Debugging Principles

The Fundamental Principle of Confirmation

Fixing a buggy program is a process of confirming, one by one, that the many things you *believe* to be true about the code *are* true. When you find that one of your assumptions are *not* true, you have found a clue to the location (if not the exact nature) of a bug.

Start small

At the beginning of the debugging process, you should run your program on easy, simple cases.

Use a top-down approach

For example, it is better to initially step over a function rather than step through it. You perform the call and then inspect the values of variables that depend on the results of the call in order to see whether or not the function worked correctly.

Use a debugging tool to determine the location of a segmentation fault

The debugger will tell you the line of code at which the fault occurred. You can then get additional use information by invoking the dubgger's *backtrace* facility, which displays the sequence of function calls leading to the invocation of the function in which the fault occurred.

Determine the location of an infinite loop by issuing an interrupt

If you suspect your program has an infinite loop, enter the debugger and run your program again, letting it execute long enough to enter the loop. Then use the debugger's interrupt command to suspend the program, and do a backgrace to see what point of the loop body has been reacher and how the program got there. (The program has not been killed; you ca reusme execution if you wish.)

Debugging Principles (cont)

Use binary search

Suppose you know that the value stored in a certain variable goes bad sometimes during the first 1,000 iterations of a loop. One way that might helop you track down the interation where the value first goes bad is to use a *watchpoint*. Another approach is to use binary search. You'd first check the variable's value at the 500th iteration; if it is still all right at that point, you'd next check the value at the 750th iteration, and so on. As another example, suppose one of the source files in your program will not even compile. The line of code cited in the compiler message generated by a syntax error is somtimes far from the actual location of the error, and so you may have trouble determining thta location. Binary search can help here: You remove (or comment out) one half of th ecode in the compilation unit, recompile the remaining code, and see if the error message persists. If it does, then the error is in that second half; if the message does not appear, then the error is in the half that you deleted. Once you deteremine which half o the code contains the bug, you further confine the bug to half of that portion, and keep going until you locate the problem.

C Tips

Ensure parameters passed by address are not null pointers before dereferencing them (can crash program).