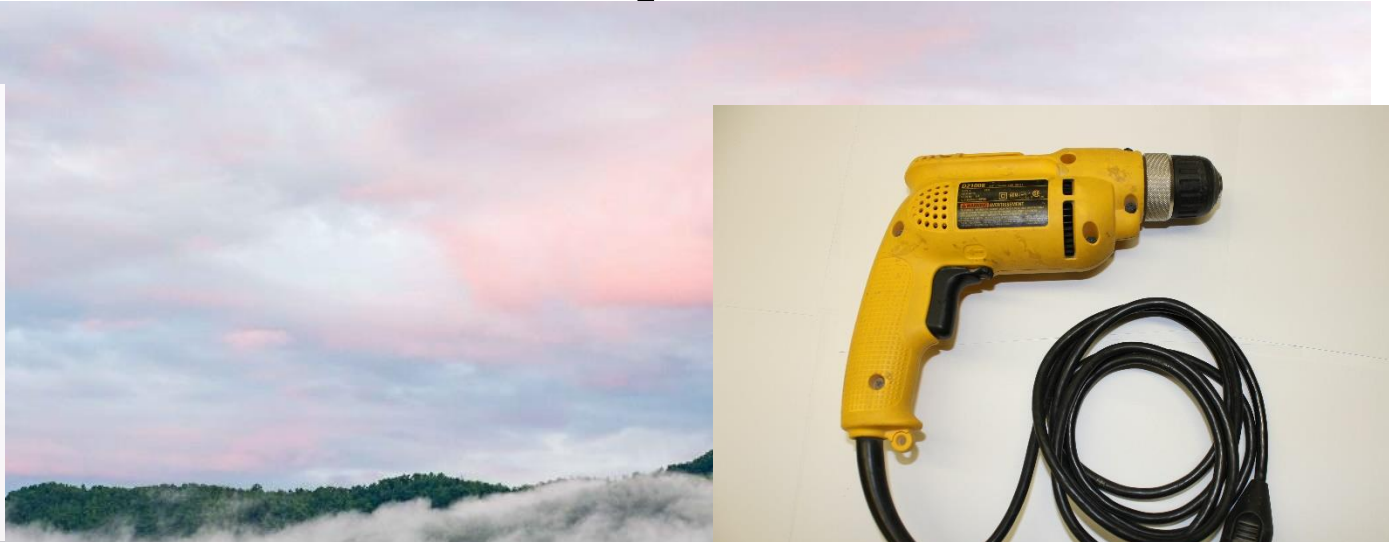


# CS-211 Development Environment

IDE vs. Command Line

Editor, Compiler, Builder, Debugger

# Picking the right tool for the job





# Integrated Development Environment

The screenshot displays the Eclipse IDE interface for the DVT (Design Verification Tool) environment. The main editor shows the Verilog code for the 'ubus\_bus\_monitor' module, which extends the 'uvm\_monitor' class. The code includes comments and Verilog syntax for property monitoring and event triggering. The left sidebar shows a project tree with files like 'ubus\_bus\_monitor.v', 'ubus\_if.v', and 'ubus\_master\_driver.v'. The right sidebar shows a 'Filter by: element\_name' search bar and a list of elements including 'vif', 'num\_transactions', 'checks\_enable', 'coverage\_enable', 'item\_collected\_port', 'state\_port', 'status', 'slave\_addr\_map', 'trans\_collected', 'cov\_transaction', 'cov\_transaction\_beat', 'addr', 'data', 'wait\_state', 'cov\_trans', 'trans\_start\_addr', 'trans\_dir', 'trans\_size', and 'trans\_addrXdir'. The bottom panel shows a 'Problems' window with 17 warnings, including 'UNDECLARED\_MODULE: Module 'ip\_gate\_clock' is not declared', 'UNDECLARED\_MODULE: Module 'ip\_sync\_reset' is not declared', 'VERILOG\_2001: NON\_STANDARD: Parameter value not enclosed in parentheses', 'VERILOG\_2001: Redefinition of macro name: IDLE', 'VERILOG\_2001: Redefinition of macro name: IDLE', and 'VERILOG\_2001: Redefinition of macro name: RX\_IDLE'.

# IDE

- Single interactive graphical user interface (GUI)
- Includes:
  - File/Project Management
  - Editor
  - Compiler
  - Builder
  - Debugger
  - Execution Environment

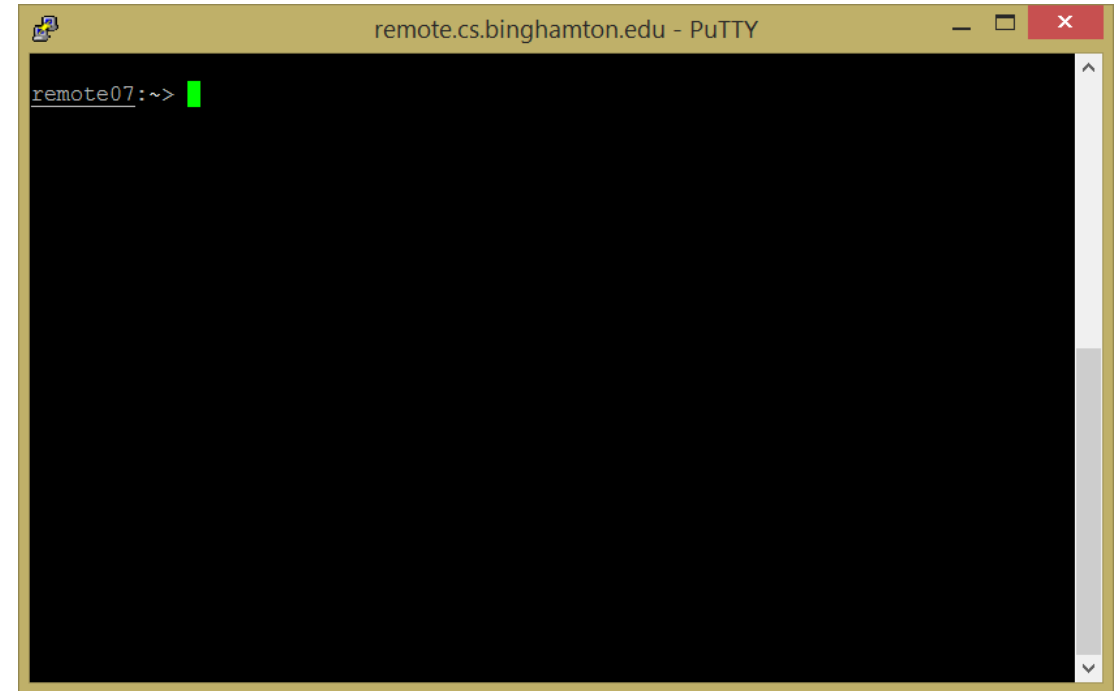
# Command Line Mentality



- 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

# UNIX shell

- “Shell” is a command line window
- “Prompt”
  - Machine name
  - Current Directory
  - “>”
- Type commands at prompt
  - Nothing happens until you hit “<enter>”
- When you hit enter...
  - Command is executed
  - Response appears
  - Another prompt appears



# Project / File Management

## UNIX File & Directory Commands

- `mkdir` to make a new project directory
- `cd` to that project
- run all other tools inside that directory
- “`tar`” to consolidate directory into a single transmittable file

# Edit Source Code

gedit

- Remember, displayed version not written to file until “save”
- If you have made edits and not saved, an “\*” will precede file name
- Start with the command “gedit <file>&”
- Recommend Edit/Preferences/
  - View - ☒ Display line numbers
  - View - ☒ Highlight matching brackets
  - Editor - ☐ Create a backup copy of files before saving



# Build Executable

make

- To start, we don't need “make”
- As projects get more complicated, we will learn “make”

# Compile Code

gcc

- For now, “gcc -o <executable\_file> -g <source\_file>”
- More as time goes on

# Debug Code

`gdb`

- Command “`gdb <executable file>`”
- Opens gdb prompt: “(gdb)”
- Type gdb commands at prompt
  - “`break <linenumber>`” - stop and prompt before executing line
  - “`run`” – start program and run to next break or end of program
  - “`step`” – execute next line of source code (to next semi-colon)
  - “`quit`” – to exit debugger

# Demo...

# Resources

- Programming in C, Chapters 1 & 2
- Wikipedia: Integrated Development Environment ([https://en.wikipedia.org/wiki/Integrated\\_development\\_environment](https://en.wikipedia.org/wiki/Integrated_development_environment))
- Wikipedia: List of Unix Commands ([https://en.wikipedia.org/wiki/List\\_of\\_Unix\\_commands](https://en.wikipedia.org/wiki/List_of_Unix_commands))
- gedit wiki (<https://wiki.gnome.org/Apps/Gedit>)
- gcc online documentation (<https://gcc.gnu.org/onlinedocs/>)
- gdb online documentation (<http://www.gnu.org/software/gdb/documentation/>)