Notes for lab01: what the diff

Structure

The lab is broken up into three sections.

The first section is a check-in with the students about the recent homework, how lecture is going, and their thoughts on the course. The primary purpose of this section is to help dispel impostor syndrome by reminding the students that any level of struggling is okay, and we are here to support them.

The second section is a rapid-fire review of the topics that have been covered recently.

The third section is a quick demo of `diff` syntax and terminal I/O redirection.

Notes

Intro and Goals

This document is hyperlinked from my name on the title slide

The second slide is the weekly reminder that students are worthy of this course and we are here to support them.

Slides 3-7 walk through the lab's goals. It is good to state them up-front so the students know where we are going

Check-In

Ask about hw00 and what people thought of lab00. If students are still having trouble with SFTP, VS Code, or `ssh`ing in to the servers; that is really important and they need to get that fixed in lab or in office hours **today**.

Review

Compilation Ask for an example of a compile command, then space bar to reveal the command. Step through the command, highlight each component, asking what they do/are.

Variables Raise the projector screen and project onto the whiteboard. Ask students for suggestions to fill out the missing spaces. Then advance through the builds and show more examples.

Operations As above

Terminal IO As above

Arrays Mute the projector and draw an array of integers on the board with the standard "row of boxes" notation, 5 elements long. Then write out some example c++ and ask the following questions:

- What is the length of this array?
- What is at `arr[2]`?
- What is at `arr[4]`?
- What is at `arr[0]`?
- How long is the array?
- What is at `arr[5]`?

Conditionals Drop the projector screen and unmute the projector. For each of the examples, ask the students to predict what prints (assume we are in `main()`, `iostream` has been `#include`d, and we are `using` the `std` `namespace`).

Once students have made their predictions, advance through the slide builds to demonstrate how we move through the program. (it is probably a good idea to step through this first on your own so you know what is going to highlight when)

Boolean Expressions Raise the screen and do as you did in the Variables section. But, once you step through all the answers, there is another row hidden which is revealed. It is a bug, using the assignment operator `=` instead of the equality comparison `==`.

Liam's pet peeve I hate it when people compare to `true` or `false`. Just use boolean expressions as boolean expressions! Advance to see the better way to write these two conditions.

Scope There is an example C++ program linked from the slide that has a scoping issue. The bug is annotated in the comments. Compile, look at the compiler error, then adjust the program to fix the bug. Don't forget to explain what the bug is.

`diff` demo

The lab spec is linked from the slide. It might be a good idea to click on that and show them what's going on.

I will do my demo with the `flights` program from Mark's version of CS 11. I'll write an input file, then run that input file through the solution, producing a ground truth file. Then I'll run that input file through my program to produce an output file. Then I will `diff` the two against each other and see that there are whitespace errors in my program.

Mention that we only `diff` after we've run the program manually to see if we are getting something reasonable. Blind `diff`ing is a very good and fast way to waste hours debugging.

Feedback

This form is really just for Liam's lab, but if you have feedback on the slides this is also a reasonable place to put it.