Project 2

The Virtual File System

Virtual File System

- Project 2: Virtual File System
  - Handouts / Description to be published by end of week
  - Due Dates (subject to change):
    - Minimum Effort Due: February 6, 2003
    - Full Project Due: February 14, 2003
  - Note: These slides are preliminary!!!

Virtual File System

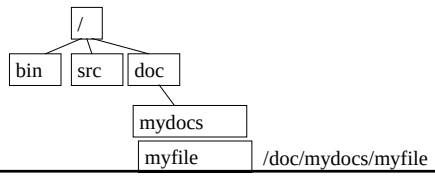
- Goal for this project is to complete a program that manages a typical file system.
  - UNIX like
  - Start with a restored file system (read from file)
  - User interface via keyboard commands

File System Entries

- The Virtual File System (VFS) consists of two types of entries
  - Text Files
  - Directories
- Each entry, regardless of type will have associated with it:
  - Name
  - Parent
  - Size
  - Permission
    - Read permission
    - Write permission

File System Directories

- Like in UNIX, directories can be nested to form a directory hierarchy
  - Root directory
  - Use of / to list subdirectories



Running VFS

- VFS can optionally be given the name of a file which contains the definition of a previously stored file/directory structure
  - In which case VFS will rebuild this structure
  - If omitted, VFS creates an empty file system with only a root directory ("/")

VFS Commands

- `ls flags entry_name`
  - If entry_name is a file, info about the file is printed.
  - If entry_name is a directory, info about directory contents is printed.
  - If entry_name is omitted, do ls on root directory
  - Flags
    - -l Give a long listing (file_name size permission)
    - -r Recursive listing
    - -lr both of the above

VFS Commands

- `access flags entry_name permissions`
  - Sets the attributes of a file or directory
  - Flags
    - -r recursive
- `backup file-name`
  - Write the contents of the current VFS into a file.
- `restore file_name`
  - Read the contents of a file and restores the VFS based on its contents.

VFS Commands

- `mkdir directory_name`
  - Creates a new subdirectory
- `rm text_entry_name`
  - Removes a text file
- `rmdir directory_entry_name`
  - Removes a directory
- `mk text_file_name size`
  - Creates a new text file

VFS Commands

- `quit (ctrl-D)`
  - Exits the program

Design – classes

- VFS – main class
- Virtual File System classes
  - Entry – entries that can exist in the file system
    - Directory – represents a directory
    - Document – represents a text file
  - VFSystem – represents the file system as a whole. All commands should result is calls to VFSystem methods.
    - Does NOT define the user interface!
  - Lot's o' Exception classes

Design – classes

- User interface classes
  - VFSView – java interface for user interface
    - VFSTextView -- gets command from keyboard input.
  - VFSCommandException – yet another exception class.

Design – classes

- VFSCommand – abstract class that represents a valid VFS command…each has an execute method.
  - VFSbackup – `backup` command
  - VFSls – `ls` command
  - VFSmk – `mk` command
  - VFSmkdir – `mkdir` command
  - VFSaccess – `access` command
  - VSFrestore – `restore` command
  - VFSrm – `rm` command
  - VFSrmdir – `rmdir` command

Classes you'll need to write

- VFS Classes:
  - Entry
  - Document
  - Directory
  - Complete the VFSystem class
  - Please use RCS

Testing your work

- You can use try to test out your classes
  - `try cs2-grd project2-test infile`
    - Will run our solution on your test data in file infile
    - You can redirect the output into a file and then compare with your output

```
- try cs2-grd project2-test infile >
  correctSolution
- java VFS infile > mySolution
- diff mySolution correctSolution
```

Submissions

- 4 submissions
  - Minimum: Entry.java & Document.java Submission 2: Directory.java
  - Submission 3: VFSystem.java
  - Submission 4: all of the above (?)
- All submissions via `try`
- NO LATE SUBMISSIONS!

Submissions

- About the minimum submission
  - Entry.java and Document.java is the minimum reasonable effort requirement for this project
    - Due Feb 6, 2003 (subject to change)
  - Must submit successfully.
    - Otherwise, you fail the course

Grading

- 100 points for functionality
  - Up to 35 point deduction for bad implementation
  - Up to 30 point deduction for bad style
    - Including non-use of RCS
- Submission percentages (subject to change)
  - Entry/Document – 30%
  - Directory – 25%
  - VFSystem – 20%
  - All classes 20%

Questions?

Next time

- Trees
  - Which you'll need for your project!
- Questions?