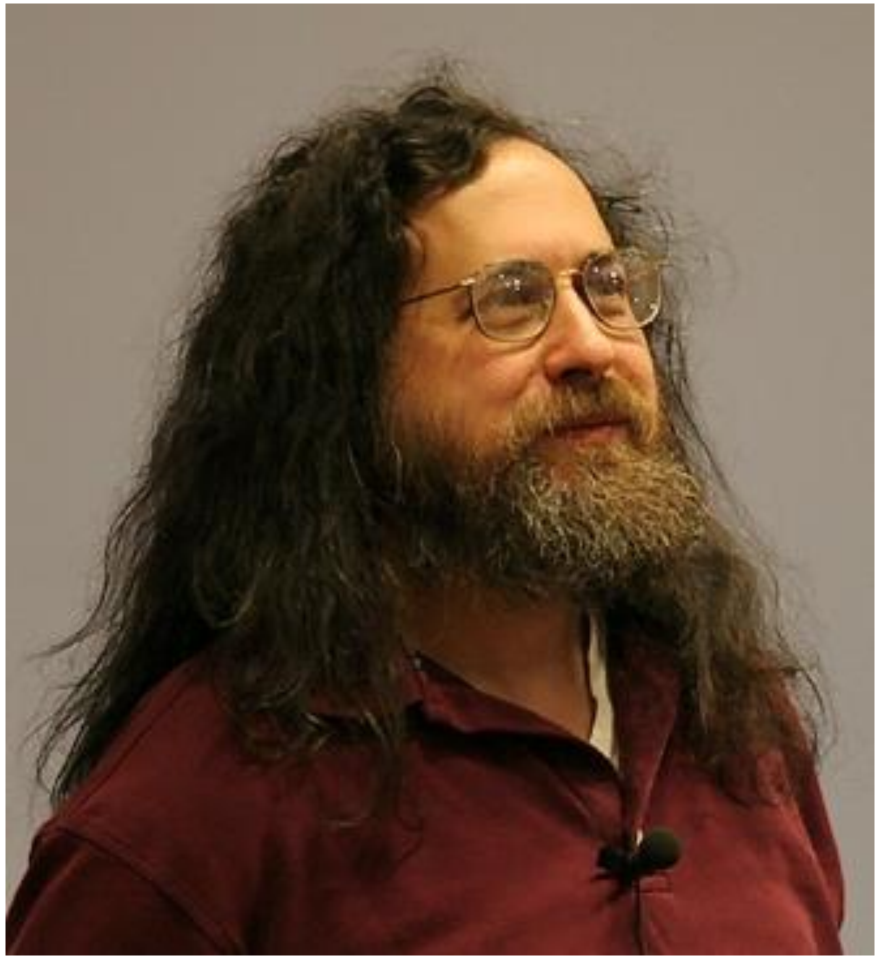
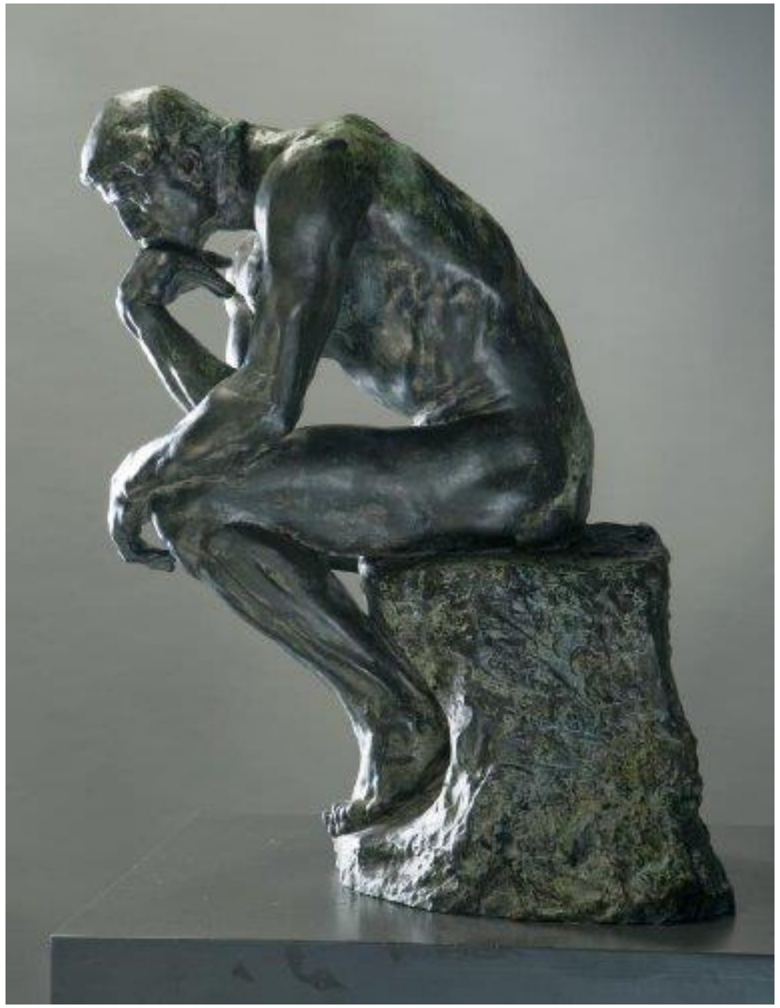
The CS-220 Development Environment

Who is this guy?

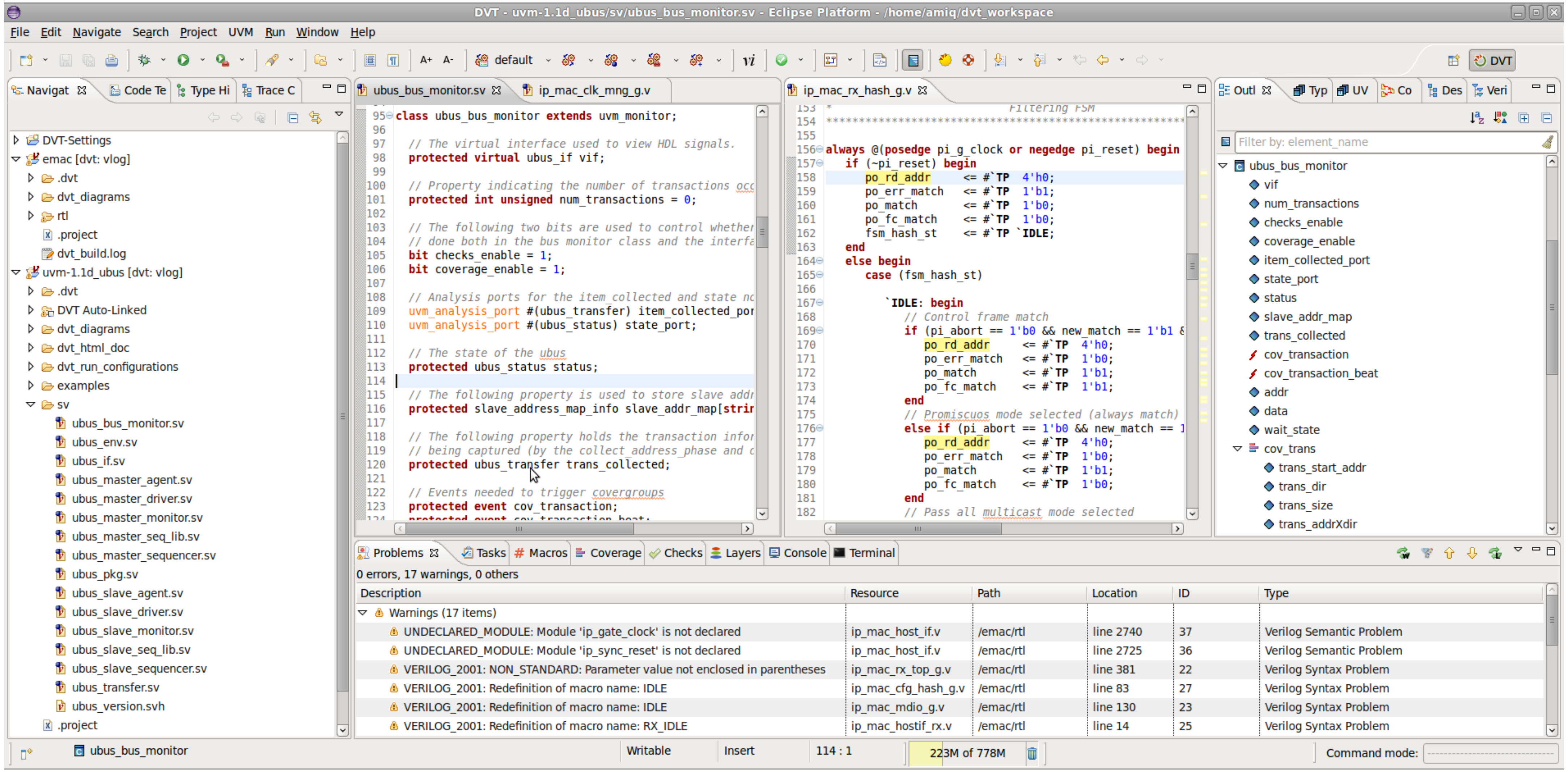


- Arguably more influential in software development than either Bill Gates or Steve Jobs
- Developed the concept of “copyleft”
- Caused DARPA to require passwords at MIT
- Created the first recursive acronym: GNU

Picking the Right Tool for the Job



Integrated Development Environment



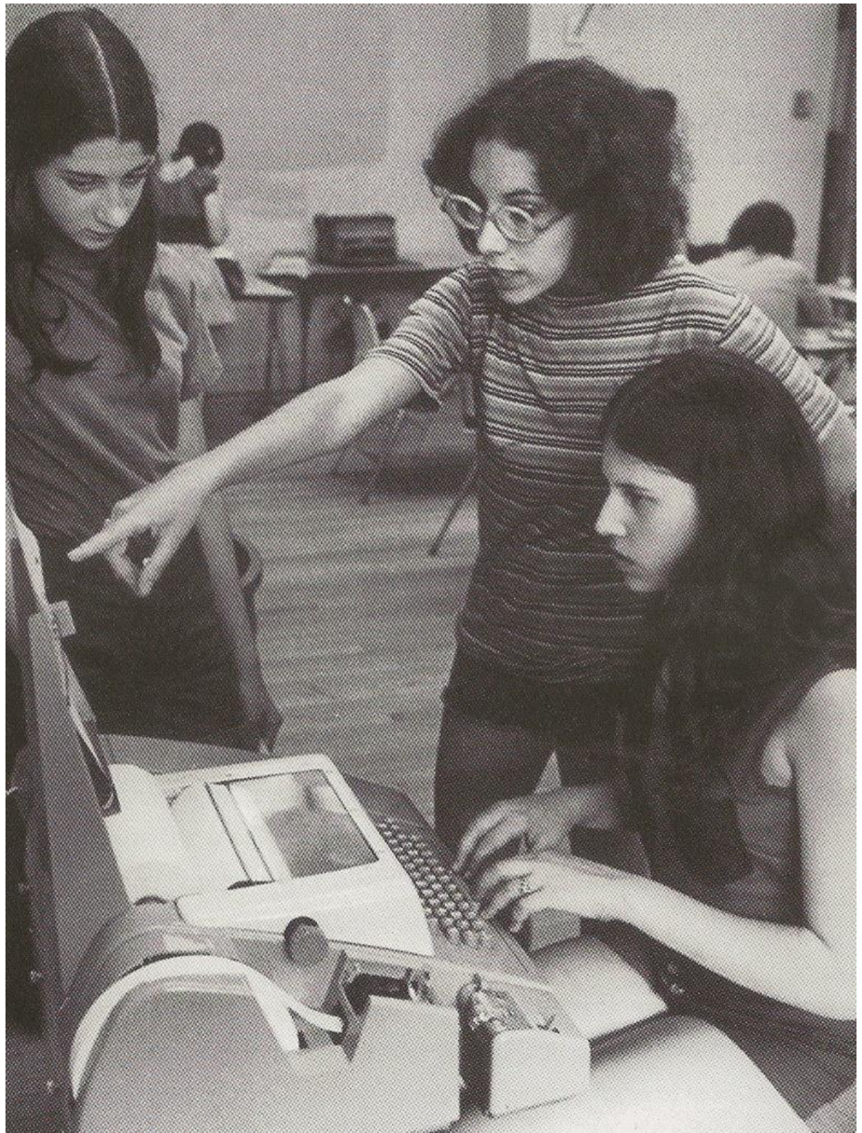
```
class ubus_bus_monitor extends uvm_monitor;

  // The virtual interface used to view HDL signals.
  protected virtual ubus_if vif;

  // Property indicating the number of transactions occ
  protected int unsigned num_transactions = 0;

  // The following two bits are used to control whether
  // done both in the bus monitor class and the interfa
  bit checks_enable = 1;
  bit coverage_enable = 1;

  // Analysis ports for the item_collected and state nc
  uvm_analysis_port #(ubus_transfer) item_collected_por
  uvm_analysis_port #(ubus_status) state_port;

  // The state of the ubus
  protected ubus_status status;


  // The following property is used to store slave addr
  protected slave_address_map_info slave_addr_map[strir

  // The following property holds the transaction infor
  // being captured (by the collect_address_phase and c
  protected ubus_transfer trans_collected;

  // Events needed to trigger covergroups
  protected event cov_transaction;
```

```
*                                   Filtering FSM
*********************************************************

always @(posedge pi_g_clock or negedge pi_reset) begin
   if (~pi_reset) begin
      po_rd_addr      <= #`TP  4'h0;
      po_err_match    <= #`TP  1'b1;
      po_match        <= #`TP  1'b0;
      po_fc_match     <= #`TP  1'b0;
      fsm_hash_st     <= #`TP `IDLE;
   end
   else begin
      case (fsm_hash_st)

         `IDLE: begin
            // Control frame match
            if (pi_abort == 1'b0 && new_match == 1'b1 &
               po_rd_addr      <= #`TP  4'h0;
               po_err_match    <= #`TP  1'b0;
               po_match        <= #`TP  1'b1;
               po_fc_match     <= #`TP  1'b1;
            end
            // Promiscuos mode selected (always match)
            else if (pi_abort == 1'b0 && new_match == 1
               po_rd_addr      <= #`TP  4'h0;
               po_err_match    <= #`TP  1'b0;
               po_match        <= #`TP  1'b1;
               po_fc_match     <= #`TP  1'b0;
            end
            // Pass all multicast mode selected
```

0 errors, 17 warnings, 0 others

Description	Resource	Path	Location	ID	Type
Warnings (17 items)					
UNDECLARED_MODULE: Module 'ip_gate_clock' is not declared	ip_mac_host_if.v	/emac/rtl	line 2740	37	Verilog Semantic Problem
UNDECLARED_MODULE: Module 'ip_sync_reset' is not declared	ip_mac_host_if.v	/emac/rtl	line 2725	36	Verilog Semantic Problem
VERILOG_2001: NON_STANDARD: Parameter value not enclosed in parentheses	ip_mac_rx_top_g.v	/emac/rtl	line 381	22	Verilog Syntax Problem
VERILOG_2001: Redefinition of macro name: IDLE	ip_mac_cfg_hash_g.v	/emac/rtl	line 83	27	Verilog Syntax Problem
VERILOG_2001: Redefinition of macro name: IDLE	ip_mac_mdio_g.v	/emac/rtl	line 130	23	Verilog Syntax Problem
VERILOG_2001: Redefinition of macro name: RX_IDLE	ip_mac_hostif_rx.v	/emac/rtl	line 14	25	Verilog Syntax Problem

Command Line Mentality



Old fashioned but simple…

…and surprisingly efficient…

… (except for editing).

Command Line Mentality

- Artisan vs. Factory Worker
- Learn to use lots of tools
- May be slower
- Take more knowledge
- Require more effort
- Tools you can take with you
- Tools universally available
- Applicable to wide variety of projects

Operating System: UNIX

- Oldest “modern” operating system
- Base for Linux, HP/UX, Solaris, AIX, Android, IOS, etc.
- Most widely used
- Available on Lab accounts
- Free version available on Windows : CYGWIN (https://www.cygwin.com/)
- Large library of free software - GNU compilers, editors, debuggers, etc. (http://en.wikipedia.org/wiki/GNU)
- Assume basic knowledge of UNIX (http://www.ee.surrey.ac.uk/Teaching/Unix)

Expect you to know (or figure out)…

- Unix text file
- Unix Directory (parent, child directories)
- Current Directory
- Home Directory
- Commands: ls, cd, mv, cp, mkdir, rmdir,
- How to open and use a command line terminal

Basic Commands

- Editor – gedit (not available everywhere)
- Compiler / Linker – gcc
- Build manager – make
- Debugger – gdb
- Project Management – git

Environment 1: G-7 Lab

- Everyone should have gotten an e-mail about LDAP access
  - LDAP = “Lightweight Directory Access Protocol”
- Homework this week… log on and try it out (run the demo)
- You get a “home” directory – Your personal disk space
- All CS machines (G-7, remote, etc.) share:
  - userid/password
  - Your home directory
- https://www.cs.binghamton.edu/~sysadmin/ to change password, etc.

Environment 2: Remote CS

- I use “PuTTy” for remote access
- Host Name: remote.cs.binghamton.edu
  - The CS machines are different from Harvey or BingSuns!
- By default, putty does not support full-screen applications
  - Can use “XLaunch” to enable remote full-screen applications (gedit)
  - Performance is not always wonderful
- Alternative: Edit locally and transfer files back and forth
  - I use “WinSCP” to transfer files / edit remote files locally

Environment 3: Unix on your Machine

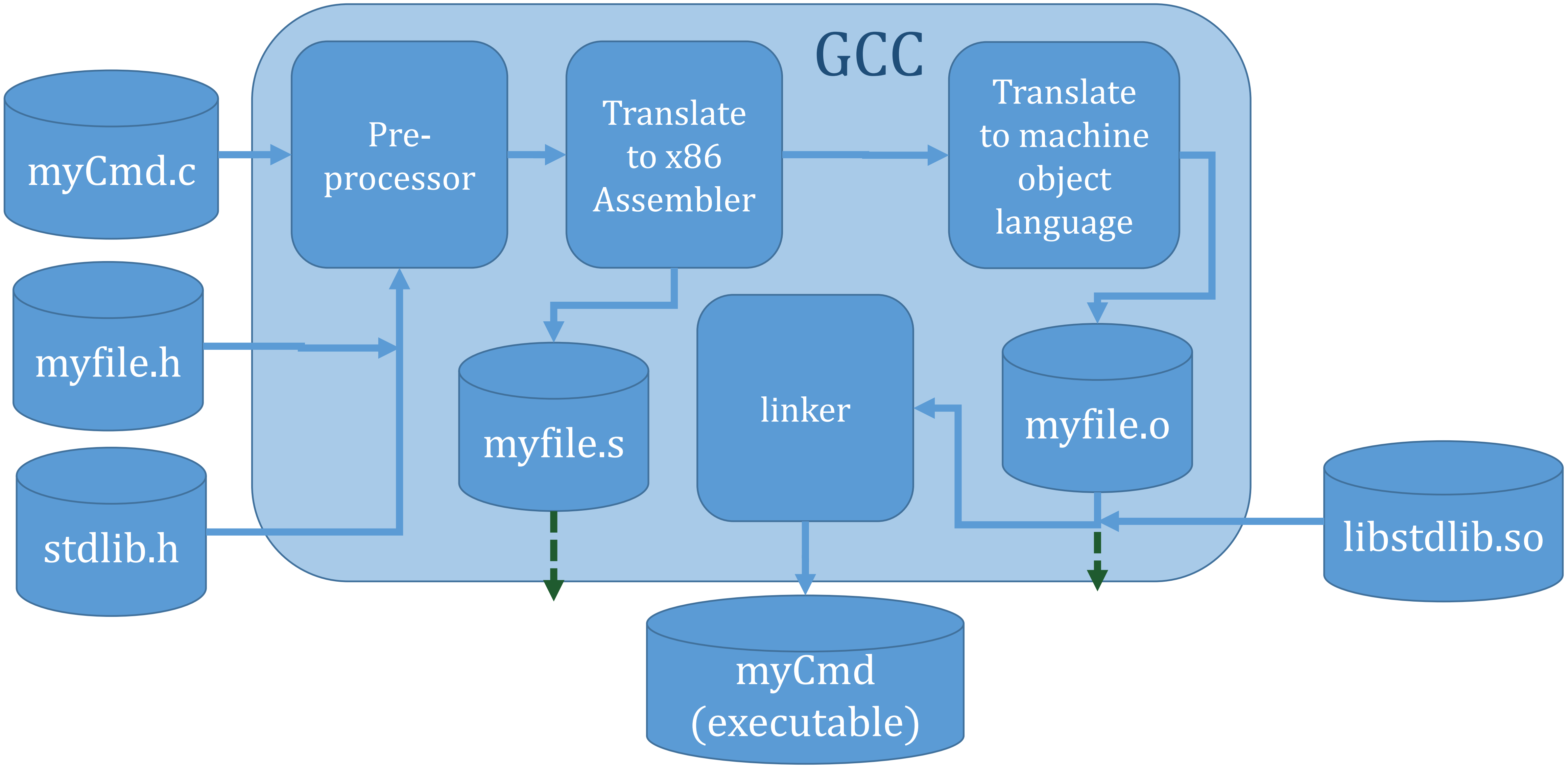
- C is notoriously non-portable!
- The underpinnings of C (e.g. X86) even less so
- You may use your own machine
  - Dual boot Linux, Apple development environment, Cygwin on Windows, Bash on Windows 10
  - You will need “gcc”, “gdb”, “gmake”, and “git” for this class
- … but test on CS machines! (You will be graded on CS machines)

GCC (Gnu C Compiler)

>gcc *options source_file(s)*

- Basic options for CS-220:
  - -o *output_file/command_name*
  - -g [include debug information in output file]
  - -Wall [turn on all warning messages]
- For example: >gcc –g –Wall –o myCmd myCmd.c
- For more detail, gcc -- help or man gcc
- Complete documentation: https://gcc.gnu.org/onlinedocs/

The compile / link process



Makefile

- File that tells "make" what to do and how to do it
- Composed of a list of "make rules"
- A rule has three parts:
  - Target – the file that this rule produces
  - Dependencies – A list of file used to make the target
  - Recipe – Unix command(s) to produce target from the dependency files
- For example:

```
mymain : mymain.c mymain.h comp.c comp.h
	gcc -g -Wall -o mymain mymain.c comp.c
```

Required Tab

Example Makefile

```
test : mycmd
	./mycmd “test string”
	./mycmd “test string 2”

mycmd : mymain.c mymain.h comp.c comp.h
	gcc –g –Wall –o mycmd mymain.c comp.c

clean:
	-rm mycmd
```

Invoking make

- Command : make *target*
- If *target* not specified, look for target “all”
- If *target* not specified and no “all” target,
  make first target in the make file

Make processing (simplified)

- Find the rule for the target specified
- Recursively make all dependencies that are targets
- If any dependencies are newer than target file, invoke recipe
  - If the target is a file, and the time/date stamp of the target is newer than the time date stamp all dependencies, skip the recipe
  - If the target is a file, and the time/date stamp of the dependency is newer than the time/date stamp of the target, invoke the recipe
  - If target is not a file (pseudo-target), assume it is “ancient” always run the recipe
- Cascading dependencies causes build of anything out of date

Debugging – GDB (Gnu DeBugger)

- gdb *executable*
  - Starts debugger, loads executable, and prompts for gdb commands
- Basic GDB commands:
  - h[elp] – help on gdb commands
  - b[reak] *location* [if *cond*] – set a breakpoint at *location*
    - *location* can be either a line number in a file or a function name
    - *cond* stop only when true
  - run *command line arguments* - invoke command and run to next breakpoint
  - c[ontinue] – continue to next breakpoint
  - s[tep] – run next program instruction, stepping into function invocations
  - n[ext] – run next program instruction, skipping over function invocations
  - p[rint] *variable or expression* - print the current value of variable or expression
  - x /*options location* - print (examine) memory at *location*
  - q[uit] – exit out of gdb

GIT – Version Control

The name "git" was given by Linus Torvalds when he wrote the very first version. He described the tool as "the stupid content tracker“ and the name as (depending on your way):

- random three-letter combination that is pronounceable, and not actually used by any common UNIX command. The fact that it is a mispronunciation of "get" may or may not be relevant.
- stupid. contemptible and despicable. simple. Take your pick from the dictionary of slang.
- "global information tracker": you're in a good mood, and it actually works for you. Angels sing, and a light suddenly fills the room.
- "goddamn idiotic truckload of sh*t": when it breaks

GitHub Service

- Software owned by Microsoft
- Uses “GIT” commands as an interface
- Requires userid/password
- Enables “cloud” storage of “repositories”
- A repository is like a sub-directory (project or package)
  - Has name, owner, permissions, versions, branches
  - May have multiple (divergent) copies!
  - Primary copy is in the cloud

GitHub Classroom

- Software package built on GitHub
- Allows me to create an "assignment"
  - Consists of a repository with a README.md and potentially starter files
  - Has an "invitation" URL
- If you "browse" the invitation, GitHub Classroom will allow you to "Accept" your invitation
  - Create a new private repository for you *assignment-gitId*
  - You may modify your repository to "do" the assignment
  - TA's and I can see your (modified) repository to grade it

GitHub Single Threaded Development

- git clone *repository-url*
  - Makes a copy of repository to local disk space
  - Repository name becomes a sub-directory in the local directory
  - You may edit files in the cloned copy of the directory (does not change cloud!)
- git add *file-name*
  - Tells git that this file has changed (been modified or added) in this copy of the repository
- git commit –m ‘*commit comment*’
  - Tells git that you want to save the “staged” changes you have made in this copy
  - May use –a flag to “auto-add” any tracked file that has been changed
- git push
  - Copy all committed changes to the cloud
- git pull
  - Copy any changes from the cloud to the current directory (clone)

Managing Assignment Deadlines

- Each pushed commit has an associated hash code
  - list of hexadecimal characters that "contains" code derived from contents of repository (if contents changes, hash code changes also)
- Get hash code of latest commit in the cloud with
  "git rev-parse HEAD"
- Cut and paste hash code in myCourses submission area
  - myCourses records time/date of hash code submission
- CA's will retrieve the version of your repository associated with that hash code to grade

Demo

- See examples/xmp_gdb/
- A jar of marbles has a random mix of
  100 red marbles,
  100 green marbles,
  100 blue marbles
- How many marbles do you need to pick out of the jar to get 10 red marbles?