Debugging with the GDB debugger

- **Compiling your program.** For C/C++ programs, use the `-g` option for the gcc compiler.
- **Using the debugger.**
  - Starting your program under the debugger. Specifying commandline arguments.
  - Stopping your program on specified places and conditions. Setting breakpoints, setting conditional breakpoints, watching variables etc.
  - Stepping through a program: instruction at a time, line at a time, over functions etc.
  - Examining what has happened, when your program has stopped. Looking at the stack frames, values of variables etc.
  - Modifying variables in your program.
  - Attaching the debugger to a program that is already running!

GDB: Demo

All gdb examples and sample debugging sessions are in the lab examples at `C-examples/gdb` folder.

- Compile a sample program (function.c) without -g and with -g to show the difference
- gcc function.c && gdb a.out
- gcc -g function.c && gdb a.out

```
    (gdb) run
    (gdb) bt
```

- Notice that without -g we get no line numbers or source code shown in the debugger.

GDB: Break Points

- In the file `function.c` set a breakpoint at the populate function.

```
(gdb) break populate
   Breakpoint 1 at 0x40065e: file function.c, line 16.
```

- Lets see what is in the array so we can track down the problem

```
(gdb) run
  Breakpoint 1, populate (size=20, b=0x602010) at function.c:16
  16     for(i = 0; i < size; b++, i++){
(gdb) p size
  $4 = 20
(gdb) p b
  $5 = (int *) 0x602010
(gdb) p *b
  $6 = 0
(gdb) p *b@size
  $7 = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19}
(gdb)
```

GDB: Conditional Break Points

- Only break when you need to! Set a conditional break point.

```
(gdb) break populate
  Breakpoint 1 at 0x4006de: file function.c, line 19.
(gdb) cond 1 b == 0
(gdb) run
  Breakpoint 1, populate (size=20, b=0x0) at function.c:19
  19 count++;
(gdb) bt
  #0  populate (size=20, b=0x0) at function.c:19
  #1  0x0000000000400774 in not_buggy (size=20, b=0x0) at function.c:34
  #2  0x00000000004007ae in main () at function.c:42
(gdb)
```

GDB: Sample Sessions

- gdb/session0. Shows how to access built-in help from inside gdb.
- gdb/session1. Shows basic usage. Shows how to examine arrays.
- gdb/session2. Shows how to examine the stack trace after a segmentation fault.
- gdb/session3. Shows how to examine the stack trace from a core file that was dumped after program crashed.
- gdb/session4. Shows the usage of breakpoints.
- gdb/session5. Shows how to stop at a breakpoint only if certain condition is true. Also shows how to look at structures and manipulate pointers in the debugger.
- gdb/session6. Shows how to attach to an already running process to debug it.

Documentation

- The gdb debugger has extensive on-line help that can be accessed by typing in `help` at the gdb prompt.
- A two page reference card is available. (Check Amit's home page in the section *Handouts for Students*).
- The complete reference manual is available in HTML.

References

- http://darkdust.net/files/GDB%20Cheat%20Sheet.pdf
- http://www.gnu.org/software/gdb/documentation/