**Debugging**

Symptom of errors: Fatal errors (seg fault), infinite loops, unexpected results

Causation of errors: Misunderstanding of algorithm, language constructs, carelessness (pointers)

Compilers don’t catch all errors

Critical part of debugging: To narrow focus to a small region of a large code / state

Systematic testing is critical in assisting debugging

* Test for boundary conditions (e.g. empty list)
* Use a collection of tests to reveal:
  + Trigger points (data becomes incorrect after executing certain fn)
  + Patterns of behaviour (final result is always one more than expected)

Develop a hypothesis to guide debugging

**Debugging via. assert()**

Pros: Allows you to specify pre-conditions in code + indicates the problem and where it occurred

Cons: Not useful for error-handling, doesn’t give indication of state (e.g. assert(n > 0) fails… but at which value of n?)

**Debugging via. printf()**

Pro: Flexible in what can be displayed

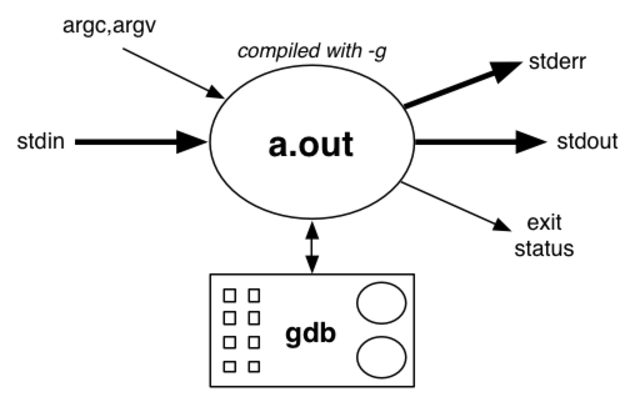
Cons: Where to place printf()? What to print? Doesn’t give control over program execution. Need to remove debugging output before deploying

* **#ifdef DEBUG** and **gcc –DDEBUG** helps with managing this.

**GDB: The GNU Debugger**

GDB provides facilities to:

* Control execution of a program (step-by-step execution, breakpoints)
  + Breakpoints are temporary “time-outs” in a program for debugging purposes
* View state of program (values stored in variables)



Programs must be compiled with **–g** option  
(in CSE, use **–gdwarf-2**)

GDB commands:

* **quit** – quits GDB program
* **help** **[CMD]** – gives info about command
* **run ARGS** – run the program
* **where** – stack trace
  + Find which function was executing when it crashed
  + Stack may have references to system error-handling functions
* **up [N]** – move down the stack
  + Allows you to skip to scope of particular function in stack
* **list [LINE]** – show code
  + Displays five lines either side of current statement
* **print EXPR** – display expression values
  + EXPR may use current values of variables
  + Special expression **a@1** shows all of the array **a**
  + Almost any C expression can be used:  
    E.g. (gdb) print (float)n/3 // prints 1/3th of value of n as a float  
    E.g. (gdb) print a[i] // prints I’th value in array a[i]
* **break [FUNC | LINE]** – set break-point
  + Stop execution and return control to GDB on entry to function FUNC or on reaching LINE
* **next** – single step (over functions)
  + Execute next statement. If statement is a function call, execute whole function.
* **step** – single step (into functions)
  + Execute next statement. If statement is a function call, go to 1st statement in function body