Assignment 4

Due Wed. 27 September 2000

- Develop a UML model for the TAU file system described in the following slides.
  (TAU stands for "TAU Ain't UNIX")
- Develop a corresponding C++ header file that will compile with an empty main program.
- Use STL (Standard Template Library) to help define components of your class.

TAU File system Specification

- There are two types of files:
  - **Ordinary**: containing a sequence of characters without further interpretation.
  - **Directory**: containing a map from names to **DirectoryEntries**, to be described in more detail shortly.

Files and Nodes

- **Files** are represented by **Nodes**. A node contains the address of the physical location of the file and other attributes, such as the owner of the file, access privileges, etc.
- For this problem, the specific attributes are not important. However, it is important that the two types of files are differentiated by separate classes of nodes:
  - **Ordinary**
  - **Directory**
- In addition, there are two *more* types of Nodes:
  - **Link**: Effectively a pointer to another node
  - **Alias**: A non-empty list of names that identify another node, in a manner to be described.

DirectoryEntries

- Each **DirectoryEntry** is accessed by **Name** and contains a pointer to a **Node**.
- Within a given **Directory**, no two **Nodes** are ever accessed by the same name.

Aliases

- An **Alias** identifies a *path* symbolically (a sequence of names), starting at a *specific directory*, which is an immutable attribute of the Alias established when the Alias is created.
- The names in the Alias are used to look up nodes in directories recursively.

Unaliasing

- The first name in the path is looked up in the Alias' directory. That DirectoryEntry is translated to a Node.
- If the Alias' path has more names, then the Node should be a Directory Node, and the process is repeated with the remaining path and the Directory thus reached.
- If the Alias path has no more names, then the Node itself is the result.

Current Directory

- The classes mentioned will be used in an application “tau” in which there is always a notion of “current” directory in every system state.

Absolute Path and Root

- There is a distinguished directory called the **root**.
- A path is either
  - **absolute**: starts at the root, or
  - **relative**: starts at the current directory

Envisioned Use Cases (1 of 3)

- tau has the following use cases:
  - **cd**: Set current directory to one specified by a path.
  - **cp**: Create an ordinary file in a specified directory by copying contents of another file.
  - **cpa**: Create a directory in a specified directory by copying a directory recursively.
  - **create**: Create an ordinary file in a specified directory (current or otherwise) by giving contents as a string.

Envisioned Use Cases (2 of 3)

- **ln**: Establish, in a specified directory, a link to a specified file.
- **lns**: Establish, in a specified directory, an alias according to a specified path.
- **mkdir**: Create a directory in a specified directory. **mv**: Move a file to another directory.
- **ren**: Rename a file.
- **rm**: Remove the identified file or directory from its directory.

Envisioned Use Cases (3 of 3)

- Use your imagination to come up with 3 more use cases, which you will name.
  -
  -
  -

What to Turn In

- Hard-copy drawing of your UML class diagram, using a professional tool, such as Powerpoint, Visio, TopDown, Rose, etc.
- Email ascii compilable header file and companion main program that loads it (use < 80 chars per line, to avoid MIME encoding).