Research Documentation

Interfaces and Documentation
Sometimes boring, always essential
Arrange set of tasks so that individual efforts can proceed with frequent re-sets involving the whole group
Well defined interfaces between different hardware blocks or layers of software very useful
—greater independence of effort
—simplifies debugging
—interfaces define the external face of system, which may then be developed and debugged without reference to the other components: restricts the number of variables that may between subsystems
—interfaces suggest a set of test that need to be performed to verify system; most project development time is consumed by testing and debugging—anything that can reduce this is welcome

Software archiving standards:
-systems that enable version control with storage of past software versions
Main purpose: enable to roll back to a previous version that worked to help in debugging software that's been changed to enable new functions, run faster, apply to new OS, etc.
Secondary purpose: to be able to trace changes (along with who's responsible) for such matters as deciding who should be credited with new inventions.
Branching systems enable both independent coding and group coordination
These are usually paired with software documentation standards, explicitly to call out changes that were made and why
GitHub is the most common repository for student work; lots of public resources, enables sharing with varying levels of access. There are some private repositories also.

Exercise: what's the purpose of documenting software or experiments?

Personal reason: for large tasks, never remember all the essential details for every piece. Reminders as to what code is doing and why, to do comments, etc. all assist with modifying code, dealing with hardware quirks, version issues, calibration etc.
—hard to remember what one was thinking at time of composition. For every consequential piece of code helpful to write about purpose, what you were thinking, so that if similar problem arises in future can adapt the code
—always comment the input/output characteristics of hardware/software. This lets you and others use the functions even if there is no understanding of what the underlying details are. This includes how it is to work.
Goes along with posing research questions that arise out of running a simulation or an experiment; can help in returning to topic or being precise in a group discussion.
—can note things that don't need to be implemented immediately at current stage of research, but may need to be revisited later. Allows easier searching of the code.

Team: The primary purpose of documenting a relatively finished product is to enable someone else to take over the task, possibly months or years later
Most teams in industry adhere to a set of rules for how products are to be documented

Public release: much greater detail is required for people outside your research group; your group shares an unspoken mindset and a large set of assumptions and practices that may not be obvious to outsiders
Note that data collected needs to be documented with relevant metadata on how it was collected, with some discussion of value/limitations of data set

Publicizing your work
Some internal opportunities: ECE department annual research review

Undergrads: UCLA Research and Creative Works Week
Posting to ECE department website

External opportunities
Interviews: research and open-ended design projects are much more interesting to employers than courses taken: requires full mix of skills needed to function in industry; be prepared with brief pitch, and extended discussion of details
Grant proposals: fellowships to students based on research (accomplished or proposed). Also helps to get money to support research group
Technical conferences: requires both a short paper and preparation of a talk—expert audience. Undergrad conferences may require more time devoted to context
Technical journals: longer papers are prepared, usually with revisions after peer review.

Exercise: how should work be reviewed within the research group before showing it to a broader public?

Solution: iterate
Sketch general publication plan (including presentations)
Assign people to the different parts with page limits
Assemble; then discuss critically—which arguments need more evidence, what needs to be cut, etc. Needs a tight argument will withstand outside scrutiny
Revise, revise, revise
Give it more time than you think it needs
Same rules apply to presentations: writing a script, practice is essential, with ability to handle prospective questions