Q: The gdb debugger refuses to display the value of my variable ªx.º Following is the source code and the gdb output:

```
- convert:sender {
        char    *fName;

        foo = NO;
        [self enableConvert];
        fName = [myText stringValue];
        <--- break here

(gdb) print fName
No symbol "fName" in current context.
```

A: When code is compiled with the -O flag, the compiler can take liberties to optimize away certain source code variables. This is accomplished by careful juggling of registers and order of expression evaluation. For example, stack variables which never have their

address taken and are used only across a very few instructions can ªdisappearº without a trace. Try compiling your code without optimization and see if this eliminates some of these surprises. The standard ªmake <appname>º rules do invoke optimization as shown by the -O flag:

```
myhost> cc -g -O -Wall  -c QCombo.m -o obj/QCombo.o
```

The standard ªmake debugº rules do not:

```
myhost> cc -g -Wall -DDEBUG  -c QCombo.m -o debug_obj/QCombo.o
```

The ªDebugº button in ProjectBuilder does not assume ªmake debug.º You have to add ªdebugº to the Args text field to pass it as an argument to the make process.

The man page for **cc** contains more detail over how one can control compiler optimizations; a small section is repeated here:

-O Optimize. Optimizing compilation takes somewhat more time, and a lot more memory for a large function.

Without -O, the compiler's goal is to reduce the cost of compilation and to make debugging produce the expected results. With -O, the compiler tries to reduce code size and execution time. Some of the -f options described below turn specific kinds of optimization on or off.

-g Produce debugging information in GDB format.

Unlike most other C compilers, GNU CC allows you to use -g with -O. The shortcuts taken by optimized code may occasionally produce surprising results: Some variables you declared may not exist at all; flow of control may briefly move where you didn't expect it; some statements may not be executed because they compute constant results or their values were already at hand; some statements may

```
execute in different places because they were moved out of loops.
Nevertheless, this makes it possible to debug optimized output if
necessary.
```

The fastest way to pursue one missing variable would be to recompile the troublesome source file from the shell, without the -O flag. Then relink, and debug the new executable.

QA843

Valid for 1.0, 2.0, 3.0