Debugging

CS 247

University of Waterloo

cs247@student.cs.uwaterloo.ca

May 30, 2014

Overview

Debugging

As we begin to write increasingly complex programs runtime errors start to crop up. It is extremely important to test to find these errors and, once they have been found, fix them. This process is known as debugging.

Debugging Methods

A common method of tracking down errors is adding print statements that allow you to track the flow of control through the program at the values of variables at key points during the execution. This method is often quite effective, but as the size of your code grows it becomes less feasible and it will not always give enough information.

Debugging Methods

An alternative method is to use a tool which allows you step through the execution of your program. For C++ programs, the GNU Project Debugger(GDB) allows us to do this while also providing a number of useful ways to check the correctness of and interact with the program.

Compiling with Debugging Information

To compile your program with debugging information use g++ with the option -g. GDB can be run with programs that do not include debugging information, but without the information it loses a lot of its usefulness.

Running GDB

GDB can be invoked with the command:

`gdb <exec>`

Which will start GDB with $<$ *exec* $>$ as the current executable file. You can also simply start GDB then select the file with:

`file <exec>`

GDB Commands

GDB has many capabilities which you can explore if you are interested, but here are some common commands for basic use.

Command	**Description**
run [args]	Run the program with the given arguments until it crashes or completes.
backtrace / bt	Print trace of current stack (list of called routines).
print < *var* >	Print value of < *var* >
watch < *var* >	Print a message every time < *var* > is changed.
break < *routine* >/< *line* >	Set breakpoint at routine or line of file.

GDB Commands Continued

Command	**Description**
step [n]	Execute next n lines. (steps into routines)
next [n]	Execute next n lines. (steps over routines)
continue [n]	Continue to next breakpoint [Skip next n breakpoints.]
set var $< var > < val >$	Sets the value of variable $< var >$ to $< val >$
quit	Exit gdb

The End