Lecture 24: Debugging and *gdb*

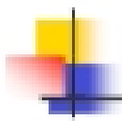
Kenneth M. Anderson
Software Methods and Tools
CSCI 3308 - Fall Semester, 2003

Today's Lecture

- Debugging
  - The concept
  - Terminology
  - Types of Debuggers
- gdb - The GNU Debugger

Debugging

- Debugging is the process of finding faults in a program after a failure has been detected
- Typically
  - a program fails a test case (or a bug is reported)
  - a programmer is given the test case, checks out the associated source code, and "debugs" the program until the fault has been corrected
  - the program, having been fixed, passes the test case and no longer exhibits the failure



Finding Faults

- The hard part of debugging is locating faults
  - Once a fault is located, it is typically straightforward to fix
- Having a failing test case is helpful
  - because the fault must be in some part of the code that was executed by the program before the failure occurred
  - The entire program does not have to be examined

Bridging the Gap

- One problem with faults is that they are not necessarily located "near" their associated failure
- Therefore, "bring the failure close to the fault"
  - the idea being that if we do, less code needs to be examined to find the fault
  - Thus, we often insert "print" statements into code to force the failure to appear as soon as possible after the fault
  - Taken to the extreme, a programmer should be able to see the value of any variable at any time during a program's execution

Debugging Tools

- This is the purpose of debugging tools
  - called "debuggers"
- A debugger allows a programmer to monitor the internal state of a program while its executing
- Two types of debuggers
  - Interpretive
  - Direct Execution

Types of Debuggers

- Interpretive debugger
  - works by reading a program and simulating its execution one line at a time
- Direct Execution Debugger
  - works by running the actual program in a special mode where the debugger can read and write the program's memory

Two styles of use

- Line-at-a-time
  - A programmer loads a program into the debugger and "steps" through the program one line at a time.
  - The debugger stops the program after each line and gives the programmer a chance to check the program's variables to see if it is operating correctly
- Breakpoints
  - A programmer loads a program into the debugger and specifies "breakpoints" at various locations in the program
  - The program runs until it hits a breakpoint

More on Breakpoints

- How are breakpoints useful?
  - If you think you have a function that might have a fault
    - set a breakpoint at the beginning of the function
    - set another breakpoint at the end of the function
    - if a program's data is correct at the beginning of the function but incorrect at the end, then there is a fault in the function
  - They also allow you to skip over "init" code and debugged code quickly; letting the programmer focus on finding the fault

Implementing Breakpoints

- How do debuggers implement breakpoints?
- Interpretive Debuggers essentially execute as a while loop

```
while (...)
    Read the next line of the program
    Is there a breakpoint on this line?
        Yes, stop and print a prompt
    Execute this instruction
end while
```

- Direct Execution debuggers implement breakpoints by "cutting-and-pasting" the executable instructions of the program itself
  - that is, they insert instructions into the program that notifies the debugger when a breakpoint has been hit

Editing Variables

- Debuggers let the programmer explore "what-if" scenarios
  - You can execute a program to a certain point, and then alter the values of the program's variables
  - You can thus explore unusual cases in the code
  - Plus, if a bug occurs, you can correct an incorrect value and see how far the program goes before it encounters another fault

Advantages

- Advantages of interpretive debuggers
  - Easier to program the debugger
  - Safer, a program cannot crash the machine (just the debugger)
- Advantages of direct execution debuggers
  - Faster
  - More accurate, the actual program instructions are being executed