Class Introduction

Overview

- Experience Survey
- Recitations
- Grading
- Project Descriptions
- Project Notes
- Teams
- Late Penalty
- Slack Days
- Error Policy
- Submission Format

Experience Survey

- I'll pass out 10mins before the end of class
  - Someone remind me if I lose track of time
- Designed to
  - Grant access to the computer lab
    - Needed for project 2 especially
  - Give me an idea of what you know so I can better tailor the class to your needs/experience
- If you missed the survey
  - Fill one out and bring/email it to me ASAP

Recitations

- HCB 216 Friday 12:20am – 1:10pm

- Allocated recitation slots
  - Used for office hours or special workshops
  - Project 1
    - LOV 301
  - Project 2, 3
    - MCH 202

Grading

- Total grade
  - Project 1 = 30%
  - Project 2 = 40%
  - Project 3 = 30%
- Project breakdown
  - Documentation = 30%
  - Correctness = 70%
- Appeals
  - See me within one week of receiving grade
  - Otherwise, grade is permanent

Project Descriptions

- Project 1
  - Command-line user interface shell
- Project 2
  - Kernel programming
- Project 3
  - File Systems

Project Notes

- Start projects as soon as they're assigned
  - Implementation is often tricky
- Ask lots of questions
  - This gives me an idea of where you are at and can better you (and the rest of the class)
  - Popular questions will be posted on the site and mentioned at the next lecture time
- Break project into smaller programs to test new features and slowly add into actual project
- Do automate testing when possible
  - Use separate program to test each of the procedures, use cases
- Spend time to write good code / maintain code
  - Helps you and your teammates finish faster
  - Helps me in terms of grading and answering implementation questions

Teams

- Can form groups of up to 3 people
- It is *highly* recommended to work in teams
- Email cop4610t@cs.fsu.edu team makeup
  - Or if you are looking for team members
- Submissions are to blackboard and are done once per team
  - Not once for each team member

Late Penalty

- 10 points off total for each day late
- 0 points after the 5th day
- Example:
  - Due Monday @11:59pm
  - Submitted Tuesday
    - Max points possible will be reduced to 90
  - Submitted Saturday
    - Max points possible will be reduced to 50
  - Submitted Sunday
    - 0 points regardless of submission content

Slack Days

- Each student is allotted 3 slack days
- 1 slack day = 1 day past submission deadline without penalty
- Can use as many as you want for each project until you run out
- Student based, not team based
  - Example:
  - Project 2 is submitted 1 day late
  - Student A uses 1 slack day and receives no penalty
  - Student B has no more slack days and receives the 10 point penalty
- Each student needs to specify the number of slack days they want to use in the project report

Error Policy

- Document known errors/bugs
- Undocumented errors will result in full point deduction if found
- By documenting your bugs:
  - You let me know that you know things aren't working correctly
  - You can tell me different ways you tried to fix it
  - Makes it easier/quicker for me to grade

Submission Format

- .tar
  - README (5pts)
  - Project Report (15pts)
  - Source Code (80pts)
    - *.c, *.h, Makefile
- Demo (project dependent)
  - Sign up for a time to demonstrate correctness of your program to me
  - Random implementation based questions will be asked for verification purposes

README

- Team members' names
- Contents of the tar archive and a description of each file
- Version of Linux you used or the server you completed the project on
- Description of Makefile commands
- Known bugs, unfinished portions of the project
- Special considerations I should know when grading

Project Report

- The project problem statement
- Assumptions that were made
- The steps taken to solve the problem
- Brief description of why your solution uses the chosen system calls and libraries
- Problems encountered
- Known bugs
  - Same as in README

Project Report

- Division of labor
- Number of slack days used per team member
- Cumulative log entries for the entire project
  - Meeting times, modifications, decisions made, accomplishments, etc
- Responses to questions (if any)
- Descriptions of additional features
  - Extra credit

Makefile

- Required
  - If not provided, 70 points will be deducted automatically
- Used to automatically, consistently build project
- At a minimum should include commands to:
  - Build system
  - Remove build targets (executables, object files, etc)
- Additional useful commands:
  - Run project
  - Test (run a testing program)
  - Backup (archives program, commits it to a git repo, etc)

Source Code

- In C programming language
- Code quality makes up the last 10 points of the documentation
  - Code needs to be readable and have a consistent layout
    - No Junk Code!
  - Variables, structures, functions, etc need to adequately describe their purpose
    - Optionally include comments to further specify use, esp for global variables and complex functions
- Actual design decisions won't impact grading, but might slow your progress
  - Be wary of global variables, goto statements, large procedures, etc
  - When possible, it's good to have a functional design, i.e. procedures that don't introduce side effects (output depends solely on input)