GDB in a nutshell

15-213 Spring 2001

Getting in and out of gdb

```
gdb          (starts gdb, files to be debugged can be later loaded)
gdb <file>   (starts gdb with <file> to debug)
gdb -h       (lists command options)

quit         (exits from gdb)
Ctrl-d       (same effect as quit)
Note: Ctrl-c does not exit from gdb, but halts the current gdb command
```

Running programs

```
run          (start your program)
run <args>   (start your program with the given arguments)
set args <args list>
             (specify the arguments to be used to run your program)
kill         (kill your program)
```

Breakpoints

```
break <function>  (set a breakpoint at entry to <function>)
break <line num>  (set a breakpoint at specified line number)
break *address    (set a breakpoint at specified address)

clear <function>  (delete the breakpoint set at entry to <function>)
clear <line num>  (delete the breakpoint set at that line number)

disable <num>     (disable the breakpoint with that number)
enable <num>      (enable the breakpoint with that number)

delete <num>      (delete the breakpoint with that number)
delete            (delete all breakpoints)
```

Working at breakpoints

```
step         (continue running program until it reaches different source line)
stepi        (execute one machine instruction)
stepi <num>  (execute <num> machine instructions)

next         (continue to the next source line in the current stack frame)
nexti        (execute one machine instruction, stepping over function)
nexti <num>  (execute <num> machine instructions, stepping over function)

continue     (resume execution)
continue <num>  (continue, ignoring this breakpoint <num> of times)

until        (continue running until a source line past the current line,
              in the current stack frame, is reached)
until <loc>  (continue running until the specified location is reached
              or the current stack frame returns)

finish       (run until the current function returns)
```

Examining the stack

```
frame <args>           (print out a stack frame)
select-frame <args>    (move from one stack frame to another)

backtrace              (print the current address and stack backtrace)
where                  (print the current address and stack backtrace)
```

Examining code

```
list                   (print out source file lines)

disas                  (display the function around the current line)
disas <addr>           (display the function around the address)
disas <addr1> <addr2>  (display the function between the addresses)
```

Examining data

```
print/f <exp>          (print out the value of <exp> with format /f)
x/nfu <addr>           (print the contents of <addr> in memory)
                        n: repeat count;
                        f: display format(i, instructions);
                        u: unit size (b, bytes, h, two bytes, w, 4 bytes)
display/f <exp>        (display the value of <exp> every time program stops)
display                (show the auto-displayed items)
delete display <num>   (stop displaying item <num>)
```

Examples:

```
print/a $pc            (print the program counter)
print $esp             (print the stack pointer)
print $eax             (print the contents of %eax)
print/x $eax           (print the contents of %eax as hex)
print/a $eax           (print the contents of %eax as an address)
print/d $eax           (print the contents of %eax as decimal)
print/t $eax           (print the contents of %eax as binary)
print/c $eax           (print the contents of %eax as a character)

print 0x100            (print the decimal representation of a hex value)
print/x 555            (print the hex. representation of a decimal value)

x/6xw 0x12345678       (print 6 words in hex from address 0x12345678)
x/10i <addr>           (print next 10 instructions)

display $eax           (print contents of %eax whenever program stops)
display/i $pc          (print the next machine instruction to be executed)
```

Information commands

```
help info
info program           (current status of the program)
info functions         (functions in program)
info stack             (backtrace of the stack)
info frame             (information about the current stack frame)
info scope             (variables local to the scope)
info variables         (global and static variables)
info registers         (registers and their contents)
info breakpoints       (status of user-settable breakpoints)
info address <symbol>  (use for looking up addresses of functions)
```