CS270 gdb summary

Department of Computer Science, University of Kentucky (Fall 2020)

0. For lab2, some C statements, open, lseek, read, memcmp, strcpy, byte order

1. To be able to use gdb, compile the program with -g option, e.g.,
   gcc -Wall -g -o match match.c

2. To run gdb on match (without any arguments for match, the program to be debugged)
   gdb match
   After getting into gdb, the typical steps are:
   1) set one or more breakpoints (*break*)
   2) run the program (*run*)
   3) repeat these in any order and for as many times as you need,
      run the next command (*n*), print values of variables (*p* or *x*), set/clear breakpoints (*b* or *clear*)
   4) quit gdb (*quit*)

3. Here is a list of basic commands after gdb is running:
   - *break* function-name or line-number, or address (set a break point)
   - *run* arguments-for-match (run the program with the args)
   - *list* [line-number] (list the programs)
   - *next* (run the next command)
   - *continue* (continue to run until next break point)
   - *print* variable-name (print the value of a variable)
   - *print /x* variable-name (print in hexadecimal format)
   - *p* pointer (register) (print the pointer address/register value)
   - *x* pointer (register) (print the content pointed to by the pointer/register)
   - *clear* line-number (delete the break point at the place)

Note:
1. Each command can be abbreviated as the first letter of the command:
   break (b), run (r), list (l), next (n), continue (c), print (p).
2. Undisplayable characters in character strings are printed out in *Octal* format.

4. Here is a list of more advanced commands for gdb:
   - *b* *address (set a break point at the address)
   - *stepi* (or *si*) (step one machine instruction, possibly into a procedure)
   - *nexti* (or *ni*) (go instruction without going through the procedure)
   - *finish* (finish all the instructions in the current function)

   - *p /x $rsi* (print out the hex value of the register)
   - *p /d $rdi* (print out the decimal value of the register)
   - *x/s $rsi* (print out the string in the memory pointed to by $rsi)
   - *x/6dw $rsp* (print out six numbers in memory pointed to by $rsp)
   - *x/3xb* 0x7fffffffe068 (print out three numbers in memory at an address, in hex and size is byte) (b1, h2, w4, g8)

Notice registers are indicated by $ in gdb, rather than % in the code.
You can use *objdump -d bomb > bomb.s* to get assembly code from machine code.