3.3 Release Notes: C Preprocessor

A new implementation of the C Preprocessor, **/lib/cpp-precomp**, was introduced in the 3.0 release to support precompiled headers. It is invoked by **cc**(1) whenever the output will be fed to the C or Objective-C compiler. There were no changes for Release 3.3.

Notes Specific to Release 3.2

New Features

The following new features have been added to the preprocessor since Release 3.1.

· **Support for RTF   source code.** Both cpp and cpp-precomp now strip RTF, if present, from files passed to cpp (and thus cc).   The RTF information (for example, fonts, colors, help links and markers, underlining, etc.) is stripped and only the ASCII information is left for normal preprocessing. This means that any part of source code can be made rich.

Additional information about using Edit to create and edit rich source code can be found in the **Edit.rtf** release note. However, tools like **grep** and **sed** may produce spurious results on RTF code (although the ProjectBuilder ªFinderº mode can be used to search for text strings in both rich and non-rich source code).

· **Automatic byte-swapping of precompiled headers.** In 3.1, by use of the **-arch** flags, precompiled headers could be generated for use during native compilation, cross compilation, and compilation for multiple architectures. However, precompiled headers generated on one architecture were often not usable during compilation on a host of a different architecture. This was a byte-swapping issue, which is handled in 3.2 by on-demand byte-swapping of precomps before checking and using them. Note that there is an approximate 20% slow-down in the usage of a precomp due to the time necessary to do this byte-swapping. Therefore, you should always attempt to build your precomps on the host architecture on which you will be doing the majority of your compilations.

Notes Specific to Release 3.1

Three distinct changes have been made regarding cpp-precomp for release 3.1:

· The standard GNU C preprocessor, **/lib/cpp**, is *no longer used* when **cc** is invoked with the **-E** option; **cc -E** always produces exactly what the compiler would see.

Note that this may be problematic when invoking **cc -E** (as opposed to **/lib/cpp**) on something other than source code (for example, a Makefile). In this case, the **-traditional-cpp** option, described below, should also be passed.

- The **-Wprecomp** option has now become the default, but only causes warnings in the rare cases when a precompiled header (or precomp) can not be used. The **-Wno-precomp** option can be used to suppress these warning messages if desired.

- The default search path for cpp-precomp has been brought into line with the default search path for cpp. Most notably, this removes **/usr/local/include/** from the standard search path. Developers are encouraged to place local, shared headers in **/LocalDeveloper/Headers/** instead. As of release 3.1, the default search path for cpp-precomp is now:

```
/NextDeveloper/Headers
/NextDeveloper/Headers/ansi
/NextDeveloper/Headers/bsd
/LocalDeveloper/Headers
/LocalDeveloper/Headers/ansi
/LocalDeveloper/Headers/bsd
/NextDeveloper/2.0CompatibleHeaders
```

By adding **-I/usr/local/include** to the cc(1) command line, headers that were previously included from that directory can again be used in this release.

Notes Specific to Release 3.0

Precompiled Headers

Precompiled headers improve compile time and reduce symbol table size. A separate file, **PrecompiledHeaders.rtf**, describes how to install and use them.

Switches

The new preprocessor supports the following new switches.

-precomp	Produce a precompiled header.
-no-precomp	Do not use precompiled headers.
-traditional-cpp	Use the standard GNU preprocessor instead of cpp-precomp, causing precompiled headers to be ignored.
-Wno-precomp	Do not give warnings when precompiled headers are found unusable (by default, such warnings are given).

Known Problems

The new preprocessor does not yet support the following switches.

-fno-asm	Do not recognize **asm**, **inline**, or **typeof** keywords (note that **-ansi** also implies **-fno-asm**).
-pedantic	Issue warnings required by the ANSI C standard.
-T	Process ANSI standard trigraph sequences.
-d	Produce a list of #define commands.
-P	Inhibit generation of # lines.
-H	Print the name of each header file used.
-dM	Output only a list of macro definitions that are in effect at the end of processing.
-dD	Pass all macro definitions into the output.
-dN	Like **-dD**, except macro arguments and contents are omitted.

Use of the above switches must be accompanied by the **-traditional-cpp** switch.

Other Changes

The output of the new preprocessor sometimes has token spacing (within a line) that is different than the GNU preprocessor. This conforms to the ANSI standard and should be transparent to users of **cc**.