Debugging Tools

CS 485G-006: Systems Programming

Lecture 8: 5 Feb 2016

gdb

- **Gdb is the gnu debugger program**
  - Runs on most systems
  - Allows you to examine a running executable (e.g., a.out)
  - It can be used without the source code that created the executable
  - If you have the source code, it is best to compile the program with debugging information (symbol tables, etc) include:
    - gcc –g hello.c
  - To run gdb on an executable file:
    - gdb a.out
  - You will then be presented with a command line interface
  - Using the command line interface, you can get help by typing
    - help
    - help *subtopic* (where *subtopic* is one of the main subtopic areas listed by typing help). Examples include:
      - help data
      - help breakpoints
    - help *command* (where *command* is a gdb command). Examples include:
      - help print
      - help run

gdb - breakpoints

- Breakpoints are points in the program where you would like the program to stop so that you can examine what is going on (e.g., look at the contents of memory, registers, variables, etc).
- In the absence of breakpoints the program will run as normal until it completes or encounters and error.
- Typically you will want to set a breakpoint (near the point in the program where you think there is a problem) before you run the program within gdb.
- Breakpoints can be enabled and disabled.
- There are several ways to set a breakpoint. Examples include:
  - break *function_name*
  - break *linenumber,* or break *filename:linenumber*
  - break *address*

gdb useful commands

(running a program)

- run
  - Run the program within gdb
- continue
  - Continue the program after encountering a breakpoint
- step
  - Run the program until it encounters the next line in the source file.
- stepi
  - Run the program until it encounters the next machine instruction
- next
  - Like step, but continue through subroutines/procedure calls

gdb useful commands

(examining memory)

- x/*nfu* address
  - Examine memory or register at location *address*, where
    - n is the number of items to examine
    - f is the format to use when displaying values
      - s for strings
      - d for integers
      - u for unsigned integers
      - x for hexidecimal
      - a for address/pointer
      - f for floating point
    - u is the unit
      - b for byte
      - h for halfword
      - w for word
      - g for giant word
- print *exp*
  - Print the value of expression *exp*
- print */f exp*
  - Print the value of expression *exp* using format *f* (see format specs above).
- printf *string, expressions*
  - print the list of *expressions* using the c-style format string *string*
- display */f exp*
  - Print the value of expression *exp* using format *f* every time there is a break in execution

gdb useful commands

(examining registers)

- info registers
  - Print a list of the register contents
- Registers can be referenced in expressions by a well-known variable name that corresponds to the usual term used for the register. Variable names start with $. For example:
  - x/d $eax
    - Prints the contents of the eax register as a number
  - x/s $eax
    - Prints the strings starting at the address represented by the address held in the eax register
  - print $pc
    - Prints the address in the pc register

gdb useful commands

(misc commands)

- bt
  - Backtrace prints the contents of the current stack
- info frame
  - Print information about the current stack frame
- info args
  - Print information about the arguments passed to the current procedure
- info locals
  - Print information about the current local variables
- list linenumber
  - List the source code lines around the linenumber specified

Assembly language debugging

- **Setting breakpoints at any machine instruction**
  - break *0x400fe9
- **Disassembling code**
  - disassemble (abbreviate as: disas)
    - Disassembles the code around the current program counter
  - disassemble 0x400fe9, 0x400fff
    - Disassembles code from 0x400fe9 to 0x400fff
  - disassemble 0x400fe9, +10
    - Disassembles code starting at 0x400fe9 and going 10 more bytes
  - set disassemble-next-line on
    - After breaking, disassemble the next instruction
- **User defined commands**
  - Use the "define" command to create your own gdb commands
  - For example, to create your own stepi function you might define

```
define mystepi
   stepi
   disassemble
end
```

  - For more information, google for "gdb user defined commands"

gdb useful help/tutorial pages

- Documentation
  - http://web.mit.edu/gnu/doc/html/gdb_1.html
  - ftp://ftp.gnu.org/old-gnu/Manuals/gdb/html_node/gdb_toc.html
  - http://www.cs.hmc.edu/~geoff/classes/hmc.cs105.200901/labs/gdbinfo.html
- Tutorials
  - Google for “gdb tutorial”
  - Lots of good tutorials