SCO®

SCO OpenServer™ Development System

SCO® Optimizing C Compiler Release Notes

Version 2.1.4

SCO OpenServer™

product release notes

SCO® Optimizing C Compiler Version 2.1.4

These release notes are divided into the following sections:

1.0 Introduction
2.0 Package Contents
3.0 Installation
4.0 Characteristics
  4.1 Driver and Compiler
  4.2 Library Files
  4.3 Byte Magazine Benchmark Anomaly
5.0 New Features
6.0 Known Limitations

1.0 Introduction

The SCO Optimizing C Compiler™ and optimized math library package generates highly optimized code for execution on Intel386™, Intel486™, Pentium® and Pentium Pro processors. These release notes identify product characteristics and limitations of the software.

This release of the product contains two software patches. The first patch prevents any possibility of a reduction in precision caused by the floating point divide flaw which exists in early steppings of the Pentium processor. This patch affects divide calculations done at compile time, run time, and in intrinsics used by the math library. For more information, please refer to the description of the -fdiv_check and -nofdiv_check options in Section 5.1.

The second patch implements a workaround for the erratum "Incorrect Decode of Certain 0F Instructions". For more information see the description of the -0f_check switch in Section 5.1. Refer to the Pentium Processor Specification Update for a detailed description of the erratum conditions.

2.0 Package Contents

The product package contains:

- 3 diskettes (Volumes 1-3) containing the C compilation system driver (`icc`), compiler front-end (`edgcpfe`), compiler back-end (`cpcom,`), math libraries (`libm.a, libm_chk.a`), profile guided optimization tools (`ictdb, idtall, imerge`), man pages, Release Notes text file (`icc_note.txt`), and a compressed version of the User Guide (`users_guide.z`).
- 1 set of printed release notes

The package requires about 5.5 Mbytes of disk space.

3.0 Installation

The SCO Optimizing C Compiler installs on SCO OpenServer™ Release 5 as part of the SCO OpenServer Development System™.

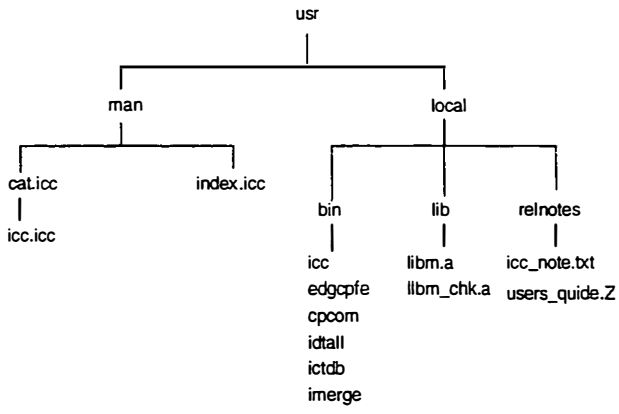
System requirements: 16 megabytes of RAM with 32 megabytes of swap space configured. More memory may be required for larger programs. Three factors affect the amount of virtual memory available to a program: the system swap space, the per-process virtual memory limit, and the per-process data space limit. The test platforms had 32-64 megabytes of virtual memory and 16-32 megabytes of physical memory.

To install the software, enter the `custom` command, then choose menu items in the following sequence: `Software, Install New`. Insert the first diskette, as requested, then follow the instructions on the screen. The installation program offers you choices, such as whether or not to overwrite a previous version of the software.

The optimized math libraries, `libm.a` and `libm_chk.a`, are installed in `/usr/local/lib`. To use the system version of this library, you must specify it using the `-L`*`path`* option and the `-nofdiv_check` option.

The `man` pages are installed automatically.

Once installed, the files are arranged in the directory structure, as follows:



If you are replacing your software with this new version, the installation backs up only the Optimizing C Compiler and library files.

To use the compiler, set the `$PATH` variable to point to `/usr/local/bin` in your `.login` or `.profile` startup file. For example, in the `.profile` file, change the `PATH` variable to the following line:

```
PATH=/usr/local/bin:$PATH
```

For the `.login` file, change the `PATH` as follows:

```
setenv PATH /usr/local/bin:$PATH
```

4.0 Characteristics

This section informs you of some unique characteristics of this compiler which might differ from other compilers you have used. For problems and anomalies that are likely to be corrected in later releases, see Section 6.0.

4.1 Driver and Compiler

The compiler generates inline code for many library function references, which usually results in faster computation. In addition, the `errno` variable is not set when the functions are expanded inline. Library routines are used, however, when you specify any of the following options at compilation: IEEE 754 conformance (`-mp`), ANSI standard conformance (`-Xc`), optimization level 0 (`-O0`), and debugging information (`-g`). So, your results can vary slightly under these circumstances.

The compiler cannot always determine the difference between a library routine and another routine with the same name. The compiler recognizes the non-library routine only if the number of arguments in the call differs from that of the library routine. To be sure the intended routine is used, use the `-nolib_inline` option whenever your program contains user supplied routines with the same names as the standard routines. The run-time library routine names are `memcmp`, `memcpy`, `memset`, `strcmp`, `strcpy`, `strlen`. The user's guide contains a list of the math library routine names.

The X Window System (X11R5) program modifies a `const struct` member with a pointer type. The SCO Optimizing C Compiler does not allow the program to modify such pointers.

For functions returning a `struct` or `union`, the SCO Optimizing C Compiler passes the `struct` or `union` using the same conventions as the current SCO cc compiler. This convention is not System V Application Binary Interface (iABI) compliant.

The current highest compilation level is `-O1` or `-O` which includes all scalar optimizations. To get further performance enhancements use the compiler options for specific optimizations. The `-mem` option invokes only memory optimization; the `-ip` option invokes only interprocedural options. If you want the compiler to perform both memory and interprocedural options, you must specify both options on the command line. In general, `-ip` can be very useful by itself but `-mem` usually needs `-ip` to produce performance improvements. For very large programs `-ip` should not be used as it will result in a very large increase in compile time with no improvement in execution time.

Under the `-ip` optimization switch, the compiler looks at the name of each routine as part of its criteria to determine when to expand a routine in line. Routines which contain the following substrings in their names are not expanded: abort, denied, err, exit, fail, fatal, fault, halt, init, interrupt, invalid, quit, rare, stop, timeout, trace, trap, and warn.

The `-tp` option can be used to generate optimal code for a particular processor. The code generated using this option could run slower on a different processor than the targeted processor. However, all code generated by this compiler runs on all 32-bit Intel Architecture processors. For this release the `-tp` switch can be set to `p6` for optimal code for the Pentium Pro microprocessor.

This compiler does not default to the same options as the cc compiler. In particular:

- icc's default optimization is `-O` or `-O1`,
- floating point exceptions are disabled,
- the default C dialect is `-Xt` for icc versus `-Xa` for cc,
- icc generates direct object modules by default,
- icc generates code to check for and correct inaccurate divides which may occur on some Pentium processors.

In general, objects compiled with icc and cc can be linked together without any problem. However, the following may cause problems in specific programs:

- Floats are passed as 4 bytes on the stack in the presence of a parameter with this compiler and as 8 bytes with cc.
- The exact layout of structures containing bit fields may be different.

The following are significant differences in behavior between icc and cc which may affect the porting of source code.

- Arguments to functions are evaluated from left to right by cc but they are evaluated from right to left by icc. This will not cause a link incompatibility but may cause illegal programs to behave differently.
- Floating point precision may vary between compilers based on how intermediate values are processed and how long values are kept in registers.

When CC is used to compile C++ programs with the `-icc` option specified, icc is used to compile the output from cfront. However, cc is used to link the object and since it does not know about the special math libraries used by the SCO Optimizing C Compiler, it does not link with the correct math library to check for the floating point divide flaw in some Pentium processors and link errors will occur. These errors can be eliminated by specifying `-nofdiv_check` on the call to CC or by using `-lm_chk -L/usr/local/lib` to link with the correct library. If the optimized math library without the divide check is needed, specify `-nofdiv_check -lm -L/usr/local/lib`.

4.2 Library Files

The SCO Optimizing C compiler allows you to use all the standard run-time libraries that are part of the standard development system for your SCO OpenServer Release 5 as part of the SCO OpenServer Development System.

In the package, you received two alternate versions of the math library, `libm.a` and `libm_chk.a` Both versions contain the same functions as the standard library. About half of the functions in the optimized library are written in assembly language and optimized for program execution speed on the Pentium processor. The optimized primitives are documented in the User's Guide.

The `libm_chk.a` library is automatically linked if needed when the `-fdiv_check` switch is active. To use the regular optimized math library with `-nofdiv_check`, you must specify the direct linker control option `-lm`. If the libraries are in a directory other than the default installation directory `/usr/local/lib`, use the `-L`*`pathname`* option to specify that directory. See the note above on compiling with CC -icc.

To link the standard library or your own version of `libm.a` without unresolved externals errors, you must specify `-nofdiv_check`. To prevent routines from being inlined, you must disable automatic inline expansion by compiling the program with the `-nolib_inline` option.

To enable exception handling as described in the Libraries chapter of the User's Guide, hardware exceptions must be explicitly enabled using the `-mP1OPT_fpsetmask_arg=` switch. By default, all exceptions are masked although the status word contains correct information on any exceptions that occurred.

4.3 Byte Magazine Benchmark Anomaly

The *Byte Magazine* arithmetic benchmark contains a variable declaration that distorts any results for the SCO Optimizing C compiler. The global variable `iter`, which is referenced outside the flow of the program by an asynchronous interrupt, in this case a signal, is not declared `volatile`, and the compiler optimizes away an assignment to this variable. Such optimization is allowed in Section 2.1.2.3 of the ANSI C standard. To obtain a usable benchmark result, declare the `iter` variable `volatile` before you compile the program.

5.0 New Features

The following features and improvements have been added since the icc release Version 2.0.

5.1 New or Modified Compiler Switches

Several switches have been added or modified for this release as follows:

`-fdiv_check`

The `-fdiv_check` option enables a software patch for the floating-point division flaw which exists in some steppings of the Pentium™ processor. This patch ensures floating-point result precision and is optimized for efficient performance on current and future Intel processors. This option is the default if `-tp px`, `-tp p5`, or no `-tp` option is specified.

The software patch affects the code the compiler generates for floating-point division and causes the compiler to generate calls for certain affected intrinsics. The compiler uses the following algorithm when it must perform floating-point division in the generated code:

1. Identify the executing processor and set a global variable. On Pentium™ processors exhibiting the floating-point division flaw, a process is followed to ensure that full precision is returned in the result of a division.

On other processors, only the original hardware division instruction is executed.

2. On a flawed processor, determine if the operands of an FDIV instruction may be susceptible to the floating-point division flaw. If the operands are found not to be susceptible to the flaw, a result is returned from a simple hardware division.

3. Once susceptible operands are identified, they are scaled to avoid problematic numeric ranges. The adjusted operands are used in a hardware division and the full precision result is then returned.

In this release of the compiler, the `-fdiv_check` option need not be specified as it is enabled by default. When this option is enabled, the compiler requires a special version of the math library, `libm_chk.a`, in order to link the user's programs. This library contains the support routines for the software patch, as well as checked versions of affected math library functions. See the User's Guide for more information on the math library.

`-g`

The use of `-g` with an explicit optimization option is allowed. When `-g` is specified with no optimization level, the optimization level is set to `-O0` and the frame pointer option, `-fp`, is activated.

When `-g` is specified with `-O` or `-mem`, debug information is generated but may not be totally accurate depending on the optimizations being done. The frame pointer, `-fp`, is not activated but may be activated by the user for more information.

The use of `-g` with `-ip` is not supported.

-M

Use the -M option to generate makefile dependency lines for each source file based on the #include lines found in the source file. The compiler displays the dependencies on standard output and prints the dependency for each included file on a separate line. The compilation process stops after preprocessing is completed.

For example, given the following contents of hello.c:

```
#include <stdio.h>
main(void)
{
      printf("Hello, World!\n");
}
```

the output generated by the command, icc -M hello.c, would be similar to the following:

```
hello.o: hello.c
hello.o: /usr/include/stdio.h
```

-nofdiv_check

The -nofdiv_check option disables the software patch for the floating-point division flaw which exists in some steppings of the Pentium™ processor. When this option is specified, the compiler uses simple hardware instructions for floating-point division and affected intrinsics. Also, the compiler does not require a special version of the math library in order to link the user's programs. This option is the default if the -tp p4 or -tp p6 options are specified.

-rcr

The -rcr option specifes that rounding control be set to round to nearest. This is the default.

-rct

The -rct options specifies that rounding control be set to truncate.

`-tp` <target>
The `-tp` option now supports the `p6` target for the Pentium Pro microprocessor.

`-0f_check`
This option forces final code selection to avoid the use of 0F instructions other than the 0F jumps. Also the fixed-register form of IMUL is used instead of the general register form. A slight code expansion and/or slow down may result.

For more information on compiler switches refer to the User's Guide or the man page.

5.2 Other New Features

The following miscellaneous new features have been added:

The compiler supports both ELF and COFF object modules. The ELF object modules are stamped with a version stamp. See the development system documentation for more information. In general, ELF binaries generated by this compiler are smaller and faster than COFF binaries generated by this compiler. The DWARF debug output generated for ELF binaries may be more complete.

The need to have two separate sets of compiler binaries to generate both COFF and ELF is removed for this release.

The compiler supports `#pragma comment` statements which allows the placement of comments in object or executable files.

5.3 Host Environment

A configuration file may be used to set compiler options. This configuration file should be called `proton.cfg` and be installed in the same directory as `icc`. Common compiler options can be placed in this file to override the current default options.

The following is an example of a valid `proton.cfg` file:

```
# Change default optimization
-O0
# Change strictest structure member alignment to 8 bytes
-Zp8
```

In the configuration file a '`#`' character causes the rest of the line to be treated as a comment and ignored. Almost any argument you would normally enter on the command line to the compiler can be placed in the `proton.cfg` file. There are a few restrictions. The `-#` option should not be specified within a configuration file. Also, there is no special handling of single and double quoted arguments.

It is also possible to specify arguments to the compiler via response files. You use @myfile to effectively insert the contents of myfile into the compiler's command line. For example, given the following contents of myfile:

```
-O
-ip
common.o        # this module contains common code
```

The command:

```
icc -o hello @myfile hello2.c
```

would be equivalent to:

```
icc -o hello -O -ip common.o hello2.c
```

The rules for the contents of response files are the same as those for the `proton.cfg` file. You may specify one or more response files on the same *command* line.

6.0 Known Limitations

This section alerts you to problematic behaviors that are likely to be fixed in subsequent releases.

- In the transition dialect (`-Xt`), the compiler does not correctly expand recursive macro invocations when there are unbalanced parentheses in the macro definition. The following is an example of this type of macro invocation:

```
#define f(a) f(1 + a
#define apply(f, n) f(n)
apply(f, 2));
```

  The compiler issues a syntax error when the `apply` macro is expanded. Such statements are handled correctly in the other language dialects.

- When the K&R dialect (`-Xk`) option is in effect, invoking a multiply recursive macro definition, such as the following, results in repeated errors for macro recursions until the error limit is reached. The expansion of this macro is handled appropriately in the other dialects.

```
#define sqrt(x) ((x)<0 ? sqrt(-x) : sqrt(x) )
```

- The compiler issues an incomplete type message for the following code construct in all dialects except the `-Xk` dialect:

```
int a[10];
main()
{
    extern int a[];
    return sizeof(a);
}
```

  The second declaration of variable `a` hides the first declaration.

- Section 3.3.16 of the ANSI C standard states "the order of evaluation of the operands [on either side of an assignment operator] is unspecified." Therefore, in an expression such as the following:

```
arr[index->field] = index++;
```

a compiler can increment the `index` variable before or after its use on the left side of the assignment statement. This compiler increments `index` before its use on the left side. If you want the compiler to increment the variable after its use on the left side, you can change such statements as follows:

```
arr[index->field] = index;
index++;
```

- If a variable or function is declared incompatibly in two different translation units, and those translation units are compiled together using `-ip` for multi-file inlining, the compiler may abort with an assertion failure. For example, if file a.c contains:

  ```
  extern int i:
  ```

  and file b.c contains:

  ```
  unsigned int i = 17;
  ```

  then `i` is declared incompatible in those two translation units.

- It is not possible to detect floating-point NANs with code such as the following:

  ```
  if(! ( x < 0 ) && !( x > 0.0 ) && !( x == 0.0 ) )
         /* Where x is a NaN */
  ```

  The only way to detect a NAN is to examine its bit pattern.

- If an ASM macro is being expanded, none of the templates match the parameter list, and at least one constant or identifier parameter is provided, the parameter list will not be pushed on the stack in the proper order for a standard function call. This problem can be avoided by making sure there is a template match for each ASM macro expansion.

- This compiler supports 64-bit long doubles (8 bytes), which conflicts with the use of any libraries that expect 80-bit long doubles (12 bytes). The use of long doubles with this compiler may break some of the standard library routines.

- When using the `-ql` switch for ELF binaries, the line numbers are not maintained correctly for blank lines when switch statements are active and your final results will be off based on the number of blank lines found. COFF binaries do not have this problem. However, there are some differences in the way line numbers are generated between cc and icc. For instance, cc generates a line number for break, while icc does not. The line numbers for the other lines are the same, however, as icc simply skips that line number.

- A postscript version of the User's Guide is included with this distribution. The online release notes state that this file can be read with a postscript reader such as ghostview. This is incorrect. This file can be printed on a postscript printer, but not viewed with ghostview. To view the file with ghostview, you must remove all but the last of the 21 %%EOF lines from the file using your favourite text editor.

199876 PAT 0987654321