CS 144 Section
GDB Tutorial

What is gdb?

- Portable debugger, runs on many UNIX-like systems.
- Works for a lot of programming languages like C, C++, Fortran etc.
- Allows tracing execution of programs, monitoring of functions and variables
- User can also alter execution, call functions, change values of variables explicitly.
- Lot less painful to debug errors like segmentation faults using gdb.

Prerequisites

- Make sure to compile program with “-g” flag.

  gcc -g <source file> -o <output file>

  (don't worry about this for your labs, your makefile includes this flag during compilation.)

- Linux should core dump on segmentation faults. Set

  ulimit -c unlimited

- Will get a “Segmentation fault (Core dumped)” message; creates a core file probably with the name core.pid

- Can analyze core file to determine cause of errors.

Start gdb

- Start debugger with program executable as argument

  gdb executable

- To analyze core,

  gdb {executable} {core-file}

- Use the *run* command to start execution of program, you can pass arguments too

  (gdb) *run arg1,arg2..*

- To restart a program running in gdb, use

  (gdb) *kill*

  and use the run command again.

Bugs?

- If buggy program, gdb presents useful information; code file, line number, and the call that caused the error.
- To find root of issue, need to step through code until you stumble upon the error.
- Useful set of commands with gdb ..

  (gdb) help

  - neat description of all gdb commands.

Useful commands

- *(gdb) bt*

  - *backtrace;* prints stack trace, will help know where exactly your program segfaulted.

- Can move to specific stack frames and inspect local variables, passed arguments.

  eg. *(gdb) frame 2*

  (gdb) info locals

  (gdb) info args

More commands...

- Set *breakpoints* to stop program at designated points

  - at a specified file-line pair,

  (gdb) b sample.c:35

  - at a specific function,

  (gdb) b func_name

- Program will pause every-time it reaches a breakpoint when running

  and prompt you for another command.

- Set *watchpoints* on variables; program pauses whenever variable is modified

  (gdb) watch var_name

Stepping through code

- Type the run command again once you have set breakpoints.
- Can proceed onto next breakpoint by typing

  (gdb) c

  - *continue*
- Can step into functions

  (gdb) s

  - *step;* executing 'just' the next line, also jumps into functions.
- (gdb) n

  - *next;* similar to step but doesn't show execution of every line of a function.

Printing values

- Can print values of variables, memory addresses of pointers, fields of structs etc.

  (gdb) p name

  (gdb) p (*emp).name

  (gdb) p list->next->next

- Lot more tricks – call, finish,where, delete, setting conditional breakpoints etc. - try *help* for more useful commands or online manuals

  http://www.cs.cmu.edu/~gilpin/tutorial/

  http://www.unknownroad.com/rtfm/gdbtut/gdbtoc.html