CS 403: Lab 4: Debugging

Getting Started

1. Boot into Linux or Windows, it's your call.

2. Download RAD1D_buggy.tar from the course website[1]. This is the model problem minus the subroutine you're writing for PS2 and with a couple of bugs thrown in. Remember, to untar the file, type tar svf RAD1D_buggy.tar.

3. Go to the appropriate section below and fix the errors!

Linux Debugging with gdb

1. Cd into RAD1D_buggy and fix Makefile so that it will build an executable that can be debugged.

2. Look at basic.cmnd and figure out what it is telling the program to do. In particular, look at C.txt, the initial conditions (open with nedit or type more C.txt). Build the program and try running it. What happens?

3. Let's find the error. First, open the .c files so that you can see what's going on (it is possible for gdb to show you code, but it's much easier to look at it in an editor. Then type gdb rad1d basic.cmnd. This starts gdb and tells it that you'll be working with rad1d using the input basic.cmnd. Since the error happens when the program is trying to read C.txt, find the subroutine that handles this (hint: it's in io.c). When you have the subroutine name, type br name where name is the name of the subroutine. To run the program to that point, type run.

4. I won't tell you how to find the error, that would be too easy. However, I will give you some important gdb commands:

[1] www.cs.cornell.edu/Courses/cs403/2002sp

Command	Arguments	Description
n		go to next statement in current subroutine
s		step into the subroutine
finish		step out of subroutine
u		go until next line is reached (try this at the end of a for/while loop to avoid going through each iteration.
p	name	value of variable name
p	name=expr	changes value of name to expr

5. You should now be able to read C.txt, but you're not done! There is another bug. To see this, compare the output file (Cfinal.txt) to the C.txt. Find the last error.

Windows Debugging with VizStudio

1. Start-up VizStudio and create a new project (I'd call it rad1d). Add the .c and .h files, then build. To run, click on the red exclamation point. What happens? Most likely, it won't run. This is because your executable is placed in the Debug directory and it is run from there. Thus, when you type "basic.cmnd," the program can't find it. There are two ways to fix this. One, you can move all of your input files (C.txt, basic.cmnd) into the Debug folder. Or, you can set the working directory to wherever you data is. To do the latter, open the settings panel (under the Project menu). Type the path of your working directory in the working directory field. You can also set the program arguments to basic.cmnd, so you don't have to type it in every time. Now run it. What happens?

2. Let's find the error. VizStudio's debugger works a lot like db: we'll still set breakpoints and step through the code, but we'll do it graphically. Since the error happens when the program is trying to read C.txt, find the subroutine that handles this (hint: it's in io.c). When you find the subroutine, right-click on the first statement in this routine and select "set breakpoint" from the menu that appears. To start the debugger, select "start debug" from the "Build" menu. You step through the code by clicking on the buttons on the palette (hold the mouse over a button for a second, and the name of the button should appear). In VizStudio-speak, "step-over" takes you to the next command, "step

into" takes you inside a subroutine, "step out" finishes the routine and takes you to the next line after it. You can also click on a line of code to place the cursor there, and then click "run to cursor" to have the debugger run ahead to where you've set the cursor. Finally, the value of the variables in the current subroutine are in a table in a separate window. Good luck.

3. You should now be able to read C.txt, but you're not done! There is another bug. To see this, compare the output file (Cfinal.txt) to the C.txt. Find the last error.