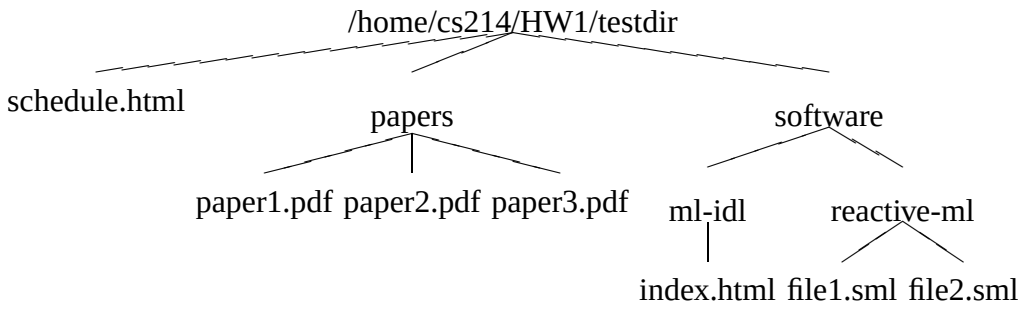
Advanced Unix Tools	**Homework 1**
CS214 Spring 2004	**Due Date: Friday, March 5, 2004**

The purpose of this homework is to write a script to "traverse" a directory hierarchy. What do I mean by traverse? Consider the following directory hierarchy, which is best visualized as a tree:



Traversing the directory hierarchy simply means going through all the nodes in the above tree, and performing an action at every node. Let me call that "acting on a node". Note that there are nodes corresponding to directory names, and nodes corresponding to file names. How do we act on a node in the directory hierarchy? Rather than bake that into our script, we'll take as arguments to the script what action to do on a directory name node, and what action to do on a file name node.

There's a bit of a subtlety in writing such tree traversal procedures. It has to do with the order in which we perform actions. More precisely, suppose we are at node `papers` in the above hierarchy. Do we first act `papers` itself, and then traverse the children (`paper1.pdf`, `paper2.pdf`, `paper3.pdf`), or do we first traverse the children, and only then act on `papers`? The first approach corresponds to a preorder traversal (pre- means before, meaning that we act on the node before traversing its children), the second corresponds to a postorder traversal (post- means after, meaning that we act on the node after traversing its children). See for instance `http://www.brpreiss.com/books/opus5/html/page259.html` for more info on tree traversals. Note that we're not just dealing with binary trees here.

Tree traversal is obviously a recursive procedure. Just look at the way I said things above, in the preorder case: to visit a node, first act on the node, then visit its children. That's an obvious recursive procedure right there.

To give you some feel for the two kinds of traversals, here is the order in which the nodes above are acted on, in a preorder traversal:

(1) `/home/cs214/HW1/testdir`

(2) `schedule.html`

(3) `papers`

(4) `paper1.pdf`

(5) `paper2.pdf`

(6) `paper3.pdf`

(7) `software`

(8) `ml-idl`

(9) `index.html`

(10) `reactive-ml`

(11) `file1.sml`

(12) `file2.sml`

Compare it with a postorder traversal:

(1) `schedule.html`

(2) `paper1.pdf`

(3) `paper2.pdf`

(4) `paper3.pdf`

(5) `papers`

(6) `index.html`

(7) `ml-idl`

(8) `file1.sml`

(9) `file2.sml`

(10) `reactive-ml`

(11) `software`

(12) `/home/cs214/HW1/testdir`

If you understand how to get the above orders, then you understand pre- and postorder traversal of trees. Congratulations.

The script: Write a script called `traverse` that takes exactly 4 arguments:

(1) the string `preorder` or `postorder`;

(2) the name of a command to act on the directory name nodes;

(3) the name of a command to act on the file name nodes;

(4) the path to a directory to traverse.

The first argument simply indicates whether to do a preorder or postorder traversal. The next two arguments are actions to perform on the nodes. We distinguish between the action on directory name nodes and the file name nodes. The last argument is just the directory to traverse.

(It is possible to have loops in a directory hierarchy, using symbolic links. Don't worry about that case. Let me be clear: you do not have to write your script so that it terminates if it encounters loop. It's fine if your script never terminates on directory hierarchies with loops.)

The action to perform should be a command that expects one argument, the name of the node (either the directory name, or the file name). The actions to perform can be anything that the shell will interpret as a command. For instance, it can be another script (depending on how you code your script, the full path to any script given as an argument may be needed for the system to be able to find it!) It can also be a build-in command. Let's see some examples.

Examples: For testing purposes, I've put the above directory hierarchy in ∼`cs214/HW1/testdir`.

If you execute the following, your script should gives you a list of nodes as in the above:

```
$ ./traverse preorder echo echo ~cs214/HW1/testdir
/home/cs214/HW1/testdir
/home/cs214/HW1/testdir/schedule.html
/home/cs214/HW1/testdir/papers
/home/cs214/HW1/testdir/paper1.pdf
/home/cs214/HW1/testdir/paper2.pdf
/home/cs214/HW1/testdir/paper3.pdf
/home/cs214/HW1/testdir/software
/home/cs214/HW1/testdir/ml-idl
/home/cs214/HW1/testdir/index.html
/home/cs214/HW1/testdir/reactive-ml
/home/cs214/HW1/testdir/file1.sml
/home/cs214/HW1/testdir/file2.sml
```

Similarly:

```
$ ./traverse postorder echo echo ~cs214/HW1/testdir
/home/cs214/HW1/testdir/schedule.html
/home/cs214/HW1/testdir/paper1.pdf
/home/cs214/HW1/testdir/paper2.pdf
/home/cs214/HW1/testdir/paper3.pdf
/home/cs214/HW1/testdir/papers
/home/cs214/HW1/testdir/index.html
/home/cs214/HW1/testdir/ml-idl
/home/cs214/HW1/testdir/file1.sml
/home/cs214/HW1/testdir/file2.sml
/home/cs214/HW1/testdir/reactive-ml
/home/cs214/HW1/testdir/software
/home/cs214/HW1/testdir
```

If you write your script correctly, you should also be able to execute something like this:

```
$ ./traverse preorder 'echo Directory: ' 'echo File     : ' ~cs214/HW1/testdir
Directory: /home/cs214/HW1/testdir
File:      /home/cs214/HW1/testdir/schedule.html
Directory: /home/cs214/HW1/testdir/papers
File:      /home/cs214/HW1/testdir/paper1.pdf
File:      /home/cs214/HW1/testdir/paper2.pdf
File:      /home/cs214/HW1/testdir/paper3.pdf
Directory: /home/cs214/HW1/testdir/software
Directory: /home/cs214/HW1/testdir/ml-idl
File:      /home/cs214/HW1/testdir/index.html
Directory: /home/cs214/HW1/testdir/reactive-ml
File:      /home/cs214/HW1/testdir/file1.sml
File:      /home/cs214/HW1/testdir/file2.sml
```

You should be able to supply an arbitrary command as an action, as long as it takes an argument, the name of a directory or file name, and does something with it.

I'll try to put some interesting scripts to use with `traverse` in the `/home/cs214/HW1` directory.

Submitting your script: Submit your script according to the instructions on the course web site.

To think about:

(1) Clearly, we don't need to provide two scripts, one for directories, one for files. Can you see how the above would work with only one script argument?

(2) Right now, there is no communication between the script invoked on the files, and the script invoked on the directory. Thus, it is not possible, in a postorder traversal, to use the information build up from the traversal of the children of the current directory when traversing the current directory, unless that information has been left somewhere. Can you think of a good way to get around this problem?