Bit
-Twince     Windows CE
-u<x>       Undefines the symbol <x>
-U          Unit options:
  -Un       Do not check where the unit name matches the file name
  -Ur       Generate release unit files (never automatically recompiled)
  -Us       Compile a system unit
-v<x>       Be verbose. <x> is a combination of the following letters:
  e : Show errors (default)           0 : Show nothing (except errors)
  w : Show warnings                   u : Show unit info
  n : Show notes                      t : Show tried/used files
  h : Show hints                     c : Show conditionals
  i : Show general info               d : Show debug info
  l : Show linenumbers                r : Rhide/GCC compatibility mode
  s : Show time stamps                q : Show message numbers
  a : Show everything                 x : Show info about invoked tools
  b : Write file names messages       p : Write tree.log with parse tree
      with full path                  v : Write fpcdebug.txt with
  z : Write output to stderr           lots of debugging info
  m<x>,<y> : Do not show messages numbered <x> and <y>
-V<x>       Append '-<x>' to the used compiler binary name (e.g. for version)
-W<x>       Target-specific options (targets)
  -WA       Specify native type application (Windows)
  -Wb       Create a bundle instead of a library (Darwin)
  -WB       Create a relocatable image (Windows, Symbian)
  -WB<x>    Set image base to <x> (Windows, Symbian)
  -WC       Specify console type application (EMX, OS/2, Windows)
  -WD       Use DEFFILE to export functions of DLL or EXE (Windows)
  -We       Use external resources (Darwin)
  -WF       Specify full-screen type application (EMX, OS/2)
  -WG       Specify graphic type application (EMX, OS/2, Windows)
  -Wi       Use internal resources (Darwin)
  -WI       Turn on/off the usage of import sections (Windows)
  -WM<x>    Minimum Mac OS X deployment version: 10.4, 10.5.1, ... (Darwin)
  -WN       Do not generate relocation code, needed for debugging (Windows)
  -WP<x>    Minimum iOS deployment version: 3.0, 5.0.1, ... (iphonesim)

```

```

-WR      Generate relocation code (Windows)
-WX      Enable executable stack (Linux)
-X       Executable options:
-X9      Generate linkerscript for GNU Binutils ld older than version 2.19.1
-Xc      Pass --shared/-dynamic to the linker (BeOS, Darwin, FreeBSD, Linux)
-Xd      Do not search default library path (sometimes required for cross-compiling)
-Xe      Use external linker
-Xf      Substitute pthread library name for linking (BSD)
-Xg      Create debuginfo in a separate file and add a debuglink section to the executable
-XD      Try to link units dynamically (defines FPC_LINK_DYNAMIC)
-Xi      Use internal linker
-XLA     Define library substitutions for linking
-XLO     Define order of library linking
-XLD     Exclude default order of standard libraries
-Xm      Generate link map
-XM<x>   Set the name of the 'main' program routine (default is 'main')
-Xn      Use target system native linker instead of GNU ld (Solaris, AIX)
-Xp<x>   First search for the compiler binary in the directory <x>
-XP<x>   Prepend the binutils names with the prefix <x>
-Xr<x>   Set the linker's rlink-path to <x> (needed for cross compile, see the manual)
-XR<x>   Prepend <x> to all linker search paths (BeOS, Darwin, FreeBSD, Linux)
-Xs      Strip all symbols from executable
-XS      Try to link units statically (default, defines FPC_LINK_STATIC)
-Xt      Link with static libraries (-static is passed to linker)
-Xv      Generate table for Virtual Entry calls
-XV      Use VLink as external linker (default on Amiga, MorphOS)
-XX      Try to smartlink units (defines FPC_LINK_SMART)

-?       Show this help
-h       Shows this help without waiting

```

Appendix B

Alphabetical list of reserved words

absolute	far	popstack
abstract	file	private
and	finally	procedure
array	for	program
as	forward	property
asm	function	protected
assembler	goto	public
begin	if	raise
break	implementation	record
case	in	reintroduce
cdecl	index	repeat
class	inherited	self
const	initialization	set
constructor	inline	shl
continue	interface	shr
cppclass	interrupt	stdcall
deprecated	is	string
destructor	label	then
div	library	to
do	mod	true
downto	name	try
else	near	type
end	nil	unimplemented
except	not	unit
exit	object	until
export	of	uses
exports	on	var
external	operator	virtual
experimental	or	while
fail	otherwise	with
false	packed	xor

Appendix C

Compiler messages

This appendix is meant to list all the compiler messages. The list of messages is generated from the compiler source itself, and should be fairly complete. At this point, only assembler errors are not in the list.

For an explanation of how to control the messages, section [5.1.2](#), page [25](#).

C.1 General compiler messages

This section gives the compiler messages which are not fatal, but which display useful information. The number of such messages can be controlled with the various verbosity level `-v` switches.

Compiler: arg1

When the `-vt` switch is used, this line tells you what compiler is used.

Compiler OS: arg1

When the `-vd` switch is used, this line tells you what the source operating system is.

Info: Target OS: arg1

When the `-vd` switch is used, this line tells you what the target operating system is.

Using executable path: arg1

When the `-vt` switch is used, this line tells you where the compiler looks for its binaries.

Using unit path: arg1

When the `-vt` switch is used, this line tells you where the compiler looks for compiled units. You can set this path with the `-Fu` option.

Using include path: arg1

When the `-vt` switch is used, this line tells you where the compiler looks for its include files (files used in `{ $I xxx }` statements). You can set this path with the `-Fi` option.

Using library path: arg1

When the `-vt` switch is used, this line tells you where the compiler looks for the libraries. You can set this path with the `-Fl` option.

Using object path: arg1

When the `-vt` switch is used, this line tells you where the compiler looks for object files you link in (files used in `{ $L xxx }` statements). You can set this path with the `-Fo` option.

Info: arg1 lines compiled, arg2 secarg3

When the `-vi` switch is used, the compiler reports the number of lines compiled, and the time it took to compile them (real time, not program time).

Fatal: No memory left

The compiler doesn't have enough memory to compile your program. There are several remedies for this:

- If you're using the build option of the compiler, try compiling the different units manually.
- If you're compiling a huge program, split it up into units, and compile these separately.
- If the previous two don't work, recompile the compiler with a bigger heap. (You can use the `-Ch` option for this, `-Ch` (see page 28).)

Info: Writing Resource String Table file: arg1

This message is shown when the compiler writes the Resource String Table file containing all the resource strings for a program.

Error: Writing Resource String Table file: arg1

This message is shown when the compiler encounters an error when writing the Resource String Table file.

Info: Fatal:

Prefix for Fatal Errors.

Info: Error:

Prefix for Errors.

Info: Warning:

Prefix for Warnings.

Info: Note:

Prefix for Notes.

Info: Hint:

Prefix for Hints.

Error: Path "arg1" does not exist

The specified path does not exist.

Fatal: Compilation aborted

Compilation was aborted.

bytes code

The size of the generated executable code, in bytes.

bytes data

The size of the generated program data, in bytes.

Info: arg1 warning(s) issued

Total number of warnings issued during compilation.

Info: arg1 hint(s) issued

Total number of hints issued during compilation.

Info: arg1 note(s) issued

Total number of notes issued during compilation.

Fatal: I/O error: arg1

During compilation an I/O error happened which allows no further compilation.

Fatal: Operating system error: arg1

During compilation an operating system error happened which allows no further compilation.

Error: Compilation raised exception internally

Compilation was aborted, due to an exception generation.

Using unit scope: arg1

When the `-vt` switch is used, this line tells you what unit scopes (namespaces) the compiler is using when looking up units. You can add a unit scope with the `-FN` option.

C.2 Scanner messages.

This section lists the messages that the scanner emits. The scanner takes care of the lexical structure of the pascal file, i.e. it tries to find reserved words, strings, etc. It also takes care of directives and conditional compilation handling.

Fatal: Unexpected end of file

This typically happens in one of the following cases:

- The source file ends before the final `end.` statement. This happens mostly when the `begin` and `end` statements are not balanced;
- An include file ends in the middle of a statement.
- A comment was not closed.

Fatal: String exceeds line

There is a missing closing `'` in a string, so it occupies multiple lines.

Fatal: illegal character "arg1" (arg2)

An illegal character was encountered in the input file.

Fatal: Syntax error, "arg1" expected but "arg2" found

This indicates that the compiler expected a different token than the one you typed. It can occur almost anywhere it is possible to make an error against the Pascal language.

Start reading includefile arg1

When you provide the `-vt` switch, the compiler tells you when it starts reading an included file.

Warning: Comment level arg1 found

When the `-vw` switch is used, then the compiler warns you if it finds nested comments. Nested comments are not allowed in Turbo Pascal and Delphi, and can be a possible source of errors.

Note: Ignored compiler switch "arg1"

With `-vn` on, the compiler warns if it ignores a switch.

Warning: Illegal compiler switch "arg1"

You included a compiler switch (i.e. `{ $ . . . }`) which the compiler does not recognise.

Warning: Misplaced global compiler switch, ignored

The compiler switch is misplaced. It must be located at the start of the compilation unit, before the `uses` clause or any declaration.

Error: Illegal char constant

This happens when you specify a character with its ASCII code, as in `#96`, but the number is either illegal, or out of range.

Fatal: Cannot open file "arg1"

Free Pascal cannot find the program or unit source file you specified on the command line.

Fatal: Cannot open include file "arg1"

Free Pascal cannot find the source file you specified in a `{ $include .. }` statement.

Error: Illegal record alignment specifier "arg1"

You are specifying `{ $PACKRECORDS n }` or `{ $ALIGN n }` with an illegal value for `n`. For `$PACKRECORDS` valid alignments are 1, 2, 4, 8, 16, 32, C, NORMAL, DEFAULT, and for `$ALIGN` valid alignments are 1, 2, 4, 8, 16, 32, ON, OFF. Under mode MacPas `$ALIGN` also supports MAC68K, POWER and RESET.

Error: Illegal enum minimum-size specifier "arg1"

You are specifying the `{ $PACKENUM n }` with an illegal value for `n`. Only 1,2,4, NORMAL or DEFAULT is valid here.

Error: \$ENDIF expected for arg1 arg2 defined in arg3 line arg4

Your conditional compilation statements are unbalanced.

Error: Syntax error while parsing a conditional compiling expression

There is an error in the expression following the `{ $if .. }`, `{ $ifc }` or `{ $setc }` compiler directives.

Error: Evaluating a conditional compiling expression

There is an error in the expression following the `{ $if .. }`, `ifcorsetc` compiler directives.

Warning: Macro contents are limited to 255 characters in length

The contents of macros cannot be longer than 255 characters.

Error: ENDIF without IF(N)DEF

Your `{ $IFDEF .. }` and `{ $ENDIF }` statements are not balanced.

Fatal: User defined: arg1

A user defined fatal error occurred. See also the [Programmer's Guide](#).

Error: User defined: arg1

A user defined error occurred. See also the [Programmer's Guide](#).

Warning: User defined: arg1

A user defined warning occurred. See also the [Programmer's Guide](#).

Note: User defined: arg1

A user defined note was encountered. See also the [Programmer's Guide](#).

Hint: User defined: arg1

A user defined hint was encountered. See also the [Programmer's Guide](#).

Info: User defined: arg1

User defined information was encountered. See also the [Programmer's Guide](#).

Error: Keyword redefined as macro has no effect

You cannot redefine keywords with macros.

Fatal: Macro buffer overflow while reading or expanding a macro

Your macro or its result was too long for the compiler.

Warning: Expanding of macros exceeds a depth of 16.

When expanding a macro, macros have been nested to a level of 16. The compiler will expand no further, since this may be a sign that recursion is used.

Warning: compiler switches are not supported in // styled comments

Compiler switches should be in normal Pascal style comments.

Handling switch "arg1"

When you set debugging info on (`-vd`) the compiler tells you when it is evaluating conditional compile statements.

ENDIF arg1 found

When you turn on conditional messages (`-vc`), the compiler tells you where it encounters conditional statements.

IFDEF arg1 found, arg2

When you turn on conditional messages (`-vc`), the compiler tells you where it encounters conditional statements.

IFOPT arg1 found, arg2

When you turn on conditional messages (`-vc`), the compiler tells you where it encounters conditional statements.

IF arg1 found, arg2

When you turn on conditional messages (`-vc`), the compiler tells you where it encounters conditional statements.

IFNDEF arg1 found, arg2

When you turn on conditional messages (`-vc`), the compiler tells you where it encounters conditional statements.

ELSE arg1 found, arg2

When you turn on conditional messages (`-vc`), the compiler tells you where it encounters conditional statements.

Skipping until...

When you turn on conditional messages (`-vc`), the compiler tells you where it encounters conditional statements, and whether it is skipping or compiling parts.

Info: Press <return> to continue

When the `-vi` switch is used, the compiler stops compilation and waits for the `Enter` key to be pressed when it encounters a `{ $STOP }` directive.

Warning: Unsupported switch "arg1"

When warnings are turned on (`-vw`), the compiler warns you about unsupported switches. This means that the switch is used in Delphi or Turbo Pascal, but not in Free Pascal.

Warning: Illegal compiler directive "arg1"

When warnings are turned on (`-vw`), the compiler warns you about unrecognised switches. For a list of recognised switches, see the [Programmer's Guide](#).

Back in arg1

When you use the `-vt` switch, the compiler tells you when it has finished reading an include file.

Warning: Unsupported application type: "arg1"

You get this warning if you specify an unknown application type with the directive `{ $APPTYPE }`.

Warning: APPTYPE is not supported by the target OS

The `{ $APPTYPE }` directive is supported by certain operating systems only.

Warning: DESCRIPTION is not supported by the target OS

The `{ $DESCRIPTION }` directive is not supported on this target OS.

Note: VERSION is not supported by target OS

The { \$VERSION } directive is not supported on this target OS.

Note: VERSION only for exes or DLLs

The { \$VERSION } directive is only used for executable or DLL sources.

Warning: Wrong format for VERSION directive "arg1"

The { \$VERSION } directive format is majorversion.minorversion where majorversion and minorversion are words.

Error: Illegal assembler style specified "arg1"

When you specify an assembler mode with the { \$ASMMODE xxx } directive, the compiler didn't recognize the mode you specified.

Warning: ASM reader switch is not possible inside asm statement, "arg1" will be effective only for next

It is not possible to switch from one assembler reader to another inside an assembler block. The new reader will be used for next assembler statements only.

Error: Wrong switch toggle, use ON/OFF or +/-

You need to use ON or OFF or a + or - to toggle the switch.

Error: Resource files are not supported for this target

The target you are compiling for doesn't support resource files.

Warning: Include environment "arg1" not found in environment

The included environment variable cannot be found in the environment; it will be replaced by an empty string instead.

Error: Illegal value for FPU register limit

Valid values for this directive are 0..8 and NORMAL/DEFAULT.

Warning: Only one resource file is supported for this target

The target you are compiling for supports only one resource file. The first resource file found is used, the others are discarded.

Warning: Macro support has been turned off

A macro declaration has been found, but macro support is currently off, so the declaration will be ignored. To turn macro support on compile with -Sm on the command line or add { \$MACRO ON } in the source.

Error: Illegal interface type specified. Valid are COM, CORBA or DEFAULT.

The interface type that was specified is not supported.

Warning: APPID is only supported for PalmOS

The { \$APPID } directive is only supported for the PalmOS target.

Warning: APPNAME is only supported for PalmOS

The { \$APPNAME } directive is only supported for the PalmOS target.

Error: Constant strings cannot be longer than 255 chars

A single string constant can contain at most 255 chars. Try splitting up the string into multiple smaller parts and concatenate them with a + operator.

Fatal: Including include files exceeds a depth of 16.

When including include files the files have been nested to a level of 16. The compiler will expand no further, since this may be a sign that recursion is used.

Fatal: Too many levels of PUSH

A maximum of 20 levels is allowed. This error occurs only in mode MacPas.

Error: A POP without a preceding PUSH

This error occurs only in mode MacPas.

Error: Macro or compile time variable "arg1" does not have any value

Thus the conditional compile time expression cannot be evaluated.

Error: Wrong switch toggle, use ON/OFF/DEFAULT or +/-/*

You need to use ON or OFF or DEFAULT or a + or - or * to toggle the switch.

Error: Mode switch "arg1" not allowed here

A mode switch has already been encountered, or, in the case of option -Mmacpas, a mode switch occurs after UNIT.

Error: Compile time variable or macro "arg1" is not defined.

Thus the conditional compile time expression cannot be evaluated. Only in mode MacPas.

Error: UTF-8 code greater than 65535 found

Free Pascal handles UTF-8 strings internally as widestrings, i.e. the char codes are limited to 65535.

Error: Malformed UTF-8 string

The given string isn't a valid UTF-8 string.

UTF-8 signature found, using UTF-8 encoding

The compiler found a UTF-8 encoding signature (\$ef, \$bb, \$bf) at the beginning of a file, so it interprets it as a UTF-8 file.

Error: Compile time expression: Wanted arg1 but got arg2 at arg3

The type-check of a compile time expression failed.

Note: APPTYPE is not supported by the target OS

The {\$APPTYPE} directive is supported by certain operating systems only.

Error: Illegal optimization specified "arg1"

You specified an optimization with the {\$OPTIMIZATION xxx} directive, and the compiler didn't recognize the optimization you specified.

Warning: SETPEFLAGS is not supported by the target OS

The {\$SETPEFLAGS} directive is not supported by the target OS.

Warning: IMAGEBASE is not supported by the target OS

The {\$IMAGEBASE} directive is not supported by the target OS.

Warning: MINSTACKSIZE is not supported by the target OS

The {\$MINSTACKSIZE} directive is not supported by the target OS.

Warning: MAXSTACKSIZE is not supported by the target OS

The {\$MAXSTACKSIZE} directive is not supported by the target OS.

Error: Illegal state "arg1" for \$WARN directive

Only ON and OFF can be used as state with a {\$WARN} compiler directive.

Error: Illegal set packing value

Only 0, 1, 2, 4, 8, DEFAULT and NORMAL are allowed as packset parameters.

Warning: PIC directive or switch ignored

Several targets, such as WINDOWS, do not support nor need PIC, so the PIC directive and switch are ignored.

Warning: The switch "arg1" is not supported by the currently selected target

Some compiler switches like \$E are not supported by all targets.

Warning: Framework-related options are only supported for Darwin/Mac OS X

Frameworks are not a known concept, or at least not supported by FPC, on operating systems other than Darwin/Mac OS X.

Error: Illegal minimal floating point constant precision "arg1"

Valid minimal precisions for floating point constants are default, 32 and 64, which mean respectively minimal (usually 32 bit), 32 bit and 64 bit precision.

Warning: Overriding name of "main" procedure multiple times, was previously set to "arg1"

The name for the main entry procedure is specified more than once. Only the last name will be used.

Warning: Illegal identifier "arg1" for \$WARN directive

Identifier is not known by a { \$WARN } compiler directive.

Error: Illegal alignment directive

The alignment directive is not valid. Either the alignment type is not known or the alignment value is not a power of two.

Fatal: It is not possible to include a file that starts with an UTF-8 BOM in a module that uses a different code page

All source code that is part of a single compilation entity (program, library, unit) must be encoded in the same code page

Warning: Directive "arg1" is ignored for the current target platform

Some directives are ignored for certain targets, such as changing the packrecords and packenum settings on managed platforms.

Warning: Current system codepage "arg1" is not available for the compiler. Switching default codepage back to "arg2".

The current system codepage is not known by the compiler. The compiler is compiled with support for several codepages built-in. The codepage of the operation system is not in that list. You will need to recompile the compiler with support for this codepage.

Warning: SETPEOPTFLAGS is not supported by the target OS

The { \$SETPEOPTFLAGS } directive is not supported by the target OS.

Error: Illegal argument for SETPEFLAGS

The given argument for SETPEFLAGS is neither a correct named value nor an ordinal value

Error: Illegal argument for SETPEOPTFLAGS

The given argument for SETPEOPTFLAGS is neither a correct named value nor an ordinal value

Error: Directive arg1 is not supported on this target

Not all compiler directives are supported on all targets.

Warning: The specified stack size is not within the valid range for the platform. Setting the stack size ignored.

The valid range for the stack size is 1024 - 67107839 on 32-bit and 64-bit platforms and 1024 - 65520 on 16-bit platforms. Additionally, for Turbo Pascal 7 compatibility reasons, specifying a stack size of 65521 on 16-bit platforms actually sets the stack size to 65520.

Warning: The specified HeapMax value is smaller than the HeapMin value. Setting HeapMax ignored.

The HeapMax value (if specified) must be greater than or equal to the HeapMin value. Otherwise, the HeapMax value is ignored.

Error: Illegal argument for HUGEPOINTERNORMALIZATION

The only allowed values for HUGEPOINTERNORMALIZATION are BORLANDC, MICROSOFTC and WATCOMC.

Error: Illegal assembler CPU instruction set specified "arg1"

When you specify an assembler CPU with the `{ $ASMCPU xxx }` directive, the compiler didn't recognize the CPU you specified.

Warning: Specified syscall convention is not useable on this target

The specified syscall convention using the `{ $SYSCALL xxx }` directive, is not useable on the current target system.

Warning: Invalid syscall convention specified

The compiler did not recognize the syscall convention specified by the `{ $SYSCALL xxx }` directive.

Warning: SETPEUSERVERSION is not supported by the target OS

The `{ $SETPEUSERVERSION }` directive is not supported by the target OS.

Warning: SETPEOSVERSION is not supported by the target OS

The `{ $SETPEOSVERSION }` directive is not supported by the target OS.

Warning: SETPESUBSYSVERSION is not supported by the target OS

The `{ $SETPESUBSYSVERSION }` directive is not supported by the target OS.

Note: Changed CPU type to be consistent with specified controller

C.3 Parser messages

This section lists all parser messages. The parser takes care of the semantics of your language, i.e. it determines if your Pascal constructs are correct.

Error: Parser - Syntax Error

An error against the Turbo Pascal language was encountered. This typically happens when an illegal character is found in the source file.

Error: INTERRUPT procedure cannot be nested

An INTERRUPT procedure must be global.

Warning: Procedure type "arg1" ignored

The specified procedure directive is ignored by FPC programs.

Error: Not all declarations of "arg1" are declared with OVERLOAD

When you want to use overloading using the OVERLOAD directive, then all declarations need to have OVERLOAD specified.

Error: Duplicate exported function name "arg1"

Exported function names inside a specific DLL must all be different.

Error: Duplicate exported function index arg1

Exported function indexes inside a specific DLL must all be different.

Error: Invalid index for exported function

DLL function index must be in the range 1 . . \$FFFF.

Warning: Relocatable DLL or executable arg1 debug info does not work, disabled.

It is currently not possible to include debug information in a relocatable DLL.

Warning: To allow debugging for win32 code you need to disable relocation with -WN option

Stabs debug info is wrong for relocatable DLL or EXES. Use -WN if you want to debug win32 executables.

Error: Constructor name must be INIT

You are declaring an object constructor with a name which is not `init`, and the `-Ss` switch is in effect. See the switch `-Ss` (see page 35).

Error: Destructor name must be DONE

You are declaring an object destructor with a name which is not `done`, and the `-Ss` switch is in effect. See the switch `-Ss` (see page 35).

Error: Procedure type INLINE not supported

You tried to compile a program with C++ style inlining, and forgot to specify the `-Si` option (`-Si` (see page 35)). The compiler doesn't support C++ styled inlining by default.

Warning: Constructor should be public

Constructors must be in the 'public' part of an object (class) declaration.

Warning: Destructor should be public

Destructors must be in the 'public' part of an object (class) declaration.

Note: Class should have one destructor only

You can declare only one destructor for a class.

Error: Local class definitions are not allowed

Classes must be defined globally. They cannot be defined inside a procedure or function.

Fatal: Anonymous class definitions are not allowed

An invalid object (class) declaration was encountered, i.e. an object or class without methods that isn't derived from another object or class. For example:

```
Type o = object
      a : longint;
      end;
```

will trigger this error.

Note: The object "arg1" has no VMT

This is a note indicating that the declared object has no virtual method table.

Error: Illegal parameter list

You are calling a function with parameters that are of a different type than the declared parameters of the function.

Error: Wrong number of parameters specified for call to "arg1"

There is an error in the parameter list of the function or procedure – the number of parameters is not correct.

Error: overloaded identifier "arg1" isn't a function

The compiler encountered a symbol with the same name as an overloaded function, but it is not a function it can overload.

Error: overloaded functions have the same parameter list

You're declaring overloaded functions, but with the same parameter list. Overloaded function must have at least 1 different parameter in their declaration.

Error: function header doesn't match the previous declaration "arg1"

You declared a function with the same parameters but different result type or function modifiers.

Error: function header "arg1" doesn't match forward : var name changes arg2 => arg3

You declared the function in the `interface` part, or with the `forward` directive, but defined it with a different parameter list.

Note: Values in enumeration types have to be ascending

Free Pascal allows enumeration constructions as in C. Examine the following two declarations:

```
type a = (A_A, A_B, A_E:=6, A_UAS:=200) ;  
type a = (A_A, A_B, A_E:=6, A_UAS:=4) ;
```

The second declaration would produce an error. `A_UAS` needs to have a value higher than `A_E`, i.e. at least 7.

Error: With cannot be used for variables in a different segment

`With` stores a variable locally on the stack, but this is not possible if the variable belongs to another segment.

Error: function nesting > 31

You can nest function definitions only 31 levels deep.

Error: range check error while evaluating constants

The constants are out of their allowed range.

Warning: range check error while evaluating constants

The constants are out of their allowed range.

Error: duplicate case label

You are specifying the same label 2 times in a `case` statement.

Error: Upper bound of case range is less than lower bound

The upper bound of a `case` label is less than the lower bound and this is useless.

Error: typed constants of classes or interfaces are not allowed

You cannot declare a constant of type class or object.

Error: functions variables of overloaded functions are not allowed

You are trying to assign an overloaded function to a procedural variable. This is not allowed.

Error: string length must be a value from 1 to 255

The length of a shortstring in Pascal is limited to 255 characters. You are trying to declare a string with length less than 1 or greater than 255.

Warning: use extended syntax of NEW and DISPOSE for instances of objects

If you have a pointer `a` to an object type, then the statement `new(a)` will not initialize the object (i.e. the constructor isn't called), although space will be allocated. You should issue the `new(a, init)` statement. This will allocate space, and call the constructor of the object.

Warning: use of NEW or DISPOSE for untyped pointers is meaningless**Error: use of NEW or DISPOSE is not possible for untyped pointers**

You cannot use `new(p)` or `dispose(p)` if `p` is an untyped pointer because no size is associated to an untyped pointer. It is accepted for compatibility in `TP` and `DELPHI` modes, but the compiler will still warn you if it finds such a construct.

Error: class identifier expected

This happens when the compiler scans a procedure declaration that contains a dot, i.e., an object or class method, but the type in front of the dot is not a known type.

Error: type identifier not allowed here

You cannot use a type inside an expression.

Error: method identifier expected

This identifier is not a method. This happens when the compiler scans a procedure declaration that contains a dot, i.e., an object or class method, but the procedure name is not a procedure of this type.

Error: function header doesn't match any method of this class "arg1"

This identifier is not a method. This happens when the compiler scans a procedure declaration that contains a dot, i.e., an object or class method, but the procedure name is not a procedure of this type.

procedure/function arg1

When using the `-vd` switch, the compiler tells you when it starts processing a procedure or function implementation.

Error: Illegal floating point constant

The compiler expects a floating point expression, and gets something else.

Error: FAIL can be used in constructors only

You are using the `fail` keyword outside a constructor method.

Error: Destructors cannot have parameters

You are declaring a destructor with a parameter list. Destructor methods cannot have parameters.

Error: Only class methods, class properties and class variables can be referred with class references

This error occurs in a situation like the following:

```
Type :  
  Tclass = Class of Tobject;  
  
Var C : Tclass;  
  
begin  
  ...  
  C.free
```

`Free` is not a class method and hence cannot be called with a class reference.

Error: Only class methods, class properties and class variables can be accessed in class methods

This is related to the previous error. You cannot call a method of an object from inside a class method. The following code would produce this error:

```
class procedure tobject.x;  
  
begin  
  free
```

Because `free` is a normal method of a class it cannot be called from a class method.

Error: Constant and CASE types do not match

One of the labels is not of the same type as the case variable.

Error: The symbol cannot be exported from a library

You can only export procedures and functions when you write a library. You cannot export variables or constants.

Warning: An inherited method is hidden by "arg1"

A method that is declared `virtual` in a parent class, should be overridden in the descendant class with the `override` directive. If you don't specify the `override` directive, you will hide the parent method; you will not override it.

Error: There is no method in an ancestor class to be overridden: "arg1"

You are trying to `override` a virtual method of a parent class that does not exist.

Error: No member is provided to access property

You specified no `read` directive for a property.

Warning: Stored property directive is not yet implemented

This message is no longer used, as the `stored` directive has been implemented.

Error: Illegal symbol for property access

There is an error in the `read` or `write` directives for an array property. When you declare an array property, you can only access it with procedures and functions. The following code would cause such an error.

```
tmyobject = class
  i : integer;
  property x [i : integer]: integer read I write i;
```

Error: Cannot access a protected field of an object here

Fields that are declared in a `protected` section of an object or class declaration cannot be accessed outside the module where the object is defined, or outside descendent object methods.

Error: Cannot access a private field of an object here

Fields that are declared in a `private` section of an object or class declaration cannot be accessed outside the module where the class is defined.

Error: Overridden methods must have the same return type: "arg2" is overridden by "arg1" which has another return type

If you declare overridden methods in a class definition, they must have the same return type.

Error: EXPORT declared functions cannot be nested

You cannot declare a function or procedure within a function or procedure that was declared as an export procedure.

Error: Methods cannot be EXPORTed

You cannot declare a procedure that is a method for an object as `exported`.

Error: Call by var for arg no. arg1 has to match exactly: Got "arg2" expected "arg3"

When calling a function declared with `var` parameters, the variables in the function call must be of exactly the same type. There is no automatic type conversion.

Error: Class isn't a parent class of the current class

When calling inherited methods, you are trying to call a method of a non-related class. You can only call an inherited method of a parent class.

Error: SELF is only allowed in methods

You are trying to use the `self` parameter outside an object's method. Only methods get passed the `self` parameters.

Error: Methods can be only in other methods called direct with type identifier of the class

A construction like `sometype.somemethod` is only allowed in a method.

Error: Illegal use of ':'

You are using the format `:` (colon) 2 times on an expression that is not a real expression.

Error: range check error in set constructor or duplicate set element

The declaration of a set contains an error. Either one of the elements is outside the range of the set type, or two of the elements are in fact the same.

Error: Pointer to object expected

You specified an illegal type in a `new` statement. The extended syntax of `new` needs an object as a parameter.

Error: Expression must be constructor call

When using the extended syntax of `new`, you must specify the constructor method of the object you are trying to create. The procedure you specified is not a constructor.

Error: Expression must be destructor call

When using the extended syntax of `dispose`, you must specify the destructor method of the object you are trying to dispose of. The procedure you specified is not a destructor.

Error: Illegal order of record elements

When declaring a constant record, you specified the fields in the wrong order.

Error: Expression type must be class or record type, got arg1

A `with` statement needs an argument that is of the type `record` or `class`. You are using `with` on an expression that is not of this type.

Error: Procedures cannot return a value

In Free Pascal, you can specify a return value for a function when using the `exit` statement. This error occurs when you try to do this with a procedure. Procedures cannot return a value.

Error: constructors, destructors and class operators must be methods

You're declaring a procedure as destructor, constructor or class operator, when the procedure isn't a class method.

Error: Operator is not overloaded

You're trying to use an overloaded operator when it is not overloaded for this type.

Error: Impossible to overload assignment for equal types

You cannot overload assignment for types that the compiler considers as equal.

Error: Impossible operator overload

The combination of operator, arguments and return type are incompatible.

Error: Re-raise isn't possible there

You are trying to re-raise an exception where it is not allowed. You can only re-raise exceptions in an `except` block.

Error: The extended syntax of new or dispose isn't allowed for a class

You cannot generate an instance of a class with the extended syntax of `new`. The constructor must be used for that. For the same reason, you cannot call `dispose` to de-allocate an instance of a class, the destructor must be used for that.

Error: Procedure overloading is switched off

When using the `-So` switch, procedure overloading is switched off. Turbo Pascal does not support function overloading.

Error: It is not possible to overload this operator. Related overloadable operators (if any) are: `arg1`

You are trying to overload an operator which cannot be overloaded. The following operators can be overloaded :

`+, -, *, /, =, >, <, <=, >=, is, as, in, **, :=`

Error: Comparative operator must return a boolean value

When overloading the `=` operator, the function must return a boolean value.

Error: Only virtual methods can be abstract

You are declaring a method as abstract, when it is not declared to be virtual.

Fatal: Use of unsupported feature: "arg1".

You're trying to force the compiler into doing something it cannot do yet.

Error: The mix of different kind of objects (class, object, interface, etc) isn't allowed

You cannot derive `objects`, `classes`, `cppclasses` and `interfaces` intertwined. E.g. a class cannot have an object as parent and vice versa.

Warning: Unknown procedure directive had to be ignored: "arg1"

The procedure directive you specified is unknown.

Error: `arg1` can be associated with only one variable

You cannot specify more than one variable before the `absolute`, `export`, `external`, `weakexternal`, `public` and `cvar` directives. As a result, for example the following construct will provide this error:

```
Var Z : Longint;  
    X,Y : Longint absolute Z;
```

Error: `absolute` can only be associated with a `var` or `const`

The address of an `absolute` directive can only point to a variable or constant. Therefore, the following code will produce this error:

```
Procedure X;  
  
var p : longint absolute x;
```

Error: Only one variable can be initialized

You cannot specify more than one variable with a initial value in Delphi mode.

Error: Abstract methods shouldn't have any definition (with function body)

Abstract methods can only be declared, you cannot implement them. They should be overridden by a descendant class.

Error: This overloaded function cannot be local (must be exported)

You are defining an overloaded function in the implementation part of a unit, but there is no corresponding declaration in the interface part of the unit.

Warning: Virtual methods are used without a constructor in "arg1"

If you declare objects or classes that contain virtual methods, you need to have a constructor and destructor to initialize them. The compiler encountered an object or class with virtual methods that doesn't have a constructor/destructor pair.

Macro defined: arg1

When `-vc` is used, the compiler tells you when it defines macros.

Macro undefined: arg1

When `-vc` is used, the compiler tells you when it undefines macros.

Macro arg1 set to arg2

When `-vc` is used, the compiler tells you what values macros get.

Info: Compiling arg1

When you turn on information messages (`-vi`), the compiler tells you what units it is recompiling.

Parsing interface of unit arg1

This tells you that the reading of the interface of the current unit has started

Parsing implementation of arg1

This tells you that the code reading of the implementation of the current unit, library or program starts

Compiling arg1 for the second time

When you request debug messages (`-vd`) the compiler tells you what units it recompiles for the second time.

Error: No property found to override

You want to override a property of a parent class, when there is, in fact, no such property in the parent class.

Error: Only one default property is allowed

You specified a property as `Default`, but the class already has a default property, and a class can have only one default property.

Error: The default property must be an array property

Only array properties of classes can be made `default` properties.

Error: Virtual constructors are only supported in class object model

You cannot have virtual constructors in objects. You can only have them in classes.

Error: No default property available

You are trying to access a default property of a class, but this class (or one of its ancestors) doesn't have a default property.

Error: The class cannot have a published section, use the `{M+}` switch

If you want a `published` section in a class definition, you must use the `{M+}` switch, which turns on generation of type information.

Error: Forward declaration of class "arg1" must be resolved here to use the class as ancestor

To be able to use an object as an ancestor object, it must be defined first. This error occurs in the following situation:

```
Type ParentClass = Class;
  ChildClass = Class(ParentClass)
  ...
end;
```

where `ParentClass` is declared but not defined.

Error: Local operators not supported

You cannot overload locally, i.e. inside procedures or function definitions.

Error: Procedure directive "arg1" not allowed in interface section

This procedure directive is not allowed in the `interface` section of a unit. You can only use it in the `implementation` section.

Error: Procedure directive "arg1" not allowed in implementation section

This procedure directive is not allowed in the `implementation` section of a unit. You can only use it in the `interface` section.

Error: Procedure directive "arg1" not allowed in procvar declaration

This procedure directive cannot be part of a procedural or function type declaration.

Error: Function is already declared Public/Forward "arg1"

You will get this error if a function is defined as `forward` twice. Or if it occurs in the `interface` section, and again as a `forward` declaration in the `implementation` section.

Error: Cannot use both EXPORT and EXTERNAL

These two procedure directives are mutually exclusive.

Hint: "arg1" not yet supported inside inline procedure/function

Inline procedures don't support this declaration.

Hint: Inlining disabled

Inlining of procedures is disabled.

Info: Writing Browser log arg1

When information messages are on, the compiler warns you when it writes the browser log (generated with the `{ $Y+ }` switch).

Hint: may be pointer dereference is missing

The compiler thinks that a pointer may need a dereference.

Fatal: Selected assembler reader not supported

The selected assembler reader (with `{ $ASMMODE xxx }`) is not supported. The compiler can be compiled with or without support for a particular assembler reader.

Error: Procedure directive "arg1" cannot be used with arg2

You specified a procedure directive that conflicts with other directives. For instance `cdecl` and `pascal` are mutually exclusive.

Error: Calling convention doesn't match forward

This error happens when you declare a function or procedure with e.g. `cdecl`; but omit this directive in the implementation, or vice versa. The calling convention is part of the function declaration, and must be repeated in the function definition.

Error: Property cannot have a default value

Set properties or indexed properties cannot have a default value.

Error: The default value of a property must be constant

The value of a `default` declared property must be known at compile time. The value you specified is only known at run time. This happens e.g. if you specify a variable name as a default value.

Error: Symbol cannot be published, can be only a class

Only class type variables can be in a `published` section of a class if they are not declared as a property.

Error: This kind of property cannot be published

Properties in a `published` section cannot be array properties. They must be moved to public sections. Properties in a `published` section must be an ordinal type, a real type, strings or sets.

Error: An import name is required

Some targets need a name for the imported procedure or a `cdecl` specifier.

Error: Division by zero

A division by zero was encountered.

Error: Invalid floating point operation

An operation on two real type values produced an overflow or a division by zero.

Error: Upper bound of range is less than lower bound

The upper bound of an array declaration is less than the lower bound and this is not possible.

Warning: string "arg1" is longer than "arg2"

The size of the constant string is larger than the size you specified in string type definition.

Error: string length is larger than array of char length

The size of the constant string is larger than the size you specified in the `Array[x..y]` of `char` definition.

Error: Illegal expression after message directive

Free Pascal supports only integer or string values as message constants.

Error: Message handlers can take only one call by ref. parameter

A method declared with the `message` directive as message handler can take only one parameter which must be declared as call by reference. Parameters are declared as call by reference using the `var`-directive.

Error: Duplicate message label: "arg1"

A label for a message is used twice in one object/class.

Error: Self can only be an explicit parameter in methods which are message handlers

The `Self` parameter can only be passed explicitly to a method which is declared as message handler.

Error: Threadvars can be only static or global

Threadvars must be static or global; you cannot declare a thread local to a procedure. Local variables are always local to a thread, because every thread has its own stack and local variables are stored on the stack.

Fatal: Direct assembler not supported for binary output format

You cannot use direct assembler when using a binary writer. Choose an other output format or use another assembler reader.

Warning: Don't load OBJPAS unit manually, use {\$mode objfpc} or {\$mode delphi} instead

You are trying to load the `ObjPas` unit manually from a `uses` clause. This is not a good idea. Use the `{ $MODE OBJFPC }` or `{ $mode delphi }` directives which load the unit automatically.

Error: OVERRIDE cannot be used in objects

Override is not supported for objects, use `virtual` instead to override a method of a parent object.

Error: Data types which require initialization/finalization cannot be used in variant records

Some data types (e.g. `ansistring`) need initialization/finalization code which is implicitly generated by the compiler. Such data types cannot be used in the variant part of a record.

Error: Resourcestrings can be only static or global

`Resourcestring` cannot be declared local, only global or using the `static` directive.

Error: Exit with argument cannot be used here

An `exit` statement with an argument for the return value cannot be used here. This can happen for example in `try..except` or `try..finally` blocks.

Error: The type of the storage symbol must be boolean

If you specify a storage symbol in a property declaration, it must be a boolean type.

Error: This symbol isn't allowed as storage symbol

You cannot use this type of symbol as storage specifier in property declaration. You can use only methods with the result type boolean, boolean class fields or boolean constants.

Error: Only classes which are compiled in \$M+ mode can be published

A class-typed field in the published section of a class can only be a class which was compiled in `{ $M+ }` or which is derived from such a class. Normally such a class should be derived from `TPersistent`.

Error: Procedure directive expected

This error is triggered when you have a `{ $Calling }` directive without a calling convention specified. It also happens when declaring a procedure in a `const` block and you used a `;` after a procedure declaration which must be followed by a procedure directive. Correct declarations are:

```
const
  p : procedure; stdcall=nil;
  p : procedure stdcall=nil;
```

Error: The value for a property index must be of an ordinal type

The value you use to index a property must be of an ordinal type, for example an integer or enumerated type.

Error: Procedure name too short to be exported

The length of the procedure/function name must be at least 2 characters long. This is because of a bug in `dlltool` which doesn't parse the `.def` file correctly with a name of length 1.

Error: No DEFFILE entry can be generated for unit global vars**Error: Compile without -WD option**

You need to compile this file without the `-WD` switch on the command line.

Fatal: You need ObjFpc (-S2) or Delphi (-Sd) mode to compile this module

You need to use `{ $MODE OBJFPC }` or `{ $MODE DELPHI }` to compile this file. Or use the corresponding command line switch, either `-Mobjfpc` or `-MDelphi`.

Error: Cannot export with index under arg1

Exporting of functions or procedures with a specified index is not supported on this target.

Error: Exporting of variables is not supported under arg1

Exporting of variables is not supported on this target.

Error: Improper GUID syntax

The GUID indication does not have the proper syntax. It should be of the form

```
{XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}
```

Where each X represents a hexadecimal digit.

Warning: Procedure named "arg1" not found that is suitable for implementing the arg2.arg3

The compiler cannot find a suitable procedure which implements the given method of an interface. A procedure with the same name is found, but the arguments do not match.

Error: interface identifier expected

This happens when the compiler scans a class declaration that contains interface function name mapping code like this:

```
type
  TMyObject = class(TObject, IDispatch)
    function IUnknown.QueryInterface=MyQueryInterface;
    ....
```

and the interface before the dot is not listed in the inheritance list.

Error: Type "arg1" cannot be used as array index type

Types like `qword` or `int64` are not allowed as array index type.

Error: Con- and destructors are not allowed in interfaces

Constructor and destructor declarations are not allowed in interfaces. In the most cases method `QueryInterface` of `IUnknown` can be used to create a new interface.

Error: Access specifiers cannot be used in INTERFACES and OBJCPROTOCOLS

The access specifiers `public`, `private`, `protected` and `published` cannot be used in interfaces, Objective-C protocols and categories because all methods of an interface/protocol/category must be public.

Error: An interface, helper or Objective-C protocol or category cannot contain fields

Declarations of fields are not allowed in interfaces, helpers and Objective-C protocols and categories. An interface/helper/protocol/category can contain only methods and properties with method read/write specifiers.

Error: Cannot declare local procedure as EXTERNAL

Declaring local procedures as external is not possible. Local procedures get hidden parameters that will make the chance of errors very high.

Warning: Some fields coming before "arg1" were not initialized

In Delphi mode, not all fields of a typed constant record have to be initialized, but the compiler warns you when it detects such situations.

Error: Some fields coming before "arg1" were not initialized

In all syntax modes but Delphi mode, you cannot leave some fields uninitialized in the middle of a typed constant record.

Warning: Some fields coming after "arg1" were not initialized

You can leave some fields at the end of a type constant record uninitialized (The compiler will initialize them to zero automatically). This may be the cause of subtle problems.

Error: VarArgs directive (or '...' in MacPas) without CDecl/CppDecl/MWPascal/StdCall and External

The varargs directive (or the "..." varargs parameter in MacPas mode) can only be used with procedures or functions that are declared with `external` and one of `cdecl`, `stdcall`, `stdcall` and `mwpascal`. This functionality is only supported to provide a compatible interface to C functions like `printf`.

Error: Self must be a normal (call-by-value) parameter

You cannot declare `Self` as a `const` or `var` parameter, it must always be a call-by-value parameter.

Error: Interface "arg1" has no interface identification

When you want to assign an interface to a constant, then the interface must have a GUID value set.

Error: Unknown class field or method identifier "arg1"

Properties must refer to a field or method in the same class.

Warning: Overriding calling convention "arg1" with "arg2"

There are two directives in the procedure declaration that specify a calling convention. Only the last directive will be used.

Error: Typed constants of the type "procedure of object" can only be initialized with NIL

You cannot assign the address of a method to a typed constant which has a 'procedure of object' type, because such a constant requires two addresses: that of the method (which is known at compile time) and that of the object or class instance it operates on (which cannot be known at compile time).

Error: Default value can only be assigned to one parameter

It is not possible to specify a default value for several parameters at once. The following is invalid:

```
Procedure MyProcedure (A,B : Integer = 0);
```

Instead, this should be declared as

```
Procedure MyProcedure (A : Integer = 0; B : Integer = 0);
```

Error: Default parameter required for "arg1"

The specified parameter requires a default value.

Warning: Use of unsupported feature!

You're trying to force the compiler into doing something it cannot do yet.

Hint: C arrays are passed by reference

Any array passed to a C function is passed by a pointer (i.e. by reference).

Error: C array of const must be the last argument

You cannot add any other argument after an `array of const` for `cdecl` functions, as the size pushed on stack for this argument is not known.

Hint: Type "arg1" redefinition

This is an indicator that a previously declared type is being redefined as something else. This may, or may not be, a potential source of errors.

Warning: cdecl'ared functions have no high parameter

Functions declared with the `cdecl` modifier do not pass an extra implicit parameter.

Warning: cdecl'ared functions do not support open strings

Openstring is not supported for functions that have the `cdecl` modifier.

Error: Cannot initialize variables declared as threadvar

Variables declared as `threadvar` cannot be initialized with a default value. The variables will always be filled with zero at the start of a new thread.

Error: Message directive is only allowed in Classes

The message directive is only supported for Class types.

Error: Procedure or Function expected

A class method can only be specified for procedures and functions.

Warning: Calling convention directive ignored: "arg1"

Some calling conventions are supported only by certain CPUs. I.e. most non-i386 ports support only the standard ABI calling convention of the CPU.

Error: REINTRODUCE cannot be used in objects

`reintroduce` is not supported for objects, Objective-C classes and Objective-C protocols.

Error: Each argument must have its own location

If locations for arguments are specified explicitly as it is required by some syscall conventions, each argument must have its own location. Things like

```
procedure p(i, j : longint 'r1');
```

are not allowed.

Error: Each argument must have an explicit location

If one argument has an explicit argument location, all arguments of a procedure must have one.

Error: Unknown argument location

The location specified for an argument isn't recognized by the compiler.

Error: 32 Bit-Integer or pointer variable expected

The libbase for MorphOS/AmigaOS can be given only as `longint`, `dword` or any pointer variable.

Error: Goto statements are not allowed between different procedures

It isn't allowed to use `goto` statements referencing labels outside the current procedure. The following example shows the problem:

```
...
procedure p1;
label
  l1;

procedure p2;
begin
  goto l1; // This goto ISN'T allowed
end;

begin
  p2
```

```
11:
end;
...
```

Fatal: Procedure too complex, it requires too many registers

Your procedure body is too long for the compiler. You should split the procedure into multiple smaller procedures.

Error: Illegal expression

This can occur under many circumstances. Usually when trying to evaluate constant expressions.

Error: Invalid integer expression

You made an expression which isn't an integer, and the compiler expects the result to be an integer.

Error: Illegal qualifier

One of the following is happening :

- You're trying to access a field of a variable that is not a record.
- You're indexing a variable that is not an array.
- You're dereferencing a variable that is not a pointer.

Error: High range limit < low range limit

You are declaring a subrange, and the high limit is less than the low limit of the range.

Error: Exit's parameter must be the name of the procedure it is used in or of a surrounding procedure

The parameter of a `exit` call in `macpas` mode must be either the name of the current subroutine or of a surrounding one

Error: Illegal assignment to for-loop variable "arg1"

The type of a `for` loop variable must be an ordinal type. Loop variables cannot be reals or strings. You also cannot assign values to loop variables inside the loop (Except in Delphi and TP modes). Use a `while` or `repeat` loop instead if you need to do something like that, since those constructs were built for that.

Error: Cannot declare local variable as EXTERNAL

Declaring local variables as external is not allowed. Only global variables can reference external variables.

Error: Procedure is already declared EXTERNAL

The procedure is already declared with the `EXTERNAL` directive in an interface or forward declaration.

Warning: Implicit uses of Variants unit

The Variant type is used in the unit without any used unit using the Variants unit. The compiler has implicitly added the Variants unit to the uses list. To remove this warning the Variants unit needs to be added to the uses statement.

Error: Class and static methods cannot be used in INTERFACES

The specifier `class` and directive `static` cannot be used in interfaces because all methods of an interface must be public.

Error: Overflow in arithmetic operation

An operation on two integer values produced an overflow.

Error: Protected or private expected

`strict` can be only used together with `protected` or `private`.

Error: SLICE cannot be used outside of parameter list

`slice` can be used only for arguments accepting an open array parameter.

Error: A DISPINTERFACE cannot have a parent class

A DISPINTERFACE is a special type of interface which cannot have a parent class. Dispinterface always derive from IDispatch type.

Error: A DISPINTERFACE needs a guid

A DISPINTERFACE always needs an interface identification (a GUID).

Warning: Overridden methods must have a related return type. This code may crash, it depends on a Delphi parser bug ("arg2" is overridden by "arg1" which has another return type)

If you declare overridden methods in a class definition, they must have the same return type. Some versions of Delphi allow you to change the return type of interface methods, and even to change procedures into functions, but the resulting code may crash depending on the types used and the way the methods are called.

Error: Dispatch IDs must be ordinal constants

The `dispid` keyword must be followed by an ordinal constant (the `dispid` index).

Error: The range of the array is too large

Regardless of the size taken up by its elements, an array cannot have more than `high(ptrint)` elements. Additionally, the range type must be a subrange of `ptrint`.

Error: The address cannot be taken of bit packed array elements and record fields

If you declare an array or record as `packed` in Mac Pascal mode (or as `packed` in any mode with `{ $bitpacking on }`), it will be packed at the bit level. This means it becomes impossible to take addresses of individual array elements or record fields. The only exception to this rule is in the case of packed arrays elements whose packed size is a multiple of 8 bits.

Error: Dynamic arrays cannot be packed

Only regular (and possibly in the future also open) arrays can be packed.

Error: Bit packed array elements and record fields cannot be used as loop variables

If you declare an array or record as `packed` in Mac Pascal mode (or as `packed` in any mode with `{ $bitpacking on }`), it will be packed at the bit level. For performance reasons, they cannot be used as loop variables.

Error: VAR, TYPE and CONST are allowed only in records, objects and classes

The usage of VAR, TYPE and CONST to declare new types inside an object is allowed only inside records, objects and classes.

Error: This type cannot be a generic

Only Classes, Objects, Interfaces and Records are allowed to be used as generic.

Warning: Don't load LINEINFO unit manually, Use the -gl compiler switch instead

Do not use the `lineinfo` unit directly, Use the `-gl` switch which automatically adds the correct unit for reading the selected type of debugging information. The unit that needs to be used depends on the type of debug information used when compiling the binary.

Error: No function result type specified for function "arg1"

The first time you declare a function you have to declare it completely, including all parameters and the result type.

Error: Specialization is only supported for generic types

Types which are not generics cannot be specialized.

Error: Generics cannot be used as parameters when specializing generics

When specializing a generic, only non-generic types can be used as parameters.

Error: Constants of objects containing a VMT are not allowed

If an object requires a VMT either because it contains a constructor or virtual methods, it's not allowed to create constants of it. In TP and Delphi mode this is allowed for compatibility reasons.

Error: Taking the address of labels defined outside the current scope isn't allowed

It isn't allowed to take the address of labels outside the current procedure.

Error: Cannot initialize variables declared as external

Variables declared as external cannot be initialized with a default value.

Error: Illegal function result type

Some types like file types cannot be used as function result.

Error: No common type possible between "arg1" and "arg2"

To perform an operation on integers, the compiler converts both operands to their common type, which appears to be an invalid type. To determine the common type of the operands, the compiler takes the minimum of the minimal values of both types, and the maximum of the maximal values of both types. The common type is then minimum..maximum.

Error: Generics without specialization cannot be used as a type for a variable

Generics must be always specialized before being used as variable type.

Warning: Register list is ignored for pure assembler routines

When using pure assembler routines, the list with modified registers is ignored.

Error: Implements property must have class or interface type

A property which implements an interface must be of type class or interface.

Error: Implements-property must implement interface of correct type, found "arg1" expected "arg2"

A property which implements an interface actually implements a different interface.

Error: Implements-property must have read specifier

A property which implements an interface must have at least a read specifier.

Error: Implements-property must not have write-specifier

A property which implements an interface may not have a write specifier.

Error: Implements-property must not have stored-specifier

A property which implements an interface may not have a stored specifier.

Error: Implements-property used on unimplemented interface: "arg1"

The interface which is implemented by a property is not an interface implemented by the class.

Error: Floating point not supported for this target

The compiler parsed a floating point expression, but it is not supported.

Error: Class "arg1" does not implement interface "arg2"

The delegated interface is not implemented by the class given in the implements clause.

Error: Type used by implements must be an interface

The `implements` keyword must be followed by an interface type.

Error: Variables cannot be exported with a different name on this target, add the name to the declaration using the "export" directive (variable name: `arg1`, declared export name: `arg2`)

On most targets it is not possible to change the name under which a variable is exported inside the `exports` statement of a library. In that case, you have to specify the export name at the point where the variable is declared, using the `export` and `alias` directives.

Error: Weak external symbols are not supported for the current target

A "weak external" symbol is a symbol which may or may not exist at (either static or dynamic) link time. This concept may not be available (or implemented yet) on the current `cpu/OS` target.

Error: Forward type definition does not match

Classes and interfaces being defined forward must have the same type when being implemented. A forward interface cannot be changed into a class.

Note: Virtual method "`arg1`" has a lower visibility (`arg2`) than parent class `arg3` (`arg4`)

The virtual method overrides an method that is declared with a higher visibility. This might give unexpected results. E.g., in case the new visibility is `private` then a call to "inherited" in a new child class will call the higher-visible method in a parent class and ignores the `private` method.

Error: Fields cannot appear after a method or property definition, start a new visibility section first

Once a method or property has been defined in a class or object, you cannot define any fields afterwards without starting a new visibility section (such as `public`, `private`, etc.). The reason is that otherwise the source code can appear ambiguous to the compiler, since it is possible to use modifiers such as `default` and `register` also as field names.

Error: Parameters or result types cannot contain local type definitions. Use a separate type definition in a type block.

In Pascal, types are not considered to be identical simply because they are semantically equivalent. Two variables or parameters are only considered to be of the same type if they refer to the same type definition. As a result, it is not allowed to define new types inside parameter lists, because then it is impossible to refer to the same type definition in the procedure headers of the interface and implementation of a unit (both procedure headers would define a separate type). Keep in mind that expressions such as "file of byte" or "string[50]" also define a new type.

Error: ABSTRACT and SEALED conflict

`ABSTRACT` and `SEALED` cannot be used together in one declaration

Error: Cannot create a descendant of the sealed class "`arg1`"

Sealed means that class cannot be derived by another class.

Error: SEALED class cannot have an ABSTRACT method

Sealed means that class cannot be derived. Therefore no one class is able to override an abstract method in a sealed class.

Error: Only virtual methods can be final

You are declaring a method as `final`, when it is not declared to be `virtual`.

Error: Final method cannot be overridden: "`arg1`"

You are trying to `override` a virtual method of a parent class that does not exist.

Error: Only one message can be used per method.

It is not possible to associate multiple messages with a single method.

Error: Invalid enumerator identifier: "arg1"

Only "MoveNext" and "Current" enumerator identifiers are supported.

Error: Enumerator identifier required

"MoveNext" or "Current" identifier must follow the `enumerator` modifier.

Error: Enumerator MoveNext pattern method is not valid. Method must be a function with the Boolean return type and no required arguments.

"MoveNext" enumerator pattern method must be a function with Boolean return type and no required arguments

Error: Enumerator Current pattern property is not valid. Property must have a getter.

"Current" enumerator pattern property must have a getter

Error: Only one enumerator MoveNext method is allowed per class/object

Class or Object can have only one enumerator MoveNext declaration.

Error: Only one enumerator Current property is allowed per class/object

Class or Object can have only one enumerator Current declaration.

Error: For in loop cannot be used for the type "arg1"

For in loop can be used not for all types. For example it cannot be used for the enumerations with jumps.

Error: Objective-C messages require their Objective-C selector name to be specified using the "message" directive.

Objective-C messages require their Objective-C name (selector name) to be specified using the `message `someName:`` procedure directive. While bindings to other languages automatically generate such names based on the identifier you use (by replacing all underscores with colons), this is unsafe since nothing prevents an Objective-C method name to contain actual colons.

Error: Objective-C does not have formal constructors nor destructors. Use the alloc, initXXX and dealloc messages.

The Objective-C language does not have any constructors or destructors. While there are some messages with a similar purpose (such as `init` and `dealloc`), these cannot be identified using automatic parsers and do not guarantee anything like Pascal constructors/destructors (e.g., you have to take care of only calling “designated” inherited “constructors”). For these reasons, we have opted to follow the standard Objective-C patterns for instance creation/destruction.

Error: Message name is too long (max. 255 characters)

Due to compiler implementation reasons, message names are currently limited to 255 characters.

Error: Objective-C message symbol name for "arg1" is too long

Due to compiler implementation reasons, mangled message names (i.e., the symbol names used in the assembler code) are currently limited to 255 characters.

Hint: Defining a new Objective-C root class. To derive from another root class (e.g., NSObject), specify it as the parent class.

If no parent class is specified for an Object Pascal class, then it automatically derives from TObject. Objective-C classes however do not automatically derive from NSObject, because one can have multiple root classes in Objective-C. For example, in the Cocoa framework both NSObject and NSProxy are root classes. Therefore, you have to explicitly define a parent class (such as NSObject) if you want to derive your Objective-C class from it.

Error: Objective-C classes cannot have published sections.

In Object Pascal, “published” determines whether or not RTTI is generated. Since the Objective-C runtime always needs RTTI for everything, this specified does not make sense for Objective-C classes.

Fatal: This module requires an Objective-C mode switch to be compiled

This error indicates the use of Objective-C language features without an Objective-C mode switch active. Enable one via the `-M` command line switch, or the `$modeswitch x` directive.

Error: Inherited methods can only be overridden in Objective-C and Java, add "override" (inherited method defined in arg1)**Hint: Inherited methods can only be overridden in Objective-C and Java, add "override" (inherited method defined in arg1).**

It is not possible to `reintroduce` methods in Objective-C or Java like in Object Pascal. Methods with the same name always map to the same virtual method entry. In order to make this clear in the source code, the compiler always requires the `override` directive to be specified when implementing overriding Objective-C or Java methods in Pascal. If the implementation is external, this rule is relaxed because Objective-C and Java do not have any `override`-style keyword (since it's the default and only behaviour in these languages), which makes it hard for automated header conversion tools to include it everywhere. The type in which the inherited method is defined is explicitly mentioned, because this may either be an `objcclass` or an `objccategory` in case of Objective-C.

Error: Message name "arg1" in inherited class is different from message name "arg2" in current class.

An overriding Objective-C method cannot have a different message name than an inherited method. The reason is that these message names uniquely define the message to the Objective-C runtime, which means that giving them a different message name breaks the “override” semantics.

Error: It is not yet possible to make unique copies of Objective-C or Java types

Duplicating an Objective-C or Java type using `type x = type y;` is not yet supported. You may be able to obtain the desired effect using `type x = objcclass(y) end;` resp. `type x = class(y) end;` instead.

Error: Objective-C categories and Object Pascal class helpers cannot be used as types

It is not possible to declare a variable as an instance of an Objective-C category or an Object Pascal class helper. A category/class helper adds methods to the scope of an existing class, but does not define a type by itself. An exception of this rule is when inheriting an Object Pascal class helper from another class helper.

Error: Categories do not override, but replace methods. Use "reintroduce" instead.**Error: Replaced methods can only be reintroduced in Objective-C, add "reintroduce" (replaced method defined in arg1).****Hint: Replaced methods can only be reintroduced in Objective-C, add "reintroduce" (replaced method defined in arg1).**

A category replaces an existing method in an Objective-C class, rather than that it overrides it. Calling an inherited method from an category method will call that method in the extended class' parent, not in the extended class itself. The replaced method in the original class is basically lost, and can no longer be called or referred to. This behaviour corresponds somewhat more closely to `reintroduce` than to `override` (although in case of `reintroduce`

in Object Pascal, hidden methods are still reachable via inherited). The type in which the inherited method is defined is explicitly mentioned, because this may either be an `objcclass` or an `objccategory`.

Error: Getter for implements interface must use the target's default calling convention.

Interface getters are called via a helper in the run time library, and hence have to use the default calling convention for the target (`register` on i386 and x86_64, `stdcall` on other architectures).

Error: Typed files cannot contain reference-counted types.

The data in a typed file cannot be of a reference counted type (such as `ansistring` or a record containing a field that is reference counted).

Error: Operator is not overloaded: arg2 "arg1"

You are trying to use an overloaded operator when it is not overloaded for this type.

Error: Operator is not overloaded: "arg1" arg2 "arg3"

You are trying to use an overloaded operator when it is not overloaded for this type.

Error: Expected another arg1 array elements

When declaring a typed constant array, you provided too few elements to initialize the array

Error: String constant too long while ansistrings are disabled

Only when a piece of code is compiled with ansistrings enabled (`{ $H+ }`), string constants longer than 255 characters are allowed.

Error: Type cannot be used as univ parameter because its size is unknown at compile time: "arg1"

`univ` parameters are compatible with all values of the same size, but this cannot be checked in case a parameter's size is unknown at compile time.

Error: Only one class constructor can be declared in class: "arg1"

You are trying to declare more than one class constructor but only one class constructor can be declared.

Error: Only one class destructor can be declared in class: "arg1"

You are trying to declare more than one class destructor but only one class destructor can be declared.

Error: Class constructors cannot have parameters

You are declaring a class constructor with a parameter list. Class constructor methods cannot have parameters.

Error: Class destructors cannot have parameters

You are declaring a class destructor with a parameter list. Class destructor methods cannot have parameters.

Fatal: This construct requires the `{ $modeswitch objectivec1 }` mode switch to be active

Objective-Pascal constructs are not supported when `{ $modeswitch ObjectiveC1 }` is not active.

Error: Unicodechar/string constants cannot be converted to ansi/shortstring at compile-time

It is not possible to use `unicodechar` and `unicodestring` constants in constant expressions that have to be converted into an `ansistring` or `shortstring` at compile time, for example inside typed constants. The reason is that the compiler cannot know what the actual ansi encoding will be at run time.

Error: For-in Objective-Pascal loops require `{ $modeswitch ObjectiveC2 }` to be active

Objective-C “fast enumeration” support was added in Objective-C 2.0, and hence the appropriate `modeswitch` has to be activated to expose this feature. Note that Objective-C 2.0 programs require Mac OS X 10.5 or later.

Error: The compiler cannot find the NSFastEnumerationProtocol or NSFastEnumerationState type in the CocoaAll unit

Objective-C for-in loops (fast enumeration) require that the compiler can find a unit called CocoaAll that contains definitions for the NSFastEnumerationProtocol and NSFastEnumerationState types. If you get this error, most likely the compiler is finding and loading an alternate CocoaAll unit.

Error: Typed constants of the type 'procedure is nested' can only be initialized with NIL and global procedures/functions

A nested procedural variable consists of two components: the address of the procedure/function to call (which is always known at compile time), and also a parent frame pointer (which is never known at compile time) in case the procedural variable contains a reference to a nested procedure/function. Therefore such typed constants can only be initialized with global functions/procedures since these do not require a parent frame pointer.

Fatal: Declaration of generic inside another generic is not allowed

At the moment, scanner supports recording of only one token buffer at the time (guarded by internal error 200511173 in tscannerfile.startrecordtokens). Since generics are implemented by recording tokens, it is not possible to have declaration of a generic (type or method) inside another generic.

Error: Forward declaration "arg1" must be resolved before a class can conform to or implement it

An Objective-C protocol or Java Interface must be fully defined before classes can conform to it. This error occurs in the following situation (example for Objective-C, but the same goes for Java interfaces):

```
Type MyProtocol = objcprotoocl;  
  ChildClass = Class(NSObject, MyProtocol)  
  ...  
end;
```

where MyProtocol is declared but not defined.

Error: Record types cannot have published sections

Published sections can be used only inside classes.

Error: Destructors are not allowed in records or helpers

Destructor declarations are not allowed in records or helpers.

Error: Class methods must be static in records

Class methods declarations are not allowed in records without static modifier. Records have no inheritance and therefore non static class methods have no sense for them.

Error: Parameterless constructors are not allowed in records or record/type helpers

Constructor declarations with no arguments are not allowed in records or record/type helpers.

Error: Either the result or at least one parameter must be of type "arg1"

It is required that either the result of the routine or at least one of its parameters be of the specified type. For example class operators either take an instance of the structured type in which they are defined, or they return one.

Error: Type parameters may require initialization/finalization - cannot be used in variant records

Type parameters may be specialized with types which (e.g. `ansistring`) need initialization/finalization code which is implicitly generated by the compiler.

Error: Variables being declared as external cannot be in a custom section

A section directive is not valid for variables being declared as external.

Error: Non-static and non-global variables cannot have a section directive

A variable placed in a custom section is always statically allocated so it must be either a static or global variable.

Error: "arg1" is not allowed in helper types

Some directives and specifiers like "virtual", "dynamic", "override" are not allowed inside helper types in mode ObjFPC (they are ignored in mode Delphi), because they have no meaning within helpers. Also "abstract" isn't allowed in either mode.

Error: Class constructors are not allowed in helpers

Class constructor declarations are not allowed in helpers.

Error: The use of "inherited" is not allowed in a record

As records don't support inheritance the use of "inherited" is prohibited for these as well as for record helpers (in mode "Delphi" only).

Error: Type declarations are not allowed in local or anonymous records

Records with types must be defined globally. Types cannot be defined inside records which are defined in a procedure or function or in anonymous records.

Error: Duplicate implements clause for interface "arg1"

A class may delegate an interface using the "implements" clause only to a single property. Delegating it multiple times is an error.

Error: Interface "arg1" cannot be delegated by "arg2", it already has method resolutions

Method resolution clause maps a method of an interface to a method of the current class. Therefore the current class has to implement the interface directly. Delegation is not possible.

Error: Interface "arg1" cannot have method resolutions, "arg2" already delegates it

Method resolution is only possible for interfaces that are implemented directly, not by delegation.

Error: Invalid codepage

When declaring a string with a given codepage, the range of valid codepages values is limited to 0 to 65535.

Error: Only fields (var-sections) and constants can be final in object types

A final (class) field must be assigned a single value in the (class) constructor, and cannot be overwritten afterwards. A final (typed) constant is read-only.

Error: Final fields are currently only supported for external classes

Support for final fields in non-external classes requires a full data flow analysis implementation in FPC, which it currently still lacks.

Error: Typed constants are not allowed here, only formal constants are

Java interfaces define a namespace in which formal constant can be defined, but since they define no storage it is not possible to define typed constants in them (those are more or less the same as initialised class fields).

Error: Constructors are not automatically inherited in the JVM; explicitly add a constructor that calls the inherited one if you need it

Java does not automatically add inherited constructors to child classes, so that they can be hidden. For compatibility with external Java code, FPC does the same. If you require access to the same constructors in a child class, define them in the child class and call the inherited one from there.

Parsing internally generated code: arg1

The compiler sometimes internally constructs Pascal code that is subsequently injected into the program. These messages display such code, in order to help with debugging errors in them.

Error: This language feature is not supported on managed VM targets

Certain language features are not supported on targets that are managed virtual machines.

Error: Calling a virtual constructor for the current instance inside another constructor is not possible on the JVM target

The JVM does not natively support virtual constructor. Unfortunately, we are not aware of a way to emulate them in a way that makes it possible to support calling virtual constructors for the current instance inside another constructor.

Error: Overriding method "arg1" cannot have a lower visibility (arg2) than in parent class arg3 (arg4)

The JVM does not allow lowering the visibility of an overriding method.

Error: Procedure/Function declared with call option NOSTACKFRAME but without ASSEMBLER

nostackframe call modifier is supposed to be used in conjunction with assembler.

Error: Procedure/Function declared with call option NOSTACKFRAME but local stack size is arg1

nostackframe call modifier used without assembler modifier might still generate local stack needs.

Error: Cannot generate property getter/setter arg1 because its name clashes with existing identifier arg2

Automatically generated getters/setters cannot have the same name as existing identifiers, because this may change the behaviour of existing code.

Warning: Automatically generated property getter/setter arg1 overrides the same-named getter/setter in class arg2

Automatically generated property getters/setters on the JVM platform are virtual methods, because the JVM does not support non-virtual methods that can be changed in child classes. This means that if a child class changes an inherited property definition, the behaviour of that property can change compared to native targets since even if a variable is declared as the parent type, by calling the virtual method the getter from the child will be used. This is different from the behaviour on native targets or when not activating automatically generated setters/getters, because in that case only the declared type of a variable influences the property behaviour.

Warning: Case mismatch between declared property getter/setter arg1 and automatically constructed name arg2, not changing declared name

If a property's specified getter/setter already corresponded to the naming convention specified by the automatic getter/setter generation setting except in terms of upper/lowercase, the compiler will print a warning because it cannot necessarily change that other declaration itself not can it add one using the correct case (it could conflict with the original declaration). Manually correct the case of the getter/setter to conform to the desired coding rules. `TChild` overrides

Error: Constants declarations are not allowed in local or anonymous records

Records with constants must be defined globally. Constants cannot be defined inside records which are defined in a procedure or function or in anonymous records.

Error: Method declarations are not allowed in local or anonymous records

Records with methods must be defined globally. Methods cannot be defined inside records which are defined in a procedure or function or in anonymous records.

Error: Property declarations are not allowed in local or anonymous records

Records with properties must be defined globally. Properties cannot be defined inside records which are defined in a procedure or function or in anonymous records.

Error: Class member declarations are not allowed in local or anonymous records

Records with class members must be defined globally. Class members cannot be defined inside records which are defined in a procedure or function or in anonymous records.

Error: Visibility section "arg1" not allowed in records

The visibility sections (protected) and (strict protected) are only useful together with inheritance. Since records do not support that they are forbidden.

Error: Directive "arg1" not allowed here

This directive is not allowed in the given context. E.g. "static" is not allowed for instance methods or class operators.

Error: Assembler blocks not allowed inside generics

The use of assembler blocks/routines is not allowed inside generics.

Error: Properties can be only static, global or inside structured types

Properties cannot be declared local, only global, using the static directive or inside structured types.

Error: Overloaded routines have the same mangled name

Some platforms, such as the JVM platform, encode the parameters in the routine name in a prescribed way, and this encoding may map different Pascal types to the same encoded (a.k.a. "mangled") name. This error can only be solved by removing or changing the conflicting definitions' parameter declarations or routine names.

Error: Default values can only be specified for value, const and constref parameters

A default parameter value allows you to not specify a value for this parameter when calling the routine, and the compiler will instead pass the specified default (constant) value. As a result, default values can only be specified for parameters that can accept constant values.

Warning: Pointer type "arg1" ignored

The specified pointer type modifier is ignored, because it is not supported on the current platform. This happens, for example, when a far pointer is declared on a non-x86 platform.

Error: Global Generic template references static symtable

A generic declared in the interface section of a unit must not reference symbols that belong solely to the implementation section of that unit.

Unit arg1 has been already compiled meanwhile.

This tells you that the recursive reading of the uses clauses triggered already a compilation of the current unit, so the current compilation can be aborted.

Error: Explicit implementation of methods for specializations of generics is not allowed

Methods introduced in a generic must be implemented for the generic. It is not possible to implement them only for specializations.

Error: Generic methods are not allowed in interfaces

Generic methods are not allowed in interfaces, because there is no way to specialize a suitable implementation.

Error: Generic methods can not be virtual

Generic methods can not be declared as virtual as there'd need to be a VMT entry for each specialization. This is however not possible with a static VMT.

Error: Dynamic packages not supported for target OS

Support for dynamic packages is not implemented for the specified target OS or it is at least not tested and thus disabled.

Error: The HardFloat directive cannot be used if soft float code is generated or fpu emulation is turned on

The `HardFloat` directive can only be used if an instruction set is used which supports floating point operations.

Error: Index arg1 is not a valid internal function index

The index specified for the `compilerproc` directive is not an index that's recognized by the compiler.

Warning: Operator overload hidden by internal operator: "arg1" arg2 "arg3"

An operator overload is defined for the specified overload, but the internal overload by the compiler takes precedence. This only happens for operators that had been overloadable before (e.g. `dynamic array + dynamic array`), but aren't anymore due to an internal operator being defined while this behavior is controllable by a modeswitch (in case of dynamic arrays that is the modeswitch `ArrayOperators`).

Error: Thread variables inside classes or records must be class variables

A `threadvar` section inside a class or record was started without it being prefixed by `class`.

Error: Only static methods and static variables can be referenced through an object type

This error occurs in a situation like the following:

```
Type
  TObj = object
    procedure test;
  end;

begin
  TObj.test;
```

`test` is not a static method and hence cannot be called through a type, but only using an instance.

C.4 Type checking errors

This section lists all errors that can occur when type checking is performed.

Error: Type mismatch

This can happen in many cases:

- The variable you're assigning to is of a different type than the expression in the assignment.
- You are calling a function or procedure with parameters that are incompatible with the parameters in the function or procedure definition.

Error: Incompatible types: got "arg1" expected "arg2"

There is no conversion possible between the two types. Another possibility is that they are declared in different declarations:

```
Var
  A1 : Array[1..10] Of Integer;
  A2 : Array[1..10] Of Integer;

Begin
  A1:=A2; { This statement also gives this error. It
           is due to the strict type checking of Pascal }
End.
```

Error: Type mismatch between "arg1" and "arg2"

The types are not equal.

Error: Type identifier expected

The identifier is not a type, or you forgot to supply a type identifier.

Error: Variable identifier expected

This happens when you pass a constant to a routine (such as `Inc var` or `Dec`) when it expects a variable. You can only pass variables as arguments to these functions.

Error: Integer expression expected, but got "arg1"

The compiler expects an expression of type integer, but gets a different type.

Error: Boolean expression expected, but got "arg1"

The expression must be a boolean type. It should be `return True` or `False`.

Error: Ordinal expression expected

The expression must be of ordinal type, i.e., maximum a `Longint`. This happens, for instance, when you specify a second argument to `Inc` or `Dec` that doesn't evaluate to an ordinal value.

Error: pointer type expected, but got "arg1"

The variable or expression isn't of the type `pointer`. This happens when you pass a variable that isn't a pointer to `New` or `Dispose`.

Error: class type expected, but got "arg1"

The variable or expression isn't of the type `class`. This happens typically when

1. The parent class in a class declaration isn't a class.
2. An exception handler (`On`) contains a type identifier that isn't a class.

Error: Can't evaluate constant expression

This error can occur when the bounds of an array you declared do not evaluate to ordinal constants.

Error: Set elements are not compatible

You are trying to perform an operation on two sets, when the set element types are not the same. The base type of a set must be the same when taking the union.

Error: Operation not implemented for sets

several binary operations are not defined for sets. These include: `div`, `mod`, `**`, `>=` and `<=`. The last two may be defined for sets in the future.

Warning: Automatic type conversion from floating type to COMP which is an integer type

An implicit type conversion from a real type to a `comp` is encountered. Since `comp` is a 64 bit integer type, this may indicate an error.

Hint: use DIV instead to get an integer result

When hints are on, then an integer division with the `'/'` operator will produce this message, because the result will then be of type `real`.

Error: String types have to match exactly in \$V+ mode

When compiling in `{ $V+ }` mode, the string you pass as a parameter should be of the exact same type as the declared parameter of the procedure.

Error: succ or pred on enums with assignments not possible

If you declare an enumeration type which has C-like assignments in it, such as in the following:

```
Tenum = (a,b,e:=5);
```

then you cannot use the `Succ` or `Pred` functions with this enumeration.

Error: Can't read or write variables of this type

You are trying to `read` or `write` a variable from or to a file of type `text`, which doesn't support that variable's type. Only integer types, reals, `pchars` and strings can be read from or written to a text file. Booleans can only be written to text files.

Error: Can't use readln or writeln on typed file

`readln` and `writeln` are only allowed for text files.

Error: Can't use read or write on untyped file.

`read` and `write` are only allowed for text or typed files.

Error: Type conflict between set elements

There is at least one set element which is of the wrong type, i.e. not of the set type.

Warning: lo/hi(dword/qword) returns the upper/lower word/dword

Free Pascal supports an overloaded version of `lo/hi` for `longint/dword/int64/qword` which returns the lower/upper word/dword of the argument. Turbo Pascal always uses a 16 bit `lo/hi` which always returns bits 0..7 for `lo` and the bits 8..15 for `hi`. If you want the Turbo Pascal behavior you have to type cast the argument to a `word` or `integer`.

Error: Integer or real expression expected

The first argument to `str` must be a real or integer type.

Error: Wrong type "arg1" in array constructor

You are trying to use a type in an array constructor which is not allowed.

Error: Incompatible type for arg no. arg1: Got "arg2", expected "arg3"

You are trying to pass an invalid type for the specified parameter.

Error: Method (variable) and Procedure (variable) are not compatible

You cannot assign a method to a procedure variable or a procedure to a method pointer.

Error: Illegal constant passed to internal math function

The constant argument passed to a `ln` or `sqrt` function is out of the definition range of these functions.

Error: Can't take the address of constant expressions

It is not possible to get the address of a constant expression, because they are not stored in memory. You can try making it a typed constant. This error can also be displayed if you try to pass a property to a var parameter.

Error: Argument cannot be assigned to

Only expressions which can be on the left side of an assignment can be passed as call by reference arguments.

Remark: Properties can be used on the left side of an assignment, nevertheless they cannot be used as arguments.

Error: Can't assign local procedure/function to procedure variable

It's not allowed to assign a local procedure/function to a procedure variable, because the calling convention of a local procedure/function is different. You can only assign local procedure/function to a void pointer.

Error: Can't assign values to an address

It is not allowed to assign a value to an address of a variable, constant, procedure or function. You can try compiling with -So if the identifier is a procedure variable.

Error: Can't assign values to const variable

It's not allowed to assign a value to a variable which is declared as a const. This is normally a parameter declared as const. To allow changing the value, pass the parameter by value, or a parameter by reference (using var).

Error: Array type required

If you are accessing a variable using an index '[<x>]' then the type must be an array. In FPC mode a pointer is also allowed.

Error: interface type expected, but got "arg1"

The compiler expected to encounter an interface type name, but got something else. The following code would produce this error:

```
Type
  TMyStream = Class(TStream, Integer)
```

Hint: Mixing signed expressions and longwords gives a 64bit result

If you divide (or calculate the modulus of) a signed expression by a longword (or vice versa), or if you have overflow and/or range checking turned on and use an arithmetic expression (+, -, *, div, mod) in which both signed numbers and longwords appear, then everything has to be evaluated in 64-bit arithmetic which is slower than normal 32-bit arithmetic. You can avoid this by typecasting one operand so it matches the result type of the other one.

Warning: Mixing signed expressions and cardinals here may cause a range check error

If you use a binary operator (and, or, xor) and one of the operands is a longword while the other one is a signed expression, then, if range checking is turned on, you may get a range check error because in such a case both operands are converted to longword before the operation is carried out. You can avoid this by typecasting one operand so it matches the result type of the other one.

Error: Typecast has different size (arg1 -> arg2) in assignment

Type casting to a type with a different size is not allowed when the variable is used in an assignment.

Error: enums with assignments cannot be used as array index

When you declared an enumeration type which has C-like assignments, such as in the following:

```
Tenum = (a,b,e:=5);
```

you cannot use it as the index of an array.

Error: Class or Object types "arg1" and "arg2" are not related

There is a typecast from one class or object to another while the class/object are not related. This will probably lead to errors.

Warning: Class types "arg1" and "arg2" are not related

There is a typecast from one class to another while the classes are not related. This will probably lead to errors.

Error: Class or interface type expected, but got "arg1"

The compiler expected a class or interface name, but got another type or identifier.

Error: Type "arg1" is not completely defined

This error occurs when a type is not complete: i.e. a pointer type which points to an undefined type.

Warning: String literal has more characters than short string length

The size of the constant string, which is assigned to a shortstring, is longer than the maximum size of the shortstring (255 characters).

Warning: Comparison might be always false due to range of constant and expression

There is a comparison between a constant and an expression where the constant is out of the valid range of values of the expression. Because of type promotion, the statement will always evaluate to false. Explicitly typecast the constant or the expression to the correct range to avoid this warning if you think the code is correct.

Warning: Comparison might be always true due to range of constant and expression

There is a comparison between a constant and an expression where the constant is out of the valid range of values of the expression. Because of type promotion, the statement will always evaluate to true. Explicitly typecast the constant or the expression to the correct range to avoid this warning if you think the code is correct.

Warning: Constructing a class "arg1" with abstract method "arg2"

An instance of a class is created which contains non-implemented abstract methods. This will probably lead to a runtime error 211 in the code if that routine is ever called. All abstract methods should be overridden.

Hint: The left operand of the IN operator should be byte sized

The left operand of the `in` operator is not an ordinal or enumeration which fits within 8 bits. This may lead to range check errors. The `in` operator currently only supports a left operand which fits within a byte. In the case of enumerations, the size of an element of an enumeration can be controlled with the `{ $PACKENUM }` or `{ $Zn }` switches.

Warning: Type size mismatch, possible loss of data / range check error

There is an assignment to a smaller type than the source type. This means that this may cause a range-check error, or may lead to possible loss of data.

Hint: Type size mismatch, possible loss of data / range check error

There is an assignment to a smaller type than the source type. This means that this may cause a range-check error, or may lead to possible loss of data.

Error: The address of an abstract method cannot be taken

An abstract method has no body, so the address of an abstract method cannot be taken.

Error: Assignments to formal parameters and open arrays are not possible

You are trying to assign a value to a formal (untyped var, const or out) parameter, or to an open array.

Error: Constant Expression expected

The compiler expects an constant expression, but gets a variable expression.

Error: Operation "arg1" not supported for types "arg2" and "arg3"

The operation is not allowed for the supplied types.

Error: Illegal type conversion: "arg1" to "arg2"

When doing a type-cast, you must take care that the sizes of the variable and the destination type are the same.

Hint: Conversion between ordinals and pointers is not portable

If you typecast a pointer to a longint (or vice-versa), this code will not compile on a machine using 64 bits addressing.

Warning: Conversion between ordinals and pointers is not portable

If you typecast a pointer to an ordinal type of a different size (or vice-versa), this can cause problems. This is a warning to help in finding the 32-bit specific code where cardinal/longint is used to typecast pointers to ordinals. A solution is to use the pprint/ptruint types instead.

Error: Can't determine which overloaded function to call

You're calling overloaded functions with a parameter that doesn't correspond to any of the declared function parameter lists. e.g. when you have declared a function with parameters `word` and `longint`, and then you call it with a parameter which is of type `integer`.

Error: Illegal counter variable

The type of a `for` loop variable must be an ordinal type. Loop variables cannot be reals or strings.

Warning: Converting constant real value to double for C variable argument, add explicit type-cast to prevent this.

In C, constant real values are double by default. For this reason, if you pass a constant real value to a variable argument part of a C function, FPC by default converts this constant to double as well. If you want to prevent this from happening, add an explicit typecast around the constant.

Error: Class or COM interface type expected, but got "arg1"

Some operators, such as the `AS` operator, are only applicable to classes or COM interfaces.

Error: Constant packed arrays are not yet supported

You cannot declare a (bit)packed array as a typed constant.

Error: Incompatible type for arg no. arg1: Got "arg2" expected "(Bit)Packed Array"

The compiler expects a (bit)packed array as the specified parameter.

Error: Incompatible type for arg no. arg1: Got "arg2" expected "(not packed) Array"

The compiler expects a regular (i.e., not packed) array as the specified parameter.

Error: Elements of packed arrays cannot be of a type which need to be initialised

Support for packed arrays of types that need initialization (such as `ansistrings`, or records which contain `ansistrings`) is not yet implemented.

Error: Constant packed records and objects are not yet supported

You cannot declare a (bit)packed array as a typed constant at this time.

Warning: Arithmetic "arg1" on untyped pointer is unportable to {T+}, suggest typecast

Addition/subtraction from an untyped pointer may work differently in `{T+}`. Use a typecast to a typed pointer.

Error: Can't take address of a subroutine marked as local

The address of a subroutine marked as local cannot be taken.

Error: Can't export subroutine marked as local from a unit

A subroutine marked as local cannot be exported from a unit.

Error: Type is not automatable: "arg1"

Only byte, integer, longint, smallint, currency, single, double, ansistring, widestring, tdatetime, variant, olevariant, wordbool and all interfaces are automatable.

Hint: Converting the operands to "arg1" before doing the add could prevent overflow errors.

Adding two types can cause overflow errors. Since you are converting the result to a larger type, you could prevent such errors by converting the operands to this type before doing the addition.

Hint: Converting the operands to "arg1" before doing the subtract could prevent overflow errors.

Subtracting two types can cause overflow errors. Since you are converting the result to a larger type, you could prevent such errors by converting the operands to this type before doing the subtraction.

Hint: Converting the operands to "arg1" before doing the multiply could prevent overflow errors.

Multiplying two types can cause overflow errors. Since you are converting the result to a larger type, you could prevent such errors by converting the operands to this type before doing the multiplication.

Warning: Converting pointers to signed integers may result in wrong comparison results and range errors, use an unsigned type instead.

The virtual address space on 32-bit machines runs from \$00000000 to \$fffffff. Many operating systems allow you to allocate memory above \$80000000. For example both WINDOWS and LINUX allow pointers in the range \$00000000 to \$bfffffff. If you convert pointers to signed types, this can cause overflow and range check errors, but also \$80000000 < \$7fffffff. This can cause random errors in code like "if p>q".

Error: Interface type arg1 has no valid GUID

When applying the as-operator to an interface or class, the desired interface (i.e. the right operand of the as-operator) must have a valid GUID.

Error: Invalid selector name "arg1"

An Objective-C selector cannot be empty, must be a valid identifier or a single colon, and if it contains at least one colon it must also end in one.

Error: Expected Objective-C method, but got arg1

A selector can only be created for Objective-C methods, not for any other kind of procedure/function/method.

Error: Expected Objective-C method or constant method name

A selector can only be created for Objective-C methods, either by specifying the name using a string constant, or by using an Objective-C method identifier that is visible in the current scope.

Error: No type info available for this type

Type information is not generated for some types, such as enumerations with gaps in their value range (this includes enumerations whose lower bound is different from zero).

Error: Ordinal or string expression expected

The expression must be an ordinal or string type.

Error: String expression expected

The expression must be a string type.

Warning: Converting 0 to NIL

Use NIL rather than 0 when initialising a pointer.

Error: Objective-C protocol type expected, but got "arg1"

The compiler expected a protocol type name, but found something else.

Error: The type "arg1" is not supported for interaction with the Objective-C and the blocks runtime.

Objective-C and Blocks make extensive use of run time type information (RTTI). This format is defined by the maintainers of the run time and can therefore not be adapted to all possible Object Pascal types. In particular, types that depend on reference counting by the compiler (such as ansistrings and certain kinds of interfaces) cannot be used as fields of Objective-C classes, cannot be directly passed to Objective-C methods or Blocks, and cannot be encoded using `objc_encode`.

Error: Class or objcclass type expected, but got "arg1"

It is only possible to create class reference types of `class` and `objcclass`

Error: Objcclass type expected

The compiler expected an `objcclass` type

Warning: Coerced univ parameter type in procedural variable may cause crash or memory corruption: arg1 to arg2

`univ` parameters are implicitly compatible with all types of the same size, also in procedural variable definitions. That means that the following code is legal, because `single` and `longint` have the same size:

```
{ $mode macpas }
Type
  TIntProc = procedure (l: univ longint);

  procedure test(s: single);
  begin
    writeln(s);
  end;

var
  p: TIntProc;
begin
  p:=test;
  p(4);
end.
```

This code may however crash on platforms that pass integers in registers and floating point values on the stack, because then the stack will be unbalanced. Note that this warning will not flag all potentially dangerous situations. when `test` returns.

Error: Type parameters of specializations of generics cannot reference the currently specialized type

Recursive specializations of generics like `Type MyType = specialize MyGeneric<MyType>;` are not possible.

Error: Type parameters are not allowed on non-generic class/record/object procedure or function

Type parameters are only allowed for methods of generic classes, records or objects

Error: Generic declaration of "arg1" differs from previous declaration

Generic declaration does not match the previous declaration

Error: Helper type expected

The compiler expected a `class helper` type.

Error: Record type expected

The compiler expected a `record` type.

Error: Derived class helper must extend a subclass of "arg1" or the class itself

If a class helper inherits from another class helper the extended class must extend either the same class as the parent class helper or a subclass of it

Error: Derived record or type helper must extend "arg1"

If a record helper inherits from another record helper it must extend the same record that the parent record helper extended.

Error: Invalid assignment, procedures return no value

This error occurs when one tries to assign the result of a procedure or destructor call. A procedure or destructor returns no value so this is not possible.

Warning: Implicit string type conversion from "arg1" to "arg2"

An implicit type conversion from an ansi string type to an unicode string type is encountered. To avoid this warning perform an explicit type conversion.

Warning: Implicit string type conversion with potential data loss from "arg1" to "arg2"

An implicit type conversion from an unicode string type to an ansi string type is encountered. This conversion can lose data since not all unicode characters may be represented in the codepage of destination string type.

Warning: Explicit string typecast from "arg1" to "arg2"

An explicit typecast from an ansi string type to an unicode string type is encountered. This warning is off by default. You can turn it on to see all suspicious string conversions.

Warning: Explicit string typecast with potential data loss from "arg1" to "arg2"

An explicit typecast from an unicode string type to an ansi string type is encountered. This conversion can lose data since not all unicode characters may be represented in the codepage of destination string type. This warning is off by default. You can turn it on to see all the places with lossy string conversions.

Warning: Unicode constant cast with potential data loss

Conversion from a `WideChar` to `AnsiChar` can lose data since now all unicode characters may be represented in the current system codepage You can nest function definitions only 31 levels deep.

Error: range check error while evaluating constants (arg1 must be between arg2 and arg3)**Warning: range check error while evaluating constants (arg1 must be between arg2 and arg3)**

The constants are outside their allowed range.

Error: This type is not supported for the Default() intrinsic

Some types like for example `Text` and `File Of X` are not supported by the `Default` intrinsic.

Error: JVM virtual class methods cannot be static

Virtual class methods cannot be static when targeting the JVM platform, because the self pointer is required for correct dispatching.

Error: Final (class) fields can only be assigned in their class' (class) constructor

It is only possible to assign a value to a final (class) field inside a (class) constructor of its owning class.

Error: It is not possible to typecast untyped parameters on managed platforms, simply assign a value to them instead.

On managed platforms, untyped parameters are translated by the compiler into the equivalent of `var x: BaseClassType`. Non-class-based types passed to such parameters are automatically wrapped (or boxed) in a class, and after the call the potentially modified value is assigned back to the original variable. On the caller side, changing untyped var/out parameters happens by simply assigning values to them (either class-based or primitive ones). On the caller side, they will be extracted and if their type does not match the original variable's, an exception will be raised.

Error: The assignment side of an expression cannot be typecasted to a supertype on managed platforms

Managed platforms guarantee type safety at the bytecode level. This means that the virtual machine must be able to statically determine that no type-unsafe assignments or operations occur. By assigning a parent class type to a variable of a child type by typecasting the assignment side to the parent class type, the type safety would no longer be guaranteed and the generated code would fail verification at run time time.

Warning: The interface method "arg1" raises the visibility of "arg2" to public when accessed via an interface instance**Error: The interface method "arg1" has a higher visibility (public) than "arg2"**

All methods in an interface have always public visibility. That means that if an interface method is implemented using a (strict) protected or private method, this method is actually publicly accessible via the interface. On the JVM target this situation results in an error because the JVM rejects such attempts to circumvent the visibility rules. On other targets this is a warning that is disabled by default because such situations are common practice, but it can be enabled in case you are concerned with keeping your code compilable for the JVM target.

Error: TYPEOF can only be used on object types with VMT

Typeof() intrinsic returns pointer to VMT of its argument. It cannot be used on object types that do not have VMT.

Error: It is not possible to define a default value for a parameter of type "arg1"

Parameters declared as structured types, such as files, variants, non-dynamic arrays and TP-style objects, cannot have a default value.

Error: Type "arg1" cannot be extended by a type helper

Types like procedural variables cannot be extended by type helpers

Error: Procedure or function must be far in order to allow taking its address: "arg1"

In certain i8086 memory models (medium, large and huge), procedures and functions have to be declared 'far' in order to allow their address to be taken.

Warning: Creating an instance of abstract class "arg1"

The specified class is declared as `abstract` and thus no instance of this class should be created. This is merely a warning for Delphi compatibility.

Error: Subroutine references cannot be declared as "of object" or "is nested", they can always refer to any kind of subroutine

Subroutine references can refer to any kind of subroutine and hence do not require specialisation for methods or nested subroutines.

Error: Procedure variables in that memory model do not store segment information

Warning: The first value of a set constructor range is greater than the second value, so the range describes an empty set.

If a set is constructed like this: `s := [9..7];`, then an empty set is generated. As this is something normally not desired, the compiler warns about it.

Error: C block reference must use CDECL or MWPASCAL calling convention.

When declaring a C block reference ensure that it uses either the `cdecl` or `mwpascal` calling convention either by adding the corresponding function directive or by using the `{SCalling}` compiler directive.

C.5 Symbol handling

This section lists all the messages that concern the handling of symbols. This means all things that have to do with procedure and variable names.

Error: Identifier not found "arg1"

The compiler doesn't know this symbol. Usually happens when you misspell the name of a variable or procedure, or when you forget to declare a variable.

Fatal: Internal Error in SymTableStack()

An internal error occurred in the compiler; If you encounter such an error, please contact the developers and try to provide an exact description of the circumstances in which the error occurs.

Error: Duplicate identifier "arg1"

The identifier was already declared in the current scope.

Hint: Identifier already defined in arg1 at line arg2

The identifier was already declared in a previous scope.

Error: Unknown identifier "arg1"

The identifier encountered has not been declared, or is used outside the scope where it is defined.

Error: Forward declaration not solved "arg1"

This can happen in two cases:

- You declare a function in the `interface` part, or with a `forward` directive, but do not implement it.
- You reference a type which isn't declared in the current `type` block.

Error: Error in type definition

There is an error in your definition of a new array type. One of the range delimiters in an array declaration is erroneous. For example, `Array [1..1.25]` will trigger this error.

Error: Forward type not resolved "arg1"

A symbol was forward defined, but no declaration was encountered.

Error: Only static variables can be used in static methods or outside methods

A static method of an object can only access static variables.

Error: Record or object or class type expected

The variable or expression isn't of the type `record` or `object` or `class`.

Error: Instances of classes or objects with an abstract method are not allowed

You are trying to generate an instance of a class which has an abstract method that wasn't overridden.

Warning: Label not defined "arg1"

A label was declared, but not defined.

Error: Label used but not defined "arg1"

A label was declared and used, but not defined.

Error: Illegal label declaration

This error should never happen; it occurs if a label is defined outside a procedure or function.

Error: GOTO and LABEL are not supported (use switch -Sg)

You must use the `-Sg` switch to compile a program which has `labels` and `goto` statements. By default, `label` and `goto` are not supported.

Error: Label not found

A `goto label` was encountered, but the label wasn't declared.

Error: identifier isn't a label

The identifier specified after the `goto` isn't of type `label`.

Error: label already defined

You are defining a label twice. You can define a label only once.

Error: illegal type declaration of set elements

The declaration of a set contains an invalid type definition.

Error: Forward class definition not resolved "arg1"

You declared a class, but you did not implement it.

Hint: Unit "arg1" not used in arg2

The unit referenced in the `uses` clause is not used.

Hint: Parameter "arg1" not used

The identifier was declared (locally or globally) but was not used (locally or globally).

Note: Local variable "arg1" not used

You have declared, but not used, a variable in a procedure or function implementation.

Hint: Value parameter "arg1" is assigned but never used

The identifier was declared (locally or globally) and assigned to, but is not used (locally or globally) after the assignment.

Note: Local variable "arg1" is assigned but never used

The variable in a procedure or function implementation is declared and assigned to, but is not used after the assignment.

Hint: Local arg1 "arg2" is not used

A local symbol is never used.

Note: Private field "arg1.arg2" is never used

The indicated private field is defined, but is never used in the code.

Note: Private field "arg1.arg2" is assigned but never used

The indicated private field is declared and assigned to, but never read.

Note: Private method "arg1.arg2" never used

The indicated private method is declared but is never used in the code.

Error: Set type expected

The variable or expression is not of type `set`. This happens in an `in` statement.

Warning: Function result does not seem to be set

You can get this warning if the compiler thinks that a function return value is not set. This will not be displayed for assembler procedures, or procedures that contain assembler blocks.

Warning: Type "arg1" is not aligned correctly in current record for C

Arrays with sizes not multiples of 4 will be wrongly aligned for C structures.

Error: Unknown record field identifier "arg1"

The field doesn't exist in the record/object definition.

Warning: Local variable "arg1" does not seem to be initialized

This message is displayed if the compiler thinks that a variable will be used (i.e. it appears in the right-hand side of an expression) when it was not initialized first (i.e. appeared in the left-hand side of an assignment).

Warning: Variable "arg1" does not seem to be initialized

This message is displayed if the compiler thinks that a variable will be used (i.e. it appears in the right-hand side of an expression) when it was not initialized first (i.e. appeared in the left-hand side of an assignment).

Error: identifier idents no member "arg1"

This error is generated when an identifier of a record, field or method is accessed while it is not defined.

Hint: Found declaration: arg1

You get this when you use the `-vh` switch. In the case of an overloaded procedure not being found. Then all candidate overloaded procedures are listed, with their parameter lists.

Error: Data element too large

You get this when you declare a data element whose size exceeds the prescribed limit (2 Gb on 80386+/68020+ processors).

Error: No matching implementation for interface method "arg1" found

There was no matching method found which could implement the interface method. Check argument types and result type of the methods.

Warning: Symbol "arg1" is deprecated

This means that a symbol (a variable, routine, etc...) which is declared as `deprecated` is used. Deprecated symbols may no longer be available in newer versions of the unit / library. Use of this symbol should be avoided as much as possible.

Warning: Symbol "arg1" is not portable

This means that a symbol (a variable, routine, etc...) which is declared as `platform` is used. This symbol's value, use and availability is platform specific and should not be used if the source code must be portable.

Warning: Symbol "arg1" is not implemented

This means that a symbol (a variable, routine, etc...) which is declared as `unimplemented` is used. This symbol is defined, but is not yet implemented on this specific platform.

Error: Can't create unique type from this type

Only simple types like ordinal, float and string types are supported when redefining a type with `type newtype = type oldtype;`.

Hint: Local variable "arg1" does not seem to be initialized

This message is displayed if the compiler thinks that a variable will be used (i.e. it appears in the right-hand side of an expression) when it was not initialized first (i.e. it did not appear in the left-hand side of an assignment).

Hint: Variable "arg1" does not seem to be initialized

This message is displayed if the compiler thinks that a variable will be used (i.e. it appears in the right-hand side of an expression) when it was not initialized first (i.e. it did not appear in the left-hand side of an assignment).

Warning: Function result variable does not seem to be initialized

This message is displayed if the compiler thinks that the function result variable will be used (i.e. it appears in the right-hand side of an expression) before it is initialized (i.e. before it appeared in the left-hand side of an assignment).

Hint: Function result variable does not seem to be initialized

This message is displayed if the compiler thinks that the function result variable will be used (i.e. it appears in the right-hand side of an expression) before it is initialized (i.e. it appears in the left-hand side of an assignment).

Warning: Variable "arg1" read but nowhere assigned

You have read the value of a variable, but nowhere assigned a value to it.

Hint: Found abstract method: arg1

When getting a warning about constructing a class/object with abstract methods you get this hint to assist you in finding the affected method.

Warning: Symbol "arg1" is experimental

This means that a symbol (a variable, routine, etc...) which is declared as `experimental` is used. Experimental symbols might disappear or change semantics in future versions. Usage of this symbol should be avoided as much as possible.

Warning: Forward declaration "arg1" not resolved, assumed external

This happens if you declare a function in the `interface` of a unit in `macpas` mode, but do not implement it.

Warning: Symbol "arg1" is belongs to a library

This means that a symbol (a variable, routine, etc...) which is declared as `library` is used. Library symbols may not be available in other libraries.

Warning: Symbol "arg1" is deprecated: "arg2"

This means that a symbol (a variable, routine, etc...) which is declared as `deprecated` is used. Deprecated symbols may no longer be available in newer versions of the unit / library. Use of this symbol should be avoided as much as possible.

Error: Cannot find an enumerator for the type "arg1"

This means that compiler cannot find an appropriate enumerator to use in the `for-in` loop. To create an enumerator you need to define an operator `enumerator` or add a `public` or `published` `GetEnumerator` method to the class or object definition.

Error: Cannot find a "MoveNext" method in enumerator "arg1"

This means that compiler cannot find a `public` `MoveNext` method with the `Boolean` return type in the enumerator class or object definition.

Error: Cannot find a "Current" property in enumerator "arg1"

This means that compiler cannot find a public `Current` property in the enumerator class or object definition.

Error: Mismatch between number of declared parameters and number of colons in message string.

In Objective-C, a message name automatically contains as many colons as parameters. In order to prevent mistakes when specifying the message name in FPC, the compiler checks whether this is also the case here. Note that in case of messages taking a variable number of arguments translated to FPC via an `array of const` parameter, this final `array of const` parameter is not counted. Neither are the hidden `self` and `_cmd` parameters.

Note: Private type "arg1.arg2" never used

The indicated private type is declared but is never used in the code.

Note: Private const "arg1.arg2" never used

The indicated private const is declared but is never used in the code.

Note: Private property "arg1.arg2" never used

The indicated private property is declared but is never used in the code.

Warning: Unit "arg1" is deprecated

This means that a unit which is declared as `deprecated` is used. Deprecated units may no longer be available in newer versions of the library. Use of this unit should be avoided as much as possible.

Warning: Unit "arg1" is deprecated: "arg2"

This means that a unit which is declared as `deprecated` is used. Deprecated units may no longer be available in newer versions of the library. Use of this unit should be avoided as much as possible.

Warning: Unit "arg1" is not portable

This means that a unit which is declared as `platform` is used. This unit use and availability is platform specific and should not be used if the source code must be portable.

Warning: Unit "arg1" is belongs to a library

This means that a unit which is declared as `library` is used. Library units may not be available in other libraries.

Warning: Unit "arg1" is not implemented

This means that a unit which is declared as `unimplemented` is used. This unit is defined, but is not yet implemented on this specific platform.

Warning: Unit "arg1" is experimental

This means that a unit which is declared as `experimental` is used. Experimental units might disappear or change semantics in future versions. Usage of this unit should be avoided as much as possible.

Error: No full definition of the formally declared class "arg1" is in scope. Add the unit containing its full definition to the uses clause.

Objective-C and Java classes can be imported formally, without using the unit in which it is fully declared. This enables making forward references to such classes and breaking circular dependencies amongst units. However, as soon as you wish to actually do something with an entity of this class type (such as access one of its fields, send a message to it, or use it to inherit from), the compiler requires the full definition of the class to be in scope.

Error: Gotos into initialization or finalization blocks of units are not allowed

Gotos into initialization or finalization blockse of units are not allowed.

Error: Invalid external name "arg1" for formal class "arg2"**Error: Complete class definition with external name "arg1" here**

When a class is declared using a formal external definition, the actual external definition (if any) must specify the same external name as the formal definition (since both definitions refer to the same actual class type).

Warning: Possible library conflict: symbol "arg1" from library "arg2" also found in library "arg3"

Some OS do not have library specific namespaces, for those OS, the function declared as "external 'libname' name 'funcname'", the 'libname' part is only a hint, funcname might also be loaded by another library. This warning appears if 'funcname' is used twice with two different library names.

Error: Cannot add implicit constructor 'Create' because identifier already used by "arg1"

Java does not automatically add inherited constructors to child classes, so that they can be hidden. However, if a class does not explicitly declare at least one constructor, the compiler is required to add a public, parameterless constructor. In Java, constructors are nameless, but in FPC they are all called "Create". Therefore, if you do not add a constructor to a Java class and furthermore use the "Create" identifier for another entity (e.g., a field, or a parameterless method), the compiler cannot satisfy this requirement.

Error: Cannot generate default constructor for class, because parent has no parameterless constructor

Java does not automatically add inherited constructors to child classes, so that they can be hidden. However, if a class does not explicitly declare at least one constructor, the compiler is required to add a public, parameterless constructor. This compiler must then call the parameterless constructor from the parent class inside this added constructor. This is however impossible if the parent class does not declare such a constructor. In this case you must add a valid constructor yourself.

Adding helper for arg1

A helper for the mentioned type is added to the current scope

Error: Found declaration: arg1

This message shows all overloaded declarations in case of an error.

Warning: Local variable "arg1" of a managed type does not seem to be initialized

This message is displayed if the compiler thinks that a variable will be used (i.e. it appears in the right-hand side of an expression) when it was not initialized first (i.e. appeared in the left-hand side of an assignment). Since the variable is managed, i. e. implicitly initialized by the compiler, this might be intended behaviour and does not necessarily mean that the code is wrong.

Warning: Variable "arg1" of a managed type does not seem to be initialized

This message is displayed if the compiler thinks that a variable will be used (i.e. it appears in the right-hand side of an expression) when it was not initialized first (i.e. appeared in the left-hand side of an assignment). Since the variable is managed, i. e. implicitly initialized by the compiler, this might be intended behaviour and does not necessarily mean that the code is wrong.

Hint: Local variable "arg1" of a managed type does not seem to be initialized

This message is displayed if the compiler thinks that a variable will be used (i.e. it appears in the right-hand side of an expression) when it was not initialized first (i.e. it did not appear in the left-hand side of an assignment). Since the variable is managed, i. e. implicitly initialized by the compiler, this might be intended behaviour and does not necessarily mean that the code is wrong.

Hint: Variable "arg1" of a managed type does not seem to be initialized

This message is displayed if the compiler thinks that a variable will be used (i.e. it appears in the right-hand side of an expression) when it was not initialized first (i.e. it did not appear in the left-hand side of an assignment). Since the variable is managed, i. e. implicitly initialized by the compiler, this might be intended behaviour and does not necessarily mean that the code is wrong.

Warning: function result variable of a managed type does not seem to be initialized

This message is displayed if the compiler thinks that the function result variable will be used (i.e. it appears in the right-hand side of an expression) before it is initialized (i.e. before it appeared in the left-hand side of an assignment). Since the variable is managed, i. e. implicitly initialized by the compiler, this might be intended behaviour and does not necessarily mean that the code is wrong.

Hint: Function result variable of a managed type does not seem to be initialized

This message is displayed if the compiler thinks that the function result variable will be used (i.e. it appears in the right-hand side of an expression) before it is initialized (i.e. it appears in the left-hand side of an assignment). Since the variable is managed, i. e. implicitly initialized by the compiler, this might be intended behaviour and does not necessarily mean that the code is wrong.

Warning: Duplicate identifier "arg1"

The identifier was already declared in an Objective-C category that's in the same scope as the current identifier. This is a warning instead of an error, because while this hides the identifier from the category, there are often many unused categories in scope.

Error: Generic type parameter "arg1" does not match with the one in the declaration

The specified generic type parameter for the generic class, record or routine does not match with the one declared in the declaration of the generic class, record or routine.

Error: Generic type parameter declared as "arg1"

Shows what the generic type parameter was originally declared as if a mismatch is found between a declaration and the definition.

Error: Record or object type expected

The variable or expression isn't of the type `record` or `object`.

C.6 Code generator messages

This section lists all messages that can be displayed if the code generator encounters an error condition.

Error: Parameter list size exceeds 65535 bytes

The I386 processor limits the parameter list to 65535 bytes. (The `RET` instruction causes this.)

Error: File types must be var parameters

You cannot specify files as value parameters, i.e., they must always be declared `var` parameters.

Error: The use of a far pointer isn't allowed there

Free Pascal doesn't support far pointers, so you cannot take the address of an expression which has a far reference as a result. The `mem` construct has a far reference as a result, so the following code will produce this error:

```
var p : pointer;  
...  
p:=@mem[a000:000];
```

Error: EXPORT declared functions cannot be called

No longer in use.

Warning: Possible illegal call of constructor or destructor

The compiler detected that a constructor or destructor is called within a a method. This will probably lead to problems, since constructors / destructors require parameters on entry.

Note: Inefficient code

Your statement seems dubious to the compiler.

Warning: unreachable code

You specified a construct which will never be executed. Example:

```
while false do  
  begin  
    {... code ...}  
  end;
```

Error: Abstract methods cannot be called directly

You cannot call an abstract method directly. Instead, you must call an overriding child method, because an abstract method isn't implemented.

Register arg1 weight arg2 arg3

Debugging message. Shown when the compiler considers a variable for keeping in the registers.

Stack frame is omitted

Some procedure/functions do not need a complete stack-frame, so it is omitted. This message will be displayed when the -vd switch is used.

Error: Object or class methods cannot be inline.

You cannot have inlined object methods.

Error: Procvar calls cannot be inline.

A procedure with a procedural variable call cannot be inlined.

Error: No code for inline procedure stored

The compiler couldn't store code for the inline procedure.

Error: Element zero of an ansi/wide- or longstring cannot be accessed, use (set)length instead

You should use `setlength` to set the length of an ansi/wide/longstring and `length` to get the length of such string type.

Error: Constructors or destructors cannot be called inside a 'with' clause

Inside a `with` clause you cannot call a constructor or destructor for the object you have in the `with` clause.

Error: Cannot call message handler methods directly

A message method handler method cannot be called directly if it contains an explicit `Self` argument.

Error: Jump in or outside of an exception block

It is not allowed to jump in or outside of an exception block like `try..finally..end;`. For example, the following code will produce this error:

```
label 1;

...

try
  if not(final) then
    goto 1;    // this line will cause an error
  finally
    ...
end;
1:
...
```

Error: Control flow statements are not allowed in a finally block

It isn't allowed to use the control flow statements `break`, `continue` and `exit` inside a finally statement. The following example shows the problem:

```
...
try
  p;
finally
  ...
  exit; // This exit ISN'T allowed
end;
...
```

If the procedure `p` raises an exception the finally block is executed. If the execution reaches the `exit`, it's unclear what to do: exit the procedure or search for another exception handler.

Warning: Parameters size exceeds limit for certain cpu's

This indicates that you are declaring more than 64K of parameters, which might not be supported on other processor targets.

Warning: Local variable size exceed limit for certain cpu's

This indicates that you are declaring more than 32K of local variables, which might not be supported on other processor targets.

Error: Local variables size exceeds supported limit

This indicates that you are declaring more than 32K of local variables, which is not supported by this processor.

Error: BREAK not allowed

You're trying to use `break` outside a loop construction.

Error: CONTINUE not allowed

You're trying to use `continue` outside a loop construction.

Fatal: Unknown compilerproc "arg1". Check if you use the correct run time library.

The compiler expects that the runtime library contains certain subroutines. If you see this error

and you didn't change the runtime library code, it's very likely that the runtime library you're using doesn't match the compiler in use. If you changed the runtime library this error means that you removed a subroutine which the compiler needs for internal use.

Fatal: Cannot find system type "arg1". Check if you use the correct run time library.

The compiler expects that the runtime library contains certain type definitions. If you see this error and you didn't change the runtime library code, it's very likely that the runtime library you're using doesn't match the compiler in use. If you changed the runtime library this error means that you removed a type which the compiler needs for internal use.

Hint: Inherited call to abstract method ignored

This message appears only in Delphi mode when you call an abstract method of a parent class via `inherited;`. The call is then ignored.

Error: Goto label "arg1" not defined or optimized away

The label used in the goto definition is not defined or optimized away by the unreachable code elimination.

Fatal: Cannot find type "arg1" in unit "arg2". Check if you use the correct run time library.

The compiler expects that the runtime library contains certain type definitions. If you see this error and you didn't change the runtime library code, it's very likely that the runtime library you're using doesn't match the compiler in use. If you changed the runtime library this error means that you removed a type which the compiler needs for internal use.

Error: Interprocedural gotos are allowed only to outer subroutines

Gotos between subroutines are only allowed if the goto jumps from an inner to an outer subroutine or from a subroutine to the main program

Error: Label must be defined in the same scope as it is declared

In ISO mode, labels must be defined in the same scope as they are declared.

Error: Leaving procedures containing explicit or implicit exceptions frames using goto is not allowed

Non-local gotos might not be used to leave procedures using exceptions either implicitly or explicitly. Procedures which use automated types like ansistrings or class constructors are affected by this too.

Error: In ISO mode, the mod operator is defined only for positive quotient

In ISO pascal, only positive values are allowed for the quotient: $n \bmod m$ is only valid if $m > 0$.

Auto inlining: arg1

Due to auto inlining turned on, the compiler auto inlines this subroutine.

Error: The function used, is not supported by the selected instruction set: arg1

Some functions cannot be implemented efficiently for certain instruction sets, one example is fused multiply/add. To avoid very inefficient code, the compiler complains in this case, so either select another instruction set or replace the function call by alternative code

Fatal: Maximum number of units (arg1) reached for the current target

Depending of target architecture, the number of units is limited. This limit has been reached. A unit counts only if it contains initialization or finalization count.

Note: Call to subroutine "arg1" marked as inline is not inlined

The directive `inline` is only a hint to the compiler. Sometimes the compiler ignores this hint, a subroutine marked as `inline` is not inlined. In this case, this hint is given. Compiling with `-vd` might result in more information why the directive `inline` is ignored.

C.7 Errors of assembling/linking stage

This section lists errors that occur when the compiler is processing the command line or handling the configuration files.

Warning: Source operating system redefined

The source operating system is redefined.

Info: Assembling (pipe) arg1

Assembling using a pipe to an external assembler.

Error: Can't create assembler file: arg1

The mentioned file cannot be created. Check if you have access permissions to create this file.

Error: Can't create object file: arg1 (error code: arg2)

The mentioned file cannot be created. Check if you have got access permissions to create this file.

Error: Can't create archive file: arg1

The mentioned file cannot be created. Check if you have access permissions to create this file.

Error: Assembler arg1 not found, switching to external assembling

The assembler program was not found. The compiler will produce a script that can be used to assemble and link the program.

Using assembler: arg1

An informational message saying which assembler is being used.

Error: Error while assembling exitcode arg1

There was an error while assembling the file using an external assembler. Consult the documentation of the assembler tool to find out more information on this error.

Error: Can't call the assembler, error arg1 switching to external assembling

An error occurred when calling an external assembler. The compiler will produce a script that can be used to assemble and link the program.

Info: Assembling arg1

An informational message stating which file is being assembled.

Info: Assembling with smartlinking arg1

An informational message stating which file is being assembled using smartlinking.

Warning: Object arg1 not found, Linking may fail !

One of the object files is missing, and linking will probably fail. Check your paths.

Warning: Library arg1 not found, Linking may fail !

One of the library files is missing, and linking will probably fail. Check your paths.

Error: Error while linking

Generic error while linking.

Error: Can't call the linker, switching to external linking

An error occurred when calling an external linker. The compiler will produce a script that can be used to assemble and link the program.

Info: Linking arg1

An informational message, showing which program or library is being linked.

Error: Util arg1 not found, switching to external linking

An external tool was not found. The compiler will produce a script that can be used to assemble and link or postprocess the program.

Using util arg1

An informational message, showing which external program (usually a postprocessor) is being used.

Error: Creation of Executables not supported

Creating executable programs is not supported for this platform, because it was not yet implemented in the compiler.

Error: Creation of Dynamic/Shared Libraries not supported

Creating dynamically loadable libraries is not supported for this platform, because it was not yet implemented in the compiler.

Error: Creation of Static Libraries not supported

Creating static libraries is not supported for this platform, because it was not yet implemented in the compiler.

Info: Closing script arg1

Informational message showing when writing of the external assembling and linking script is finished.

Error: resource compiler "arg1" not found, switching to external mode

An external resource compiler was not found. The compiler will produce a script that can be used to assemble, compile resources and link or postprocess the program.

Info: Compiling resource arg1

An informational message, showing which resource is being compiled.

unit arg1 cannot be statically linked, switching to smart linking

Static linking was requested, but a unit which is not statically linkable was used.

unit arg1 cannot be smart linked, switching to static linking

Smart linking was requested, but a unit which is not smart-linkable was used.

unit arg1 cannot be shared linked, switching to static linking

Shared linking was requested, but a unit which is not shared-linkable was used.

Error: unit arg1 cannot be smart or static linked

Smart or static linking was requested, but a unit which cannot be used for either was used.

Error: unit arg1 cannot be shared or static linked

Shared or static linking was requested, but a unit which cannot be used for either was used.

Calling resource compiler "arg1" with "arg2" as command line

An informational message showing which command line is used for the resource compiler.

Error: Error while compiling resources

The resource compiler or converter returned an error.

Error: Can't call the resource compiler "arg1", switching to external mode

An error occurred when calling a resource compiler. The compiler will produce a script that can be used to assemble, compile resources and link or postprocess the program.

Error: Can't open resource file "arg1"

An error occurred resource file cannot be opened.

Error: Can't write resource file "arg1"

An error occurred resource file cannot be written.

Note: File "arg1" not found for backquoted cat command

The compiler did not find the file that should be expanded into linker parameters

Warning: "arg1" not found, this will probably cause a linking failure

The compiler adds certain startup code files to the linker only when they are found. If they are not found, they are not added and this might cause a linking failure.

C.8 Executable information messages.

This section lists all messages that the compiler emits when an executable program is produced, and only when the internal linker is used.

Fatal: Can't post process executable arg1

Fatal error when the compiler is unable to post-process an executable.

Fatal: Can't open executable arg1

Fatal error when the compiler cannot open the file for the executable.

Size of Code: arg1 bytes

Informational message showing the size of the produced code section.

Size of initialized data: arg1 bytes

Informational message showing the size of the initialized data section.

Size of uninitialized data: arg1 bytes

Informational message showing the size of the uninitialized data section.

Stack space reserved: arg1 bytes

Informational message showing the stack size that the compiler reserved for the executable.

Stack space committed: arg1 bytes

Informational message showing the stack size that the compiler committed for the executable.

C.9 Linker messages

This section lists messages produced by internal linker.

Fatal: Executable image size is too big for arg1 target.

Fatal error when resulting executable is too big.

Warning: Object file "arg1" contains 32-bit absolute relocation to symbol "arg2".

Warning when 64-bit object file contains 32-bit absolute relocations. In such case an executable image can be loaded into lower 4Gb of address space only.

Error: Program segment too large (exceeds 64k by arg1 bytes)

Error when a 16-bit program is compiled in the tiny memory model, but its size exceeds 64k

Error: Code segment "arg1" too large (exceeds 64k by arg2 bytes)

Error when a 16-bit program's code segment exceeds 64k bytes

Error: Data segment "arg1" too large (exceeds 64k by arg2 bytes)

Error when a 16-bit program's data segment exceeds 64k bytes

Error: Segment "arg1" too large (exceeds 64k by arg2 bytes)

Error when a 16-bit program contains a segment that exceeds 64k bytes

Error: Group "arg1" too large (exceeds 64k by arg2 bytes)

Error when a 16-bit program's object modules define a segment group that exceeds 64k bytes

Error: Cannot create a .COM file, because the program contains segment relocations

Error occurs, when creating a tiny model DOS .COM file, but at least one of the program's object modules contains segment relocations. Segment relocations might be caused by the use of the Seg() function or by the SEG assembler directive (either in pascal's built-in inline assembler, or in an externally linked assembly module).

Warning: Program "arg1" uses experimental CheckPointer option

Error: Multiple defined symbol "arg1"

The specified symbol is already defined inside the whole collection of object files.

Error: COMDAT selection mode arg1 not supported (section: "arg1")

The specified COMDAT selection mode is not supported.

Error: Associative section expected for COMDAT section "arg1"

The specified COMDAT section is specified as expecting an associative section, but none is specified.

Error: COMDAT section selection mode doesn't match for section "arg1" and symbol "arg2"

All COMDAT symbols/sections need to use the same selection mode.

Error: Associative COMDAT section for section "arg1" not found

The COMDAT section expects an associative section, but it was not found inside the object file.

Discarding duplicate symbol "arg1" due to COMDAT selection mode

The COMDAT section specifies that any section with the same name might be selected and this specific section was selected to be discarded.

Discarding duplicate symbol "arg1" with same size due to COMDAT selection mode

The COMDAT section specifies that any section with the same name and size might be selected and this specific section was selected to be discarded.

Discarding duplicate symbol "arg1" with same content due to COMDAT selection mode

The COMDAT section specifies that any section with the same name and content might be selected and this specific section was selected to be discarded.

Replacing duplicate symbol "arg1" with smaller size due to COMDAT selection mode

The COMDAT section specifies that the largest section with the same name should be selected this specific section was larger than the previous largest one.

Error: Size of duplicate COMDAT symbol "arg1" differs

The COMDAT section specifies that all sections with the same name need to have the same size, but this section had a different size.

Error: Content of duplicate COMDAT symbol "arg1" differs

The COMDAT section specifies that all sections with the same name need to have the same content, but this section had a different size.

Error: COMDAT selection mode for symbol "arg1" differs

C.10 Unit loading messages.

This section lists all messages that can occur when the compiler is loading a unit from disk into memory. Many of these messages are informational messages.

Unitsearch: arg1

When you use the `-vt` option, the compiler tells you where it tries to find unit files.

PPU Loading arg1

When the `-vt` switch is used, the compiler tells you what units it loads.

PPU Name: arg1

When you use the `-vu` flag, the unit name is shown.

PPU Flags: arg1

When you use the `-vu` flag, the unit flags are shown.

PPU Crc: arg1

When you use the `-vu` flag, the unit CRC check is shown.

PPU Time: arg1

When you use the `-vu` flag, the time the unit was compiled is shown.

PPU File too short

The ppufile is too short, not all declarations are present.

PPU Invalid Header (no PPU at the begin)

A unit file contains as the first three bytes the ASCII codes of the characters PPU.

PPU Invalid Version arg1

This unit file was compiled with a different version of the compiler, and cannot be read.

PPU is compiled for another processor

This unit file was compiled for a different processor type, and cannot be read.

PPU is compiled for another target

This unit file was compiled for a different target, and cannot be read.

PPU Source: arg1

When you use the `-vu` flag, the unit source file name is shown.

Writing arg1

When you specify the `-vu` switch, the compiler will tell you where it writes the unit file.

Fatal: Can't Write PPU-File

An error occurred when writing the unit file.

Fatal: Error reading PPU-File

This means that the unit file was corrupted, and contains invalid information. Recompilation will be necessary.

Fatal: unexpected end of PPU-File

Unexpected end of file. This may mean that the PPU file is corrupted.

Fatal: Invalid PPU-File entry: arg1

The unit the compiler is trying to read is corrupted, or generated with a newer version of the compiler.

Fatal: PPU Dbx count problem

There is an inconsistency in the debugging information of the unit.

Error: Illegal unit name: arg1 (expecting arg2)

The name of the unit does not match the file name. There might to two reasons: either there is a spelling mistake in the unit name or there is a unit with a 8.3 name where the 8 characters are equal to first 8 characters of the name of a unit with a longer name. However, this unit is not found. Example: Program contains `uses mytestunit;`, the unit file or source of `mytestunit` are not available but there is a source with the name `mytestun`. Then compiler tries to compile and use that one, however the expected unit name does not match.

Fatal: Too much units

Free Pascal has a limit of 1024 units in a program. You can change this behavior by changing the `maxunits` constant in the `fmodule.pas` file of the compiler, and recompiling the compiler.

Fatal: Circular unit reference between arg1 and arg2

Two units are using each other in the interface part. This is only allowed in the `implementation` part. At least one unit must contain the other one in the `implementation` section.

Fatal: Can't compile unit arg1, no sources available

A unit was found that needs to be recompiled, but no sources are available.

Fatal: Can't find unit arg1 used by arg2

You tried to use a unit of which the PPU file isn't found by the compiler. Check your configuration file for the unit paths.

Warning: Unit arg1 was not found but arg2 exists

This error message is no longer used.

Fatal: Unit arg1 searched but arg2 found

DOS truncation of 8 letters for unit PPU files may lead to problems when unit name is longer than 8 letters.

Warning: Compiling the system unit requires the -Us switch

When recompiling the system unit (it needs special treatment), the `-Us` switch must be specified.

Fatal: There were arg1 errors compiling module, stopping

When the compiler encounters a fatal error or too many errors in a module then it stops with this message.

Load from arg1 (arg2) unit arg3

When you use the `-vu` flag, which unit is loaded from which unit is shown.

Recompiling arg1, checksum changed for arg2

The unit is recompiled because the checksum of a unit it depends on has changed.

Recompiling arg1, source found only

When you use the `-vu` flag, these messages tell you why the current unit is recompiled.

Recompiling unit, static lib is older than ppufile

When you use the `-vu` flag, the compiler warns if the static library of the unit is older than the unit file itself.

Recompiling unit, shared lib is older than ppufile

When you use the `-vu` flag, the compiler warns if the shared library of the unit is older than the unit file itself.

Recompiling unit, obj and asm are older than ppufile

When you use the `-vu` flag, the compiler warns if the assembler or object file of the unit is older than the unit file itself.

Recompiling unit, obj is older than asm

When you use the `-vu` flag, the compiler warns if the assembler file of the unit is older than the object file of the unit.

Parsing interface of arg1

When you use the `-vu` flag, the compiler warns that it starts parsing the interface part of the unit.

Parsing implementation of arg1

When you use the `-vu` flag, the compiler warns that it starts parsing the implementation part of the unit.

Second load for unit arg1

When you use the `-vu` flag, the compiler warns that it starts recompiling a unit for the second time. This can happen with interdependent units.

PPU Check file arg1 time arg2

When you use the `-vu` flag, the compiler shows the filename and date and time of the file on which a recompile depends.

Warning: Can't recompile unit arg1, but found modified include files

A unit was found to have modified include files, but some source files were not found, so recompilation is impossible.

File arg1 is newer than the one used for creating PPU file arg2

A modified source file for a compiler unit was found.

Trying to use a unit which was compiled with a different FPU mode

Trying to compile code while using units which were not compiled with the same floating point format mode. Either all code should be compiled with FPU emulation on, or with FPU emulation off.

Loading interface units from arg1

When you use the `-vu` flag, the compiler warns that it is starting to load the units defined in the interface part of the unit.

Loading implementation units from arg1

When you use the `-vu` flag, the compiler warns that it is starting to load the units defined in the implementation part of the unit.

Interface CRC changed for unit arg1

When you use the `-vu` flag, the compiler warns that the CRC calculated for the interface has been changed after the implementation has been parsed.

Implementation CRC changed for unit arg1

When you use the `-vu` flag, the compiler warns that the CRC calculated has been changed after the implementation has been parsed.

Finished compiling unit arg1

When you use the `-vu` flag, the compiler warns that it has finished compiling the unit.

Adding dependency: arg1 depends on arg2

When you use the `-vu` flag, the compiler warns that it has added a dependency between the two units.

No reload, is caller: arg1

When you use the `-vu` flag, the compiler warns that it will not reload the unit because it is the unit that wants to load this unit.

No reload, already in second compile: arg1

When you use the `-vu` flag, the compiler warns that it will not reload the unit because it is already in a second recompile.

Flag for reload: arg1

When you use the `-vu` flag, the compiler warns that it has to reload the unit.

Forced reloading

When you use the `-vu` flag, the compiler warns that it is reloading the unit because it was required.

Previous state of arg1: arg2

When you use the `-vu` flag, the compiler shows the previous state of the unit.

Already compiling arg1, setting second compile

When you use the `-vu` flag, the compiler warns that it is starting to recompile a unit for the second time. This can happen with interdependent units.

Loading unit arg1

When you use the `-vu` flag, the compiler warns that it starts loading the unit.

Finished loading unit arg1

When you use the `-vu` flag, the compiler warns that it finished loading the unit.

Registering new unit arg1

When you use the `-vu` flag, the compiler warns that it has found a new unit and is registering it in the internal lists.

Re-resolving unit arg1

When you use the `-vu` flag, the compiler warns that it has to recalculate the internal data of the unit.

Skipping re-resolving unit arg1, still loading used units

When you use the `-vu` flag, the compiler warns that it is skipping the recalculation of the internal data of the unit because there is no data to recalculate.

Unloading resource unit arg1 (not needed)

When you use the `-vu` flag, the compiler warns that it is unloading the resource handling unit, since no resources are used.

Error: Unit arg1 was compiled using a different whole program optimization feedback input (arg2, arg3); recompile it without wpo or use the same wpo feedback input file for this compilation invocation

When a unit has been compiled using a particular whole program optimization (wpo) feedback file (`-FW<x> -OW<x>`), this compiled version of the unit is specialised for that particular compilation scenario and cannot be used in any other context. It has to be recompiled before you can use it in another program or with another wpo feedback input file.

Indirect interface (objects/classes) CRC changed for unit arg1

When you use the `-vu` flag, the compiler warns that the indirect CRC calculated for the unit (this is the CRC of all classes/objects/interfaces/... in the interfaces of units directly or indirectly used by this unit in the interface) has been changed after the implementation has been parsed.

PPU is compiled for another i8086 memory model

This unit file was compiled for a different i8086 memory model and cannot be read.

Loading unit arg1 from package arg2

The unit is loaded from a package.

Fatal: Internal type "arg1" was not found. Check if you use the correct run time library.

The compiler expects that the runtime library contains certain types. If you see this error and you didn't change the runtime library code, it's very likely that the runtime library you're using doesn't match the compiler in use. If you changed the runtime library this error means that you removed a type which the compiler needs for internal use.

Fatal: Internal type "arg1" does not look as expected. Check if you use the correct run time library.

The compiler expects that the runtime library contains certain types. If you see this error and you didn't change the runtime library code, it's very likely that the runtime library you're using doesn't match the compiler in use. If you changed the runtime library this error means that you changed a type which the compiler needs for internal use and which needs to have a certain structure.

C.11 Command line handling errors

This section lists errors that occur when the compiler is processing the command line or handling the configuration files.

Warning: Only one source file supported, changing source file to compile from "arg1" into "arg2"

You can specify only one source file on the command line. The last one will be compiled, others will be ignored. This may indicate that you forgot a ' - ' sign.

Warning: DEF file can be created only for OS/2

This option can only be specified when you're compiling for OS/2.

Error: nested response files are not supported

You cannot nest response files with the @file command line option.

Fatal: No source file name in command line

The compiler expects a source file name on the command line.

Note: No option inside arg1 config file

The compiler didn't find any option in that config file.

Error: Illegal parameter: arg1

You specified an unknown option.

Hint: -? writes help pages

When an unknown option is given, this message is displayed.

Fatal: Too many config files nested

You can only nest up to 16 config files.

Fatal: Unable to open file arg1

The option file cannot be found.

Reading further options from arg1

Displayed when you have notes turned on, and the compiler switches to another options file.

Warning: Target is already set to: arg1

Displayed if more than one -T option is specified.

Warning: Shared libs not supported on DOS platform, reverting to static

If you specify -CD for the DOS platform, this message is displayed. The compiler supports only static libraries under DOS.

Fatal: In options file arg1 at line arg2 too many #IF(N)DEFs encountered

The #IF (N) DEF statements in the options file are not balanced with the #ENDIF statements.

Fatal: In options file arg1 at line arg2 unexpected #ENDIFs encountered

The #IF (N) DEF statements in the options file are not balanced with the #ENDIF statements.

Fatal: Open conditional at the end of the options file

The #IF (N) DEF statements in the options file are not balanced with the #ENDIF statements.

Warning: Debug information generation is not supported by this executable

It is possible to have a compiler executable that doesn't support the generation of debugging info. If you use such an executable with the -g switch, this warning will be displayed.

Hint: Try recompiling with -dGDB

It is possible to have a compiler executable that doesn't support the generation of debugging info. If you use such an executable with the -g switch, this warning will be displayed.

Warning: You are using the obsolete switch arg1

This warns you when you use a switch that is not needed/supported anymore. It is recommended that you remove the switch to overcome problems in the future, when the meaning of the switch may change.

Warning: You are using the obsolete switch arg1, please use arg2

This warns you when you use a switch that is not supported anymore. You must now use the second switch instead. It is recommended that you change the switch to overcome problems in the future, when the meaning of the switch may change.

Note: Switching assembler to default source writing assembler

This notifies you that the assembler has been changed because you used the -a switch, which cannot be used with a binary assembler writer.

Warning: Assembler output selected "arg1" is not compatible with "arg2"**Warning: "arg1" assembler use forced**

The assembler output selected cannot generate object files with the correct format. Therefore, the default assembler for this target is used instead.

Reading options from file arg1

Options are also read from this file.

Reading options from environment arg1

Options are also read from this environment string.

Handling option "arg1"

Debug info that an option is found and will be handled.

***** press enter *****

Message shown when help is shown page per page. When pressing the ENTER Key, the next page of help is shown. If you press q and then ENTER, the compiler exits.

Hint: Start of reading config file arg1

Start of configuration file parsing.

Hint: End of reading config file arg1

End of configuration file parsing.

interpreting option "arg1"

The compiler is interpreting an option

interpreting firstpass option "arg1"

The compiler is interpreting an option for the first time.

interpreting file option "arg1"

The compiler is interpreting an option which it read from the configuration file.

Reading config file "arg1"

The compiler is starting to read the configuration file.

found source file name "arg1"

Additional information about options. Displayed when you have the debug option turned on.

Error: Unknown codepage "arg1"

An unknown codepage for the source files was requested. The compiler is compiled with support for several codepages built-in. The requested codepage is not in that list. You will need to recompile the compiler with support for the codepage you need.

Fatal: Config file arg1 is a directory

Directories cannot be used as configuration files.

Warning: Assembler output selected "arg1" cannot generate debug info, debugging disabled

The selected assembler output cannot generate debugging information, debugging option is therefore disabled.

Warning: Use of ppc386.cfg is deprecated, please use fpc.cfg instead

Using ppc386.cfg is still supported for historical reasons, however, for a multiplatform system the naming makes no sense anymore. Please continue to use fpc.cfg instead.

Fatal: In options file arg1 at line arg2 #ELSE directive without #IF(N)DEF found

An #ELSE statement was found in the options file without a matching #IF (N) DEF statement.

Fatal: Option "arg1" is not, or not yet, supported on the current target platform

Not all options are supported or implemented for all target platforms. This message informs you that a chosen option is incompatible with the currently selected target platform.

Fatal: The feature "arg1" is not, or not yet, supported on the selected target platform

Not all features are supported or implemented for all target platforms. This message informs you that a chosen feature is incompatible with the currently selected target platform.

Note: DWARF debug information cannot be used with smart linking on this target, switching to static linking

Smart linking is currently incompatible with DWARF debug information on most platforms, so smart linking is disabled in such cases.

Warning: Option "arg1" is ignored for the current target platform.

Not all options are supported or implemented for all target platforms. This message informs you that a chosen option is ignored for the currently selected target platform.

Warning: Disabling external debug information because it is unsupported for the selected target/debug format combination.

Not all debug formats can be stored in an external file on all platforms. In particular, on Mac OS X only DWARF debug information can be stored externally.

Note: DWARF debug information cannot be used with smart linking with external assembler, disabling static library creation.

Smart linking is currently incompatible with DWARF debug information on most platforms, so smart linking is disabled in such cases.

Error: Invalid value for MACOSX_DEPLOYMENT_TARGET environment variable: arg1

Error: Invalid value for IPHONEOS_DEPLOYMENT_TARGET environment variable: arg1

On Mac OS X, the MACOSX_DEPLOYMENT_TARGET/IPHONEOS_DEPLOYMENT_TARGET environment variable can be used to set the default target OS version. In case of Mac OS X, it has to be of the format XY.Z or XY.Z.AB with X, Y,Z , A and B all digits from 0-9. In case of iOS, it has to be X.Z.A, where X, Z and A can all be either 1 or 2 digits from 0-9.

Error: You must use a FPU type of VFPV2, VFPV3 or VFPV3_D16 when using the EABIHF ABI target

The EABIHF (VFP hardfloat) ABI target can only be used with VFP FPUs.

Warning: The selected debug format is not supported on the current target, not changing the current setting

Not all targets support all debug formats (in particular, Stabs is not supported on 64 bit targets).

Error: argument to "arg1" is missing

Displayed when parameter must be followed by an argument.

Error: malformed parameter: arg1

Given argument is not valid for parameter.

Warning: Smart linking requires external linker

Error: Creating .COM files is not supported in the current memory model. Only the tiny memory model supports making .COM files.

Do not enable experimental -gc option if -Ur option is given.

Warning: Experimental CheckPointer option not enabled because it is incompatible with -Ur option.

The compiler binary only supports a single target architecture. Invoke the fpc binary if you wish to select a compiler binary for a different target architecture.

Error: Unsupported target architecture -Parg1, invoke the "fpc" compiler driver instead.

The ppc<target> executables support only a single target architecture. They do not support the -P switch. The compiler driver "fpc" handles this switch, so you have to invoke compiling by using "fpc" instead

Error: Feature switches are only supported while compiling the system unit.

To selected a certain feature, the system unit must be compiled with this feature enabled. All other units inherited the features set by the system unit through the ppu of the system unit.

Note: The selected debug format is not supported by the internal linker, switching to external linking

Error: You can't use both options (arg1) (arg2) at same time.

C.12 Whole program optimization messages

This section lists errors that occur when the compiler is performing whole program optimization.

Fatal: Cannot open whole program optimization feedback file "arg1"

The compiler cannot open the specified feedback file with whole program optimization information.

Processing whole program optimization information in wpo feedback file "arg1"

The compiler starts processing whole program optimization information found in the named file.

Finished processing the whole program optimization information in wpo feedback file "arg1"

The compiler has finished processing the whole program optimization information found in the named file.

Error: Expected section header, but got "arg2" at line arg1 of wpo feedback file

The compiler expected a section header in the whole program optimization file (starting with %), but did not find it.

Warning: No handler registered for whole program optimization section "arg2" at line arg1 of wpo feedback file, ignoring

The compiler has no handler to deal with the mentioned whole program optimization information section, and will therefore ignore it and skip to the next section.

Found whole program optimization section "arg1" with information about "arg2"

The compiler encountered a section with whole program optimization information, and according to its handler this section contains information usable for the mentioned purpose.

Fatal: The selected whole program optimizations require a previously generated feedback file (use -Fw to specify)

The compiler needs information gathered during a previous compilation run to perform the selected whole program optimizations. You can specify the location of the feedback file containing this information using the -Fw switch.

Error: No collected information necessary to perform "arg1" whole program optimization found

While you pointed the compiler to a file containing whole program optimization feedback, it did not contain the information necessary to perform the selected optimizations. You most likely have to recompile the program using the appropriate -OWxxx switch.

Fatal: Specify a whole program optimization feedback file to store the generated info in (using -FW)

You have to specify the feedback file in which the compiler has to store the whole program optimization feedback that is generated during the compilation run. This can be done using the -FW switch.

Error: Not generating any whole program optimization information, yet a feedback file was specified (using -FW)

The compiler was instructed to store whole program optimization feedback into a file specified using -FW, but not to actually generate any whole program optimization feedback. The classes of to be generated information can be specified using -OWxxx.

Error: Not performing any whole program optimizations, yet an input feedback file was specified (using -Fw)

The compiler was not instructed to perform any whole program optimizations (no -Owxxx parameters), but nevertheless an input file with such feedback was specified (using -Fwyyy). Since this can indicate that you forgot to specify an -Owxxx parameter, the compiler generates an error in this case.

Skipping whole program optimization section "arg1", because not needed by the requested optimizations

The whole program optimization feedback file contains a section with information that is not required by the selected whole program optimizations.

Warning: Overriding previously read information for "arg1" from feedback input file using information in section "arg2"

The feedback file contains multiple sections that provide the same class of information (e.g., information about which virtual methods can be devirtualized). In this case, the information in last encountered section is used. Turn on debugging output (-vd) to see which class of information is provided by each section.

Error: Cannot extract symbol liveness information from program when stripping symbols, use -Xs-

Certain symbol liveness collectors extract the symbol information from the linked program. If the symbol information is stripped (option -Xs), this is not possible.

Error: Cannot extract symbol liveness information from program when not linking

Certain symbol liveness collectors extract the symbol information from the linked program. If the program is not linked by the compiler, this is not possible.

Fatal: Cannot find "arg1" or "arg2" to extract symbol liveness information from linked program

Certain symbol liveness collectors need a helper program to extract the symbol information from the linked program. This helper program is normally 'nm', which is part of the GNU binutils.

Error: Error during reading symbol liveness information produced by "arg1"

An error occurred during the reading of the symbol liveness file that was generated using the 'nm' or 'objdump' program. The reason can be that it was shorter than expected, or that its format was not understood.

Fatal: Error executing "arg1" (exitcode: arg2) to extract symbol information from linked program

Certain symbol liveness collectors need a helper program to extract the symbol information from the linked program. The helper program produced the reported error code when it was run on the linked program.

Error: Collection of symbol liveness information can only help when using smart linking, use -CX -XX

Whether or not a symbol is live is determined by looking whether it exists in the final linked program. Without smart linking/dead code stripping, all symbols are always included, regardless of whether they are actually used or not. So in that case all symbols will be seen as live, which makes this optimization ineffective.

Error: Cannot create specified whole program optimisation feedback file "arg1"

The compiler is unable to create the file specified using the -FW parameter to store the whole program optimisation information.

C.13 Package loading messages.

This section lists all messages that can occur when the compiler is loading a package from disk into memory, saving a package from memory to disk or when parsing packages in general. Many of these messages are informational messages.

Fatal: Can't find package arg1

You tried to use a package of which the PCP file isn't found by the compiler. Check your configuration file for the package paths.

PCP file for package arg1 found

The PCP file for the specified package was found.

Error: Duplicate package arg1

The package was already specified as required package and may not be specified a second time.

Error: Unit arg1 can not be part of a package

The unit can not be part of a package because the `DenyPackageUnit` directive is enabled for the unit.

Note: Unit arg1 is implicitly imported into package arg2

The unit was not specified as part of the `contains` section and is also not included in one of the required packages. Add the unit to the `contains` section to increase compatibility with other packages.

Fatal: Failed to create PCP file arg2 for package arg1

The PCP file for the package could not be created.

Fatal: Failed to read PCP file for package arg1

The PCP file for the package could not be read.

PCP loading arg1

When the `-vt` switch is used, the compiler tells you what packages it loads.

PCP Name: arg1

When you use the `-vu` flag, the package name is shown.

PCP Flags: arg1

When you use the `-vu` flag, the package flags are shown.

PCP Crc: arg1

When you use the `-vu` flag, the package CRC check is shown.

PCP Time: arg1

When you use the `-vu` flag, the time the package was compiled is shown.

PCP File too short

The PCP file is too short, not all declarations are present.

PCP Invalid Header (no PCP at the begin)

A package file contains as the first three bytes the ASCII codes of the characters `PCP`.

PCP Invalid Version arg1

This package file was compiled with a different version of the compiler, and cannot be read.

PCP is compiled for another processor

This package file was compiled for a different processor type, and cannot be read.

PCP is compiled for another target

This package file was compiled for a different target, and cannot be read.

Writing arg1

When you specify the `-vu` switch, the compiler will tell you where it writes the package file.

Fatal: Can't Write PCP-File

An error occurred when writing the package file.

Fatal: Error reading PCP-File

This means that the package file was corrupted, and contains invalid information. Recompile will be necessary.

Fatal: unexpected end of PCP-File

Unexpected end of file. This may mean that the PCP file is corrupted.

Fatal: Invalid PCP-File entry: arg1

The unit the compiler is trying to read is corrupted, or generated with a newer version of the compiler.

Trying to use a unit which was compiled with a different FPU mode

Trying to compile code while using units which were not compiled with the same floating point format mode. Either all code should be compiled with FPU emulation on, or with FPU emulation off.

Package search: arg1

When you use the `-vt` option, the compiler tells you where it tries to find package files.

Required package arg1

When you specify the `-vu` switch, the compiler will tell you which packages a package requires.

Contained unit arg1

When you specify the `-vu` switch, the compiler will tell you which units a package contains.

Error: Unit arg1 is already contained in package arg2

A unit specified in a contains sections must not be part of a required package. Note that a unit might have become part of another package by indirectly including it.

Warning: Unit arg1 is imported from indirectly required package arg2

If a unit from a package that is not part of the `requires` section is used then the package should require this unit directly to avoid confusion.

PPL filename arg1

The name of the binary package library that is stored in the PCP.

C.14 Assembler reader errors.

This section lists the errors that are generated by the inline assembler reader. They are *not* the messages of the assembler itself.

C.14.1 General assembler errors

Divide by zero in asm evaluator This fatal error is reported when a constant assembler expression performs a division by zero.

Evaluator stack overflow, Evaluator stack underflow These fatal error is reported when a constant assembler expression is too big to be evaluated by the constant parser. Try reducing the number of terms.

Invalid numeric format in asm evaluator This fatal error is reported when a non-numeric value is detected by the constant parser. Normally this error should never occur.

Invalid Operator in asm evaluator This fatal error is reported when a mathematical operator is detected by the constant parser. Normally this error should never occur.

Unknown error in asm evaluator This fatal error is reported when an internal error is detected by the constant parser. Normally this error should never occur.

Invalid numeric value This warning is emitted when a conversion from octal, binary or hexadecimal to decimal is outside of the supported range.

Escape sequence ignored This error is emitted when a non ANSI C escape sequence is detected in a C string.

Asm syntax error - Prefix not found This occurs when trying to use a non-valid prefix instruction.

Asm syntax error - Trying to add more than one prefix This occurs when you try to add more than one prefix instruction.

Asm syntax error - Opcode not found You have tried to use an unsupported or unknown opcode.

Constant value out of bounds This error is reported when the constant parser determines that the value you are using is out of bounds, either with the opcode or with the constant declaration used.

Non-label pattern contains @ This only applied to the m68k and Intel styled assembler. This is reported when you try to use a non-label identifier with an '@' prefix.

Internal error in Findtype()

Internal Error in ConcatOpcode()

Internal Error converting binary

Internal Error converting hexadecimal

Internal Error converting octal

Internal Error in BuildScaling()

Internal Error in BuildConstant()

internal error in BuildReference()

internal error in HandleExtend()

Internal error in ConcatLabeledInstr() These errors should never occur. If they do then you have found a new bug in the assembler parsers. Please contact one of the developers.

Opcode not in table, operands not checked This warning only occurs when compiling the system unit, or related files. No checking is performed on the operands of the opcodes.

@CODE and @DATA not supported This Turbo Pascal construct is not supported.

SEG and OFFSET not supported This Turbo Pascal construct is not supported.

Modulo not supported Modulo constant operation is not supported.

Floating point binary representation ignored

Floating point hexadecimal representation ignored

Floating point octal representation ignored These warnings occur when a floating point constant is declared in a base other than decimal. No conversion can be done on these formats. You should use a decimal representation instead.

Identifier supposed external This warning occurs when a symbol is not found in the symbol table. It is therefore considered external.

Functions with void return value can't return any value in asm code Only routines with a return value can have a return value set.

Error in binary constant

Error in octal constant

Error in hexadecimal constant

Error in integer constant These errors are reported when you tried using a constant expression that is invalid or whose value is out of range.

Invalid labeled opcode

Asm syntax error - error in reference

Invalid Opcode

Invalid combination of opcode and operands

Invalid size in reference

Invalid middle sized operand

Invalid three operand opcode

Assembler syntax error

Invalid operand type You tried using an invalid combination of opcode and operands. Check the syntax and if you are sure it is correct, please contact one of the developers.

Unknown identifier The identifier you are trying to access does not exist, or is not within the current scope.

Trying to define an index register more than once

Trying to define a segment register twice

Trying to define a base register twice You are trying to define an index/segment register more than once.

Invalid field specifier The record or object field you are trying to access does not exist, or is incorrect.

Invalid scaling factor

Invalid scaling value

Scaling value only allowed with index Allowed scaling values are 1,2,4 or 8.

Cannot use SELF outside a method You are trying to access the SELF identifier for objects outside a method.

Invalid combination of prefix and opcode This opcode cannot be prefixed by this instruction.

Invalid combination of override and opcode This opcode cannot be overridden by this combination.

Too many operands on line At most three operand instructions exist on the m68k, and i386, you are probably trying to use an invalid syntax for this opcode.

Duplicate local symbol You are trying to redefine a local symbol, such as a local label.

Unknown label identifier

Undefined local symbol

local symbol not found inside asm statement This label does not seem to have been defined in the current scope.

Assemble node syntax error

Not a directive or local symbol The assembler statement is invalid, or you are not using a recognized directive.

C.14.2 I386 specific errors

repeat prefix and a segment override on <= i386 ... A problem with interrupts and a prefix instruction may occur and may cause false results on 386 and earlier computers.

Fwait can cause emulation problems with emu387 This warning is reported when using the FWAIT instruction. It can cause emulation problems on systems which use the em387.dxe emulator.

You need GNU as version >= 2.81 to compile this MMX code MMX assembler code can only be compiled using GAS v2.8.1 or later.

NEAR ignored

FAR ignored NEAR and FAR are ignored in the Intel assemblers, but are still accepted for compatibility with the 16-bit code model.

Invalid size for MOVSX/MOVZX

16-bit base in 32-bit segment

16-bit index in 32-bit segment 16-bit addressing is not supported. You must use 32-bit addressing.

Constant reference not allowed It is not allowed to try to address a constant memory address in protected mode.

Segment overrides not supported Intel style (eg: rep ds stosb) segment overrides are not supported by the assembler parser.

Expressions of the form [sreg:reg...] are currently not supported To access a memory operand in a different segment, you should use the sreg:[reg...] syntax instead of [sreg:reg...]

Size suffix and destination register do not match In intel AT&T syntax, you are using a register size which does not concord with the operand size specified.

Invalid assembler syntax. No ref with brackets

Trying to use a negative index register

Local symbols not allowed as references

Invalid operand in bracket expression

Invalid symbol name:

Invalid Reference syntax

Invalid string as opcode operand:

Null label references are not allowed

Using a defined name as a local label

Invalid constant symbol
Invalid constant expression
/ at beginning of line not allowed
NOR not supported
Invalid floating point register name
Invalid floating point constant:
Asm syntax error - Should start with bracket
Asm syntax error - register:
Asm syntax error - in opcode operand
Invalid String expression
Constant expression out of bounds
Invalid or missing opcode
Invalid real constant expression
Parenthesis are not allowed
Invalid Reference
Cannot use __SELF outside a method
Cannot use __OLDEBP outside a nested procedure
Invalid segment override expression
Strings not allowed as constants
Switching sections is not allowed in an assembler block
Invalid global definition
Line separator expected
Invalid local common definition
Invalid global common definition
assembler code not returned to text
invalid opcode size
Invalid character: <
Invalid character: >
Unsupported opcode
Invalid suffix for intel assembler
Extended not supported in this mode
Comp not supported in this mode
Invalid Operand:
Override operator not supported

C.14.3 m68k specific errors.

Increment and Decrement mode not allowed together You are trying to use dec/inc mode together.

Invalid Register list in movem/fmovem The register list is invalid. Normally a range of registers should be separated by - and individual registers should be separated by a slash.

Invalid Register list for opcode

68020+ mode required to assemble

Appendix D

Run-time errors

Applications generated by Free Pascal might generate run-time errors when certain abnormal conditions are detected in the application. This appendix lists the possible run-time errors and gives information on why they might be produced.

- 1 Invalid function number** An invalid operating system call was attempted.
- 2 File not found** Reported when trying to erase, rename or open a non-existent file.
- 3 Path not found** Reported by the directory handling routines when a path does not exist or is invalid. Also reported when trying to access a non-existent file.
- 4 Too many open files** The maximum number of files currently opened by your process has been reached. Certain operating systems limit the number of files which can be opened concurrently, and this error can occur when this limit has been reached.
- 5 File access denied** Permission to access the file is denied. This error might be caused by one of several reasons:
- Trying to open for writing a file which is read-only, or which is actually a directory.
 - File is currently locked or used by another process.
 - Trying to create a new file, or directory while a file or directory of the same name already exists.
 - Trying to read from a file which was opened in write-only mode.
 - Trying to write from a file which was opened in read-only mode.
 - Trying to remove a directory or file while it is not possible.
 - No permission to access the file or directory.
- 6 Invalid file handle** If this happens, the file variable you are using is trashed; it indicates that your memory is corrupted.
- 12 Invalid file access code** Reported when a reset or rewrite is called with an invalid `FileMode` value.
- 15 Invalid drive number** The number given to the `GetDir` or `ChDir` function specifies a non-existent disk.
- 16 Cannot remove current directory** Reported when trying to remove the currently active directory.

- 17 Cannot rename across drives** You cannot rename a file such that it would end up on another disk or partition.
- 100 Disk read error** An error occurred when reading from disk. Typically happens when you try to read past the end of a file.
- 101 Disk write error** Reported when the disk is full, and you're trying to write to it.
- 102 File not assigned** This is reported by `Reset`, `Rewrite`, `Append`, `Rename` and `Erase`, if you call them with an unassigned file as a parameter.
- 103 File not open** Reported by the following functions: `Close`, `Read`, `Write`, `Seek`, `EOF`, `FilePos`, `FileSize`, `Flush`, `BlockRead`, and `BlockWrite` if the file is not open.
- 104 File not open for input** Reported by `Read`, `BlockRead`, `EOF`, `Eoln`, `SeekEOF` or `SeekEoln` if the file is not opened with `Reset`.
- 105 File not open for output** Reported by `write` if a text file isn't opened with `Rewrite`.
- 106 Invalid numeric format** Reported when a non-numeric value is read from a text file, and a numeric value was expected.
- 107 Invalid enumeration** Reported when a text representation of an enumerated constant cannot be created in a call to `str` or `write(ln)`.
- 150 Disk is write-protected** (Critical error)
- 151 Bad drive request struct length** (Critical error)
- 152 Drive not ready** (Critical error)
- 154 CRC error in data** (Critical error)
- 156 Disk seek error** (Critical error)
- 157 Unknown media type** (Critical error)
- 158 Sector Not Found** (Critical error)
- 159 Printer out of paper** (Critical error)
- 160 Device write fault** (Critical error)
- 161 Device read fault** (Critical error)
- 162 Hardware failure** (Critical error)
- 200 Division by zero** The application attempted to divide a number by zero.
- 201 Range check error** If you compiled your program with range checking on, then you can get this error in the following cases:
1. An array was accessed with an index outside its declared range.
 2. Trying to assign a value to a variable outside its range (for instance an enumerated type).
- 202 Stack overflow error** The stack has grown beyond its maximum size (in which case the size of local variables should be reduced to avoid this error), or the stack has become corrupt. This error is only reported when stack checking is enabled.

203 Heap overflow error The heap has grown beyond its boundaries. This is caused when trying to allocate memory explicitly with `New`, `GetMem` or `ReallocMem`, or when a class or object instance is created and no memory is left. Please note that, by default, Free Pascal provides a growing heap, i.e. the heap will try to allocate more memory if needed. However, if the heap has reached the maximum size allowed by the operating system or hardware, then you will get this error.

204 Invalid pointer operation You will get this in several cases:

- if you call `Dispose` or `FreeMem` with an invalid pointer
- in case `New` or `GetMem` is called, and there is no more memory available. The behavior in this case depends on the setting of `ReturnNilIfGrowHeapFails`. If it is `True`, then `Nil` is returned. if `False`, then `runerror 204` is raised.

205 Floating point overflow You are trying to use or produce real numbers that are too large.

206 Floating point underflow You are trying to use or produce real numbers that are too small.

207 Invalid floating point operation Can occur if you try to calculate the square root or logarithm of a negative number.

210 Object not initialized When compiled with range checking on, a program will report this error if you call a virtual method without having called its object's constructor.

211 Call to abstract method Your program tried to execute an abstract virtual method. Abstract methods should be overridden, and the overriding method should be called.

212 Stream registration error This occurs when an invalid type is registered in the objects unit.

213 Collection index out of range You are trying to access a collection item with an invalid index (objects unit).

214 Collection overflow error The collection has reached its maximal size, and you are trying to add another element (objects unit).

215 Arithmetic overflow error This error is reported when the result of an arithmetic operation is outside of its supported range. Contrary to Turbo Pascal, this error is only reported for 32-bit or 64-bit arithmetic overflows. This is due to the fact that everything is converted to 32-bit or 64-bit before doing the actual arithmetic operation.

216 General Protection fault The application tried to access invalid memory space. This can be caused by several problems:

1. Dereferencing a `nil` pointer.
2. Trying to access memory which is out of bounds (for example, calling `move` with an invalid length).

217 Unhandled exception occurred An exception occurred, and there was no exception handler present. The `sysutils` unit installs a default exception handler which catches all exceptions and exits gracefully.

218 Invalid value specified Error 218 occurs when an invalid value was specified to a system call, for instance when specifying a negative value to a `seek()` call.

219 Invalid typecast Thrown when an invalid typecast is attempted on a class using the `as` operator. This error is also thrown when an object or class is typecast to an invalid class or object and a virtual method of that class or object is called. This last error is only detected if the `-CR` compiler option is used.

- 222 Variant dispatch error** No dispatch method to call from variant.
- 223 Variant array create** The variant array creation failed. Usually when there is not enough memory.
- 224 Variant is not an array** This error occurs when a variant array operation is attempted on a variant which is not an array.
- 225 Var Array Bounds check error** This error occurs when a variant array index is out of bounds.
- 227 Assertion failed error** An assertion failed, and no `AssertErrorProc` procedural variable was installed.
- 229 Safecall error check** This error occurs if a safecall check fails, and no handler routine is available.
- 231 Exception stack corrupted** This error occurs when the exception object is retrieved and none is available.
- 232 Threads not supported** Thread management relies on a separate driver on some operating systems (notably, Unixes). The unit with this driver needs to be specified on the `uses` clause of the program, preferably as the first unit (`cthreads` on unix).

Appendix E

A sample gdb.ini file

Here you have a sample `gdb.ini` file listing, which gives better results when using `gdb`. Under LINUX you should put this in a `.gdbinit` file in your home directory or the current directory.

```
set print demangle off
set gnutarget auto
set verbose on
set complaints 1000
set language c++
set print vtbl on
set print object on
set print symbol-filename on
set print pretty on
disp /i $eip
```

The current versions of GDB understand sets and strings, in older versions it was not possible to print them.

Appendix F

Options and settings

In table (F.1) a summary of available boolean compiler directives and the corresponding command line options are listed. Other directives and the corresponding options are shown in table (F.2). For more information about the command-line options, see chapter 5, page 24. For more information about the directives, see the [Programmer's Guide](#).

Table F.1: Boolean Options and directives

Short	long	Opt	Explanation
\$A [+/-]	\$ALIGN [ON/OFF]		Data alignment
\$B [+/-]	\$BOOLEVAL [ON/OFF]		Boolean evaluation mode
\$C [+/-]	\$ASSERTIONS [ON/OFF]	-Sa	Include assertions
\$D [+/-]	\$DEBUGINFO [ON/OFF]	-g	Include debug info
\$E [+/-]			Coprocessor emulation
\$F [+/-]			Far or near function (ignored)
\$G [+/-]			Generate 80286 code (ignored)
	\$GOTO [ON/OFF]	-Sg	Support GOTO and Label
	\$HINTS [ON/OFF]	-vh	Show hints
\$H [+/-]	\$LONGSTRINGS [ON/OFF]	-Sh	Use ansistrings
\$I [+/-]	\$IOCHECKS [ON/OFF]	-Ci	Check I/O operation result
	\$INLINE [ON/OFF]	-Si	Allow inline code
\$L [+/-]	\$LOCALSYMBOLS [ON/OFF]		Local symbol information
\$M [+/-]	\$TYPEINFO [ON/OFF]		Generate RTTI for classes
	\$MMX [ON/OFF]		Intel MMX support
\$N [+/-]			Floating point support
	\$NOTES [ON/OFF]	-vn	Emit notes
\$O [+/-]			Support overlays (ignored)
\$P [+/-]	\$OPENSTRINGS [ON/OFF]		Support open strings
\$Q [+/-]	\$OVERFLOWCHECKS [ON/OFF]	-Co	Overflow checking
\$R [+/-]	\$RANGECHECKS [ON/OFF]	-Cr	Range checks
\$S [+/-]		-Ct	Stack checks
	\$SMARTLINK [ON/OFF]	-CX	Use smartlinking
	\$STATIC [ON/OFF]	-St	Allow use of static
\$T [+/-]	\$TYPEDADDRESS [ON/OFF]		Typed addresses

Table F.2: Options and directives

Short	long	Opt	Explanation
	\$APPTYPE	-W	Application type (Win32/OS2)
	\$ASMMODE	-R	Assembler reader mode
	\$DEFINE	-d	Define symbol
	\$DESCRIPTION		Set program description
	\$ELSE		Conditional compilation switch
	\$ENDIF		Conditional compilation end
	\$FATAL		Report fatal error
	\$HINT		Emit hint message
\$I file	\$INCLUDE		Include file or literal text
	\$IF		Conditional compilation start
	\$IFDEF NAME		Conditional compilation start
	\$IFNDEF		Conditional compilation start
	\$IFOPT		Conditional compilation start
	\$INCLUDEPATH	-Fi	Set include path
	\$INFO		Emit information message
\$L file	\$LINK		Link object file
	\$LIBRARYPATH	-Fl	Set library path
	\$LINKLIB name		Link library
\$M MIN,MAX	\$MEMORY		Set memory sizes
	\$MACRO	-Sm	Allow use of macros
	\$MESSAGE		Emit message
	\$MODE		Set compatibility mode
	\$NOTE		Emit note message
	\$OBJECTPATH	-Fo	Set object path
	\$OUTPUT	-A	Set output format
	\$PACKENUM		Enumeration type size
	\$PACKRECORDS		Record element alignment
	\$SATURATION		Saturation (ignored)
	\$STOP		Stop compilation
	\$UNDEF	-u	Undefine symbol

Appendix G

Getting the latest sources or installers

Free Pascal is under continuous development. From time to time, a new set of installers with what are considered stable sources are made: these are the releases. They can be downloaded from the Free Pascal website. The downloads usually contain the sources from which the release is made.

If for some reason, a newer set of files is needed - for instance, because certain bugs that prevent a program from functioning correctly have been fixed, it is possible to download the latest source files and recompile them.

Note that the latest sources may or may not compile: sometimes things get broken, and then the downloaded sources are useless. For the fixes branches (mentioned below) the sources should always compile, so it may be best to use these only.

There are 3 ways to get the latest version.

G.1 Download via Subversion

All Free Pascal sources are in subversion, and can be downloaded anonymously from the Subversion server. With a suitable Subversion client, the following locations can be used:

`http://svn.freepascal.org/svn/fpc/trunk/`

This repository contains the latest sources of the compiler, RTL and packages. This is the active development branch.

The documentation and all examples from the documentation are in the following repository

`http://svn.freepascal.org/svn/fpcdocs/trunk/`

All the files needed to make a release can be retrieved from

`http://svn.freepascal.org/svn/fpcbuild/trunk/`

This repository contains external links to the other 2 repositories, and contains all scripts, demos and other files needed to construct a new release of Free Pascal.

Free Pascal maintains a fixes branch, which is used to create new releases after a major version change. The branches are located in

`http://svn.freepascal.org/svn/fpc/branches/fpc\_X\_Y`

Where X and Y make up the major release number of Free Pascal. For instance, the fixes used to make the 2.6.x versions of Free Pascal are available in

`http://svn.freepascal.org/svn/fpc/branches/fixes\_2\_6`

The Subversion archive is mirrored on the server

`svn2.freepascal.org`

G.2 Downloading a source zip

Every day, a zip is made which contains the sources as they are on this day, they are available from the FTP site:

`http://ftp.freepascal.org/develop.var`

This will lead to a download of the sources of the development branch:

`ftp://ftp.freepascal.org/pub/fpc/snapshot/trunk/source/fpc.zip`

and also of the fixes branch:

`ftp://ftp.freepascal.org/pub/fpc/snapshot/fixes/source/fpc.zip`

The creation of the zip files is an automated process, and so these files are created every day.

G.3 Downloading a snapshot

Some members of the Free Pascal team also maintain installable snapshots. These are installers, made with the sources of that day. Since the sources are not guaranteed to work, a snapshot of a certain day may not be available, or the person responsible for it didn't have the opportunity to make one: these snapshots may or may not be available. They can be downloaded from the same page as the daily source zip:

`http://ftp.freepascal.org/develop.var`

The snapshots are made for the development branch as well as for the fixes branch. They are available from

`ftp://ftp.freepascal.org/pub/fpc/snapshot/trunk/`

and

`ftp://ftp.freepascal.org/pub/fpc/snapshot/fixes/`

respectively.

The FTP site is mirrored, it may be faster to use a mirror.