265 project overview

- Semester long project, involving 6 labs (roughly biweekly) and a final submission, worth 40% of grade overall
- Will be distributed and collected via our git processes
- In first lab you'll be obtaining a git repo of code from the instructor, and with each subsequent lab you'll be pulling changes to the repo and merging them with your work so far
- Each lab will add new requirements for you to complete
- For final project submission (Dec 10) you're expected to have fixed any issues noted in your earlier lab submissions

Requirements

- Formal details are on the course project page, the project standards page, and the six lab pages (lab pages will be released roughly biweekly, along with repo updates)
- Links for each are provided off the main page for 265
- Various labs will involve working with git, bash, man pages, C++/g++, makefiles, gdb, gprof, gtk, unicode, etc
- The code will only compile/run on the csci servers: all project work must be done on the servers, and must follow the git practices described in the code standards

Quad trees

- Similar to the concept of binary trees, but each node has four children instead of two
- Used for a wide variety of purposes, from collision detection to image compression
- We’re using a (rather unusual) variant of quadtrees, so don’t expect to plug in quadtree code found on the web
- Each level of the quadtree subdivides portions of a 2D map into four quadrants, labelled 0-3
- Address strings specify locations in the map, e.g. “013” means quadrant 0 at top level, then quadrant 1 within that, then quadrant 3 within that

C++ demo program

- The project will involve work on a C++ demo program for a collection of classes (Address, Item, ItemTree), plus a lot of supporting content (makefile, scripts, test data, man pages, etc)
- Address class works with the address strings, e.g. “013”
- Item class represents data items we’ll be storing in the quadtree
- ItemTree class represents the quadtree itself
- Demo program uses all three classes
- Eventually will have header (.h) and implementation (.cpp) file for each, plus .o files and the demo executable