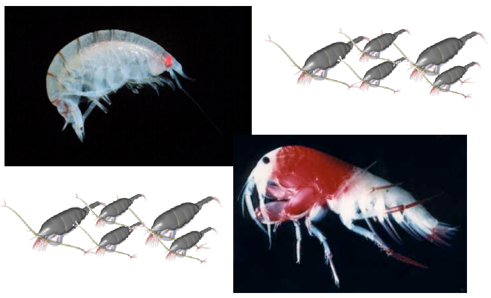
Debugging



Outline

- Announcements:
  - HW II dueFridayl
  - Q1 on model problem is answered in key to PS1
    - You may use the file-type I specify
    - Or design your own
    - Just make sure Q1 and Q2 are consistent
- Old fashioned debugging
- Debugging tools

Old-Fashioned Debugging

- The point of debugging is to find your errors
- Simplest technique is **checkpointing**
  - Place an output statements around calls to subroutines
    - Printf("Entering subroutine A")
    - A();
    - Printf("Completed subroutine B")
  - If your program crashes in A, you won't see the second line
- Work into subroutines, bracketing sections of code with outputs until you find where the error occurs.

Old-Fashioned Debugging

- Checkpointing is nice because it works on any system that can run your code
- But, requires lots of compiles as you zero in on bug.
- Can also output data values
- WARNING: Finding the line where the program crashes is not enough, you need to know why!
  - The problem could result from a previous statement
  - In this case, figure out where the variables on the offending line are set, and work backwards

Middle-age debugging

- UNIX standard debugging program is db (gdb on Linux)
- gdb allows you to watch your program run
  - Set breakpoint--position in code, execution will stop when reached
  - Step through program line-by-line
  - Examine value of variables

db

- To use db, compile with -g flag
  - On most systems, can't use -g with -O (optimizations)
- Then type
  - (g)db program inputs
- Db will start and it will "grab" your program

db

- Typical session,
  - Set breakpoint(s) (br)
  - Run until you hit a breakpoint (run)
  - Step through some lines (n, s)
  - Look at some variables (p)
  - Continue to next breakpoint (c)
  - Quit (q)

db

- Setting breakpoints
  - break ReadComm-- sets a breakpoint at ReadComm
  - break io.c:21 --sets a breakpoint at line 21 in io.c

db

- Stepping through
  - Typing run will take you to the next breakpoint
  - You can step through line by line by typing n(ext)
    - gdb will display the next line of code to be executed
  - If the next line is a call to another subroutine
    - Typing s(tep) will "step into" that rountine
    - Typing n(ext) will skip to the next routine

db

- Viewing variables
  - You can check the value of a variable by typing “p name”
    - This may not be what you expect
      - p array will give the memory address of the array
      - p array[0] will give the value at the first location
- db is complicated
  - Type man db or man gdb to see more info

Modern Debugging

- IDE’s like VizStudio have graphical debuggers
  - On some systems, this is just a GUI for db