Debugging in C

ITSC 2181: Introduction to Computer Systems
UNC Charlotte
College of Computing and Informatics

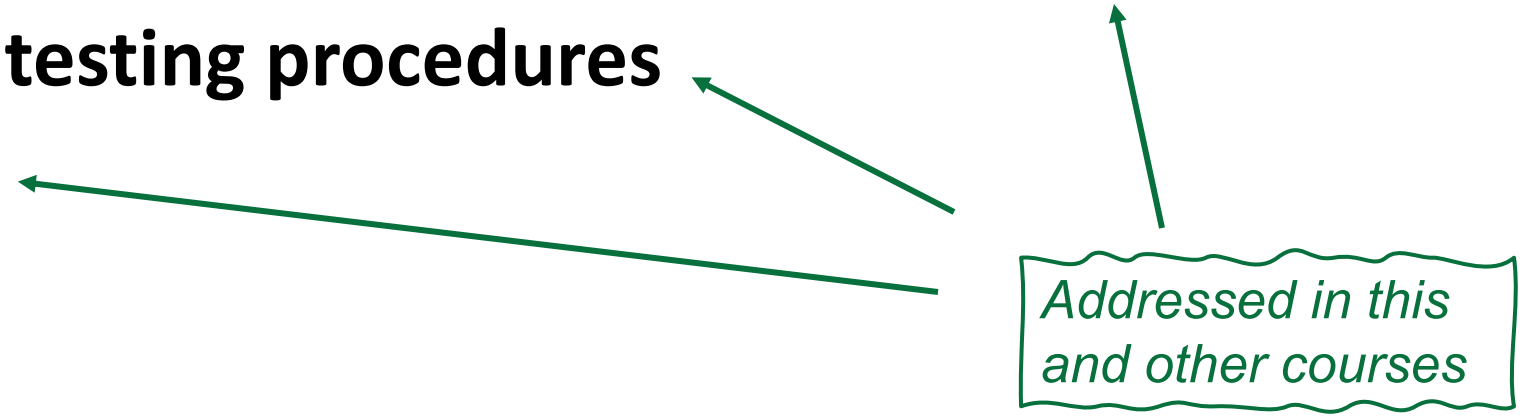
Debugging?

- A **very significant part** of software development is testing, debugging, and bug fixing.

- A bug is a defect in programming code.

Why Do Bugs Happen ?

- OS problem? Compiler? Hardware? – not likely
- Unclear requirements / specifications, constantly changing, unreasonable schedules, …
- **Lack of mastery of programming tools / language**
- **Inadequate testing procedures**
- **Faulty logic**



Source Level **Debugging**

- Symbolic debugging lets you single step through program, and modify/examine variables while program executes
- On the Linux platform: **`gdb`**
- On the MacOS: **`lldb`**
- Source-level debuggers built into most IDEs

gdb commands

`list <line>` `list <function>`	list (show) 10 lines of code at specified location in program
`list <fst_line>,<lst_line>`	List from first line to last line
`run`	start running the program
`continue`	continue execution
`step`	single step execution, including into functions that are called
`next`	single step over function calls
`print <var>` `printf "fmt", <var>`	show variable value

gdb commands (cont'd)

`display <var>` `undisplay <var>`	show variable each time execution stops

`break <line>`
`break <function>`
`break <line> if <cond>`
	set breakpoints (including conditional breakpoints)

`info breakpoints` `delete breakpoint <n>`	list, and delete, breakpoints
`set <var> <expr>`	set variable to a value
`where` `backtrace full`	show the call stack, and arguments and local variables

Finding Bugs

1. Test **as you write the code** (write *test harness*)
   - Make sure you remove it before delivery

2. Write trivial programs to test your mastery of the programming language, library functions, etc.

3. Working backwards from an error: **divide and conquer**

Finding Bugs (cont'd)

4. Make the bug reproducible (eliminate all variations in execution conditions)
5. Try simple things first (*sanity checking*)
   - including, check the inputs
6. Inspect your code and think about it!
7. Ask for help, explain code / bug to TA/IA or instructor
8. Write an automated test program or script

References

- S. J. Matthews, T. Newhall and K. C. Webb, *Dive into Systems*, Version 1.2. Free online textbook, available at: https://diveintosystems.org/book/

- K. N. King, *C Programming: A Modern Approach*, 2nd Edition. W. W. Norton & Company. 2008.

- D.S. Malik, *C++ Programming: From Problem Analysis to Program Design*, Seventh Edition. Cengage Learning. 2014.