Signals

/run/media/lijunjie/资料/Learning Video/writing-running-fixing-code/03_testing-and-debugging/03_debugging-tools/08_signals_instructions.html

Whenever your program receives a signal, *gdb* will stop the program and give you control. There are three particularly common signals that come up during debugging. The first is SIGSEGV, which indicates a segmentation fault. If your program is segfaulting, then just running it in *gdb* can help you gather a lot of information about what is happening. When the segfault happens, *gdb* will stop, and your program will be on the line where the segfault happened. You can then begin inspecting the state of the program (printing out variables) to see what went wrong.

Another common signal is SIGABRT, which happens when you program calls *abort()* or fails an *assert* . As with segfaults, if your code is failing asserts, then running it in gdb can be incredibly useful—you will get control of the program at the point where *assert* causes the program to abort, and (after going up a few frames back into your own code), see exactly what was going on when the problem happened.

The other signal that is useful is SIGINT, which happens when the program is interrupted — *e.g.* , by the user pressing Control-c (inside emacs, you have to press C-c C-c: Control-C twice). If your program is getting stuck in an infinite loop, you can run it in *gdb* , and then after a while, press Control-c. You can then see where the program is, and what it is doing. You are not guaranteed to be in the right place (you may interrupt the program before it gets into the infinite loop), but if you wait sufficiently long, you will typically end up where you want. You can then see what is happening, and why you are not getting the behavior you want.