3.3 Release Notes: Compiler Tools

This document contains release notes for the 3.3, 3.2, 3.1, and 3.0 releases of the Compiler Tools. Items specific to the 3.3 release are listed first, and the Release 3.2, 3.1, and 3.0 notes follow.

This file contains information about the following topics:

- The NeXT GNU based assemblers
- The NeXT Mach-O link editor
- Mach-O object tools (**nm**, **otool**, and so on)

Numbers in parentheses following command names, as in **rld**(3), refer to Unix manual page sections.

Notes Specific to Release 3.3

New Feature

The major feature for Release 3.3 is a true Mach-O assembler, available for all platforms. See ªThe NeXT Mach-O GNU-based Assemblerº for more information.

Problems Fixed in Release 3.1

These problems have been fixed in Release 3.3:

Reference: 37610

Problem: Objective C messages are counted twice when profiling.

Description: If a method is compiled for profiling, it gets counted by two separate mechanisms, resulting in a double count.

Reference: 41317

Problem: The assembler for HP 9000 systems does not support cache control hints.

Description: You can't get cache control hints from an instruction like LDCWS.

Notes Specific to Release 3.2

Known Problems

These new bugs have appeared since Release 3.1.

Reference	37610
Problem	Objective C messages are counted twice when profiling.
Description	If a method is compiled for profiling, it gets counted by two separate mechanisms, resulting in a double count.

Notes Specific to Release 3.1

New Features

The major feature for the 3.1 release is the support for multiple architectures and cross development. The supported processor architectures are m68k and i386. The functionality of the 3.1 release is the same for all supported architectures as it was previously for the m68k architecture. The 3.1 release also adds the new functionality of multiple architecture files (ªfatº files).

- By default all tools target for the host they are executed on. Target architectures

are specified by **-arch** *flag* to the tools. More than one target architecture can be specified. Tools that operate on object files (that don't create them) can also have **-arch all** specified to select all architectures. See the man pages for the specific tools for more details on how it operates on fat files.

The -*o* option to *otool*(1) is now fully working along with printing of the sections __protocol, __string_object and __runtime_setup in the __OBJC segment.

Bugs Fixed in Release 3.1

These bugs have been fixed in Release 3.1:

Reference	27954
Problem	Some names in m68k assembly code appeared to behave as reserved words.
Description	Two-character symbol names beginning with `b', `d', or `i' were assumed by the assembler to indicate the m68k pseudo-registers `bc', `dc', or `ic', masking the desired use of the symbol name.
Reference	23178
Problem	Compiling with optimization produced some incorrect result with **float**s.

Description	An assembler bug caused comparison of small floating point numbers not to work in every case.
Reference	19664
Problem	**gprof** crashes when used on a program with bundles (NXBundle objects).
Description	The profiling tools would not work with code loaded from bundles.

Notes Specific to Release 3.0

These notes were included with the Release 3.0 version of the Compiler Tools.

This file contains information about the following topics:

- The NeXT GNU based assembler
- The NeXT Mach-O link editor
- Mach-O object tools (nm, otool, and so on)

The major change is that the assembler is much better. The other major feature is a new assembler manual. The assembler manual documents all features of the

assembler (previously no assembler manual existed).

The NeXT GNU-based Assemblers

The NeXT assembler is now a fully supported program and no longer viewed just as a back end to the compiler code generator. Also all outstanding bugs from the 2.x releases have been fixed. Exhaustive testing of all the instructions and operands of the assembler has been done resulting in a much more robust assembler. A UNIX manual page now exists for the assembler.

Major New Features

- Simple user defined macros (see **.macro**, and so on)
- Conditional assembly (see **.if**, and so on)
- C language style assembler expressions with C precedence
- All m68k mmu instructions are now supported
- All previous warnings are now treated as errors and cause the assembler to exit and not create an object file when an error is encountered (errors such as unknown pseudo ops, mismatching operands to machine instructions, and so on)

New Documentation

- Assembler manual (available on-line in **/NextLibrary/Documentation/NextDev/Assembler**)
- *as*(1) Unix manual page

The on-line assembler manual is indexed for use in Digital Librarian. A target for the manual is on the NextDeveloper bookshelf (**/NextLibrary/Bookshelves/NextDeveloper.bshlf**).

Notable Bug Fixes

- For the m68k, floating-point instructions that allow immediate operands were handled incorrectly. Previously an instruction like ªfmuls #2,fp0º would not place a 2.0 in fp0 the binary bit pattern of 2 (for double formats, like fmuld, the immediate value would be trashed). The new assembler now treats this immediate like all others, as an integer and then converts it to a floating-point value. A new feature has also been added so that bit patterns can be used for floating-point immediate values by prefixing with 0b, which then causes the rest of the value to be assumed as hexadecimal number.

Known Bugs

- 68k assembler does not handle packed floating-point immediates. This is treated as an error.

- The logical operators && and || are not implemented in expressions and can't be used.

The NeXT Mach-O link editor

New Features

- The new **-ObjC** flag provides objective-C linking semantics with respect to libraries. When this flag is present all library members for the specified libraries that define an objective-C class or category are loaded from the library. This solves problems with classes not being linked into the application and getting errors from the objective-C runtime when the class is used. This should be used whenever building a NeXT application and is the default in Project Builder. With this feature a library that contains objective-C can no longer be specified more than once (this includes libsys_s.a which is automatically included by cc(1) so it should not be specified in addition to that or multiply defined symbols will result).

- The new **-all_load** flag provides a way to link in all the members of the specified libraries. When this flag is present all library members for the specified libraries are always loaded from the library. This solves problems with respect to the use of *rld*(3), *objc_loadModules*(3) and NXBundles where the application wants to make available all of the library routines to the code it dynamically loads. This

provides a more general solution that which is provided with the ª-u libsys_sº like flags with respect to the NeXT supplied shared libraries. This solution works for all types of libraries including those not supplied by NeXT.

- The new **-m** flag allows multiply defined symbols with a warning instead of treating them as a hard error. In this case the first symbol is used for linking and is the value of the symbol. This should only be used as a stopgap measure when the source code is unavailable to be fixed.

- The new function void **rld_forget_symbol(**NXStream ***stream*, const char€***symbol_name***)** was added for use by Franz Lisp to better support its implementation of foreign functions (see the *rld*(3) Unix manual page for details).

- The *rld*(3) package now works more automatically with the debugger *gdb*(1) for debugging code that is loaded in to a program by the program itself. If the program loads into itself with *rld*(3) then the output_filename parameter of *rld_load*(3) and the use of gdb's add-file command is no longer needed. These are still needed if the program has a program other than itself loading objects into it.

Mach-O Object Tools

New Features

- The tool *strip*(1) now allows all combinations of its flags to better support stripping executables for later use with *rld*(3). This allows much easier control of the api that the executable wants to provide to the objects that it will load with *rld* and it will not have to publish symbols that are not part of its API. For example an executable that wishes to allow only a subset of its global symbols but all of the shared libraries globals to be used would be stripped with:

```
strip -s api_symbols -A executable
```

  Here the file *api_symbols* would contain only those symbols from the executable that it wishes the objects loaded with *rld* to have access to. The other major change to *strip*(1) was to properly handle relocation entries and never strip them. Thus, with the addition of the **-u** flag to save all undefined symbols, *strip*(1) allows an object (made up of other objects) to strip its global name space so not to collide with other symbols. For example an object that is made up of a number of other objects that will be loaded into an executable would built and then stripped with:

```
ld -o relocatable.o -r a.o b.o c.o
strip -s api_symbols -u relocatable.o
```

  which would leave only the undefined symbols and symbols listed in the file *api_symbols* in the object file. In this case *strip*(1) has updated the relocation entries to reflect the new symbol table.

Notable Bug Fixes

- The m68k disassembler in *otool*(1) now correctly disassembles all m68k instruction from all of the m68k chips including the mmu instructions in the 030 and 040. Also the disassembly format exactly matches the assembler format for all instructions and operand formats.