GDB debugger

gdb

To compile a program for use with gdb, use the '-g' compiler switch

- Add debug symbols and do not reorder instructions

Better graphical interfaces

- Most debuggers provide the same functionality
- gdb -tui
  - layout split, layout regs
- DDD: http://www.gnu.org/software/ddd/

Controlling program execution

run

- Starts the program

step

- Step program until it reaches a different source line.

next

- Step program, proceeding through subroutine calls.
- Single step to the next source line, not into the call.

continue

- Continue program execution after signal or breakpoint.

Controlling program execution

break, del

- Set and delete breakpoints at particular lines of code

watch, rwatch, awatch

- Data breakpoints
- Stop when the value of an expression changes (watch), when expression is read (rwatch), or either (awatch)

Printing out code and data

print

- Print expression
- Basic
  - print argc
  - print argv[0]
- print {type} addr
  - (gdb) p {char *} 0xbfffdce4
- print /x addr
  - '/x' says to print in hex. See "help x" for more formats
  - Same as examine memory address command (x)
- printf "format string" arg-list
  - (gdb) printf "%s\n", argv[0]

list

- Display source code (useful for setting breakpoints)

Other Useful Commands

where, backtrace

- Produces a backtrace - the chain of function calls that brought the program to its current place.

up, down

- Change scope in stack

info

- Get information
- 'info' alone prints a list of info commands
- 'info br' : a table of all breakpoints and watchpoints
- 'info reg' : the machine registers

quit

- Exit the debugger

Example Program

```
1   #include <stdio.h>
2   void sub(int i)
3   {
4         char* here[900];
5         sprintf((char *)here,"Function %s in %s", __FUNCTION__ , __FILE__);
6         printf("%s @ line %d\n", here, __LINE__);
7   }
8
9   void sub2(int j)
10   { printf("%d\n",j); }
11
12   int main(int argc, char** argv)
13   {
14         int x;
15         x = 30;
16         sub2(x);
17         x = 90;
18         sub2(x);
19         sub(3);
20         printf("%s %d\n",argv[0],argc);
21         return(0);
22   }
```

Walkthrough example

```
% gcc –g –o gdb_ex gdb_ex.c
% gdb gdb_ex
(gdb) set args a b c d    set program arguments
(gdb) list 1,22           list source file
(gdb) break 14            break at source line at program start
(gdb) break sub           subroutine break
(gdb) break 6
(gdb) run           start program (breaks at line 14)
(gdb) p argv              hex address of argv (char**)
(gdb) p argv[0]           prints “gdb_ex”
(gdb) p argv[1]           prints “a”
(gdb) p strlen(argv[1])   prints 1
(gdb) p argc              prints 5
(gdb) p /x argc                 prints 0x5
(gdb) p x           uninitialized variable
(gdb) n             go to next line
(gdb) p x           x now 30
(gdb) p /x &x             print address of x
(gdb) x/w &x              print contents at address of x
```

Walkthrough example

```
(gdb) n                   go to next line (execute entire call)
(gdb) s                   go to next source instr
(gdb) s                   go to next source instr (follow call)
(gdb) continue            go until next breakpoint (breaks at line 6 in sub)
(gdb) where               list stack trace
(gdb) p x                 x no longer scoped
(gdb) up                  change scope
(gdb) p x                 x in scope, prints 90
(gdb) del 3               delete breakpoint
(gdb) continue            finish

(gdb) info br             get breakpoints
(gdb) del 1               delete breakpoint
(gdb) break main          breakpoint main
(gdb) run                 start program
(gdb) watch x             set a data write watchpoint
(gdb) c                   watchpoint triggered
```

DDD