Q: When a link is in progress, it seems to take over my machine. It's sluggish to activate another application and do any other work. Is there a cure for this?

A: From a scheduling point of view, there's not much you can do to convince the kernel to give fewer resources to the linker. The linker is designed as the kernel's Dream Program by asking for certain size blocks with a certain regular pattern.

However, the linker is I/O bound. It's moving a lot of bits back and forth so anything you can do to make the travel time faster is a win: faster hard disks, local access to files rather reading files all over NFS, etc. Constructing the symbol table is a large resource hog; linking without symbols is much faster than with. However, often you need the symbols for debugging while in development. You can use the ª-xº option to **ld(1)** to remove symbols from the sections of your code that you're not debugging currently, and then compile new sections with symbols so that only the parts you are working on have symbol information. This is assuming you are concentrating your debugging efforts on only the changed parts.

To do this, you need to do the following for each of your .o files:

```
localhost> ld -r -x foo.o
localhost> mv a.out foo.o
```

Then when you modify your sources, **make(1**) just recompiles the object file with full symbols if you have the ª-g ºflag on.

NeXT's software engineers use systems with the smallest supported memory configurations to ensure they are developing software that is usable on the base configuration. We have implemented compile servers to minimize the impact of large compiles on these local machines. The compile servers maintain the sources locally and all compiles take place there.

QA737

Valid for 1.0, 2.0, 3.0