OPENSTEP 4.1 Release Notes: Compiler

This file contains developer release notes for the Windows and PDO compilers shipped with OPENSTEP Enterprise Release 4.1.

In this release, the compilers are based on the GNU C compiler version 2.7.2.

New Features

- If the name of your source file ends in **.cc**, **.cxx**, **.cpp**, or **.C**, **gcc** will attempt to compile your program with the C++ compiler. Similarly, if the name of your source file ends in **.mm** or **.M**, **gcc** will attempt to compile your program with the Objective-C++ compiler.
- The Objective-C++ compiler is now much more useable than the one that was included with PDO 4.0.

The -Wmost Compiler Flag

The **-Wmost** compiler flag is equivalent to FSF's **-Wall**, except that it doesn't turn on **-Wparenthesis**. **-Wmost** also suppresses warning messages about inline functions and static constants that are not actually used. This flag is for internal use and is not officially supported. It's definition may change in a future release.

Known Bugs and Limitations

The following bugs or limitations are worth noting for the GNU C and C++ compilers for this prerelease.

- Constant strings (both **char *** and NSString) should be 7-bit only. Unless constant strings are 7-bit, your code will be non-portable as compilers will

deal with 8-bit strings in a machine-dependent encoding.

- **Objective-C++ global constructors**. The Objective-C++ compiler sometimes ignores global constructors; they don't always get called.
- **Random name given to executable by default** (66861). If you create an executable and don't use the **-o** flag to explicitly tell **gcc** what to name it, **gcc** will most likely give it a random name.
- **Using pipes to communicate between compiler passes** (61306). The **-pipe** flag does not work.
- **-ObjC++ requires -lstdc++ when using C++ streams on PDO** (69156). When compiling C++ programs that use C++ streams with **gcc** on PDO platforms, if you specify the **-ObjC++** flag you must also specify the **-lstdc++** flag. So, for example, a program "foo" that uses **cout** (and therefore includes **iostream.h**) would be compiled using **gcc** as follows:

```
gcc -ObjC++ foo.cc -lstdc++
```

- **__declspec(dllexport) __stdcall doesn't work** (69194). On Windows NT, functions that are declared as **__declspec(dllexport)__stdcall** aren't handled properly. This may affect some Windows header files that you include in your programs.
- **The compiler sometimes tries to create a library instead of an executable** (69087). This happens on Windows NT when a function is declared as **__declspec(dllimport)** (perhaps in a header file), but the function is actually defined in the file being compiled. The workaround is to remove the offending **__declspec(dllimport)**.
- **-Wno-precomp flag isn't supported** (63746). On Windows NT and on PDO platforms, **gcc** refuses to do anything when the **-Wno-precomp** flag is specified. Currently, the precomp option isn't supported.
- **Reporting bugs** (68122). In some cases, when the compiler detects an internal error, it prints a message requesting that you send a bug report to some electronic mail address. Please disregard this message. All compiler bugs should be reported to the appropriate channels at NeXT Software.
- **cc -MM Foo.mm produces Foo.mm.o** (40491). Compiling a file that has either a **.M** or **.mm** extension with **-M** or **-MM** will produce a file whose name is the same as the source file with the addition of **.o**. The original extension is not stripped off before constructing the name of the **.o** file.
- **-traditional-cpp acts differently on Mach than on NT or PDO** (60175). On

Windows NT and on PDO, **-traditional-cpp** changes the behavior of the preprocessor in the same way that **-traditional** does. This can result in compilation errors when recompiling an existing NEXTSTEP project on Windows NT or on PDO.

- **Inconsistent function declarations involving stdcall produce unexpected results** (69506). On Windows NT, if a function is forward-declared to be **stdcall** but not declared to be **stdcall** in the actual function definition, the compiler will emit code to pop the arguments off the stack, but won't adjust the function name.
- **The linker complains about objects exported as CONSTANT** (70212). On Windows NT, if you're building a framework and you create your own DEF file for it, defining exported objects as CONSTANT will produce a warning from the linker advising you to use the word DATA instead. If you substitute the word DATA for CONSTANT in your DEF file, some or all of your objects won't be exported correctly; the linker will be unable to find them. As a workaround, simply leave the declarations CONSTANT and ignore the linker warnings.
- **-static option causes linking to fail** (70326). The **-static** and **-dynamic** compiler flags are meaningless on Windows NT, and shouldn't be used.
- **Programs that use posing sometimes crash** (72283). If a program compiled with the Objective-C compiler contains an object that accesses its superclass while posing on a category, it will crash. To workaround this problem, either rewrite your code so that it doesn't do this, or compile the program with the Objective-C++ compiler (to do this, use the **-x objective-c++** switch).
- **Objective-C++ is broken in PDO4.0 on Solaris** (69089). In this release of PDO, **gcc** doesn't compile Objective-C++ code correctly on Solaris machines. However, you can compile both Objective-C and C++ code—providing that they are in separate source files—with **gcc**.
- **Static constructors can't be used for run-time class initialization** (54831). The PDO compiler can't apply a user-defined constructor to a global or static C++ object and send an Objective-C message in the same file. To work around this problem, eliminate the constructor, the global, or the Objective-C code.