# disown (detach a job from the shell)

When you run a graphical program in the shell and close the terminal window, the program is terminated. This can be dangerous because you may lose unsaved program data. It doesn't make any difference whether the process is running in the shell's foreground or background, it's terminated either way.

When you want to avoid this, use the shell builtin *disown*. It detaches a process from the shell so that it isn't terminated when the shell exits. Invoke *disown* like this:

*disown %JOBNUM*

The argument %JOBNUM is the job number of the process you want to disown, such as %1. Remember that you can display the job numbers of any processes running in the shell using the *jobs* command (see Chapter 1). The disowned process will not appear in the job list any longer, as it has been detached from the shell. If you call disown without arguments, it will detach the currently active job.

Let's try this out by disowning the program *xeyes* (you can use any graphical program for this experiment):

*xeyes &*
*disown*

Now when you close your terminal window, the program *xeyes* will keep running. Note that some shells default to this behavior, so you may not need to use *disown* on your system. You may need to use it on other systems, though, so be careful to avoid data loss.

You can start a program in a disowned state by invoking it like this:

*(myprogram &)*

This is as if you had entered *myprogram &* and *disown*. The parentheses instruct the shell to run *myprogram* in a subshell which is beyond the shell's job control. I always run graphical programs like this for safety reasons.