GDB

find bugs
gdb tutorial: a basic gdb session*

[moriah:~] 104) gdb test
  ...
  (gdb) _
  (gdb) break 7
  Breakpoint 1 at 0x8048228: file test.c, line 7.
  (gdb) run <arg 1> <arg 2> ...
  Starting program: /net/grad/erickee/test
  Breakpoint 1, main () at test.c:7
  7    testfcn();
  (gdb) s
  testfcn () at test.c:14
  14   printf("Hello World\n");
  (gdb) n
  15   }
  (gdb) continue
  Continuing.
  Program exited normally.
  (gdb) q

  * Parameter ‘test’ is the name of the program to debug
  * gdb outputs some uninteresting gdb metadata
  * gdb waits at a blank prompt. What now?
  * Set a breakpoint at line 4 of the code using break
  * gdb reports the address of the break
  * test isn’t running; run starts execution w/optimal arguments
  * gdb reports that things are going to happen!
  * execution halts at the breakpoint on line 7 of test.c
  * gdb prints the C code at line 7 (not yet executed)
  * Use the s command to step into the function testfcn()
  * execution halts at next line of code encountered by cpu
  * next line is at line 14 in test.c (this happens to be our code)
  * Use n command to step over the printf(...) function
  * execution halts at next line of code
  * Use continue to run the program until next breakpoint
  * Because there are no other breakpoints, the program ends
  * Use q to quit gdb

* This information and more can be found on the course website by clicking on “Textbook and Resources” and then “gdb”
gdb tutorial: command reference

• We just used the following commands
  – break : sets a breakpoint
  – run : runs from beginning to first breakpoint
  – start : runs to the start of main()
  – s : executes the next line, even if inside a new function call
  – n : execs next line but skips over function calls
  – continue : resumes execution until next breakpoint is reached
  – quit : exits gdb

• What other commands does gdb offer? (many…)
  – finish : finishes executing code in current function (aka “step out”)
  – delete n: deletes breakpoint number n
  – print X: prints the value of the variable X
  – l : (lower case l) Lists 10 lines of code around the current line
  – print X=3 : change the value of X to 3 (print will execute any command including function calls)
Valgrind

find tougher bugs
What does Valgrind do?

• Automatically detects bugs
  – Memory management bugs
  – Threading bugs (*helgrind*)
    • Not working under current version of Valgrind

• Memory management bugs
  – Compile your code with the -g option
  – Run:
    • `valgrind --leak-check=yes myprog <myarg1> ...`

http://valgrind.org/
What can Memcheck Find?

• Detects memory management problems
  – Checks all reads and writes to memory
  – Intercepts all calls to malloc and free

• For example:
  – Using uninitialized memory
  – Reading/writing free’d memory
  – Reading/writing off end of malloc’d blocks
  – Leaks: lost pointers to malloc’d blocks
  – A couple of other things, see:
    • http://valgrind.org/docs/manual/manual-intro.html#manual-intro.overview
Valgrind output

- ==23321== Invalid write of size 4
- ==23321== at 0x804840F: f (leakoverflow.c:71)
- ==23321== by 0x804842C: main (leakoverflow.c:77)
- ==23321== Address 0x41A3050 is 0 bytes after a block of size 40 alloc'd
- ==23321== at 0x4022525: malloc (vg_replace_malloc.c:149)
- ==23321== by 0x8048405: f (leakoverflow.c:69)
- ==23321== by 0x804842C: main (leakoverflow.c:77)

Everything is working perfectly!

- ==23321==
- ==23321== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 13 from 1)
- ==23321== malloc/free: in use at exit: 40 bytes in 1 blocks.
- ==23321== malloc/free: 1 allocs, 0 frees, 40 bytes allocated.
- ==23321== For counts of detected errors, rerun with: -v
- ==23321== searching for pointers to 1 not-freed blocks.
- ==23321== checked 47,932 bytes.
- ==23321==
- ==23321== LEAK SUMMARY:
- ==23321==   definitely lost: 40 bytes in 1 blocks.
- ==23321==   possibly lost: 0 bytes in 0 blocks.
- ==23321== still reachable: 0 bytes in 0 blocks.
- ==23321== suppressed: 0 bytes in 0 blocks.
- ==23321== Rerun with --leak-check=full to see details of leaked memory.