

Preliminary 3.0 Release Notes: Objective-C++ Compiler

Objective-C++

The Release 3.0 Objective-C++ compiler is based on version 1.39.1 of the GNU Compiler which directly supports version 2.0 of the C++ language, as specified by AT&T (the Release 2.0 compiler was based on version 1.36.4 of G++). The GNU compiler has been extended to recognize Objective-C constructs within C++ source files.

New Features

Integrated Compiler Driver

The Release 3.0 compiler driver `/bin/cc` handles C, Objective-C, C++ and Objective-C++ source files. The C++ compiler driver program `/bin/cc++` is no longer needed (but is still present for compatibility). The compiler driver determines the appropriate language from the suffix of the source files or command line options. Files ending in `.c` are taken to be C source files, `.m` indicates an Objective-C source file, and `.cc`, `.cxx`, and `.C` are recognized as C++ source files. The `-ObjC` flag specifies that source files should be considered to be Objective-C source files regardless of their extension. Similarly, the `-ObjC++` flag specifies

that source files should be considered to be Objective-C++ source files. The `/bin/cc++` driver is simply a shell script which calls the `/bin/cc` driver with the `-ObjC++` flag.

New Constructor and Destructor Sections

The Release 3.0 compiler no longer uses the **collect** program to gather references to C++ constructors and destructors (which requires linking twice). The compiler now automatically places these references in the new **__constructor** and **__destructor** sections of the **__TEXT** segment, which are automatically coalesced by the linker. Link time should be substantially decreased.

C++ constructors are called from startup routines in **crt0.o**, and destructors are registered to be called in the reverse order when the program terminates using **atexit()**. The archive library `/usr/lib/libc++.a` present in Release 2.0 is no longer needed.

Known Limitations

Placement new syntax not supported

The *placement new* syntax is not supported. For example:

```
T *tPtr = new(2, f) T;
```

Thus, it is not useful to implement the function `operator new()` with additional arguments, since there is no way to specify the additional arguments using the `new` operator.