COP5570 Programming Assignment 1 (Warm-up)
A Primitive Centralized File Organizer

OBJECTIVES

- Get familiar with the UNIX programming environment
- Experience with distributed systems
- Use 'make'

DESCRIPTION

Managing files in different machines can be tedious without a global view. In this project, you will develop a primitive file organizer that creates a global view of all files in different machines. The file organizer allows (1) files to be copied between machines (cp), (2) files on different machines to be displayed (cat), (3) directories to be made on different machines (mkdir), and (4) the content of directories to be displayed (ls). Detailed description follows.

1. The file/directory in the organizer is identified by a global path descriptor of the form 'machine:path'. To limit the scope of your global file system, a root path is associated with each machine managed by the file organizer. The machines and the root paths are stored in a configuration file where each line in the file contains the machine name and the root path. An example configuration file follows:

   ```
   xyuan   # loginname in the first line
   linprog1 /home/faculty/xyuan/tmp # machine dir pair all other lines
   linprog2 /tmp/xyuan
   linprog3 /tmp/xyuan
   ```

   Using this example configuration file, the global path descriptor 'linprog3:/new/example1.c' identifies the file /tmp/xyuan/new/example1.c on linprog3.

2. The command 'a.out config_file' starts the file organizer.

3. The file organizer should support the following commands.

   - **cp src_file dst_file**: copy src_file to dst_file. Both src_file and dst_file are global path descriptors. In the implementation, the command 'cp machine1:path1 machine2:path2' should be realized by command
     'ssh -q machine1 scp -q root1/path1 username@machine2:root2/path2'.
   - **cat filename**: display the file. Here, filename is a global path descriptor.
   - **mkdir dir**: make a new directory dir, which is a global path descriptor.
   - **cd dir**: change directory to dir (a global path descriptor).
   - **ls**: display the content in the current directory.
   - **ls dir**: display the content in dir.
   - **quit**: quit the program.

4. You can assume all user inputs are correct.

5. You should try the sample executable to see how the program should behave.

GRADING POLICY

This program should be developed on the linprog stack. A proper makefile should be used in this project: (1) Command 'make' should produce the executable; (2) Command 'make demo' should run the program (use the existing executable if it has been generated or produce the executable if it is not available); (3) Command 'make clean' should remove .o files and executables from the directory; and (4) the program should be compiled with '-Wall -ansi -pedantic' flags. Compiling the program with flags (-Wall -ansi -pedantic) should not have any warning message.

- Proper makefile (10 pts)
- 'cp' (5 pts)
- 'cat' (5 pts)
- 'mkdir' (5 pts)
- 'ls' (5 pts)
- 'cd' (5 pts)
- 'quit' (5 pts)
- Demo (10 pts). You must design your demo to show all features in your program WITHIN THE GIVEN TIME. Failing to do so will lose not only the demo points, but also the points for the features that you do not have time to demo. You cannot touch your source code (unless you are asked to) during (and after) demo. One will lose all 10 points if the source code has to be modified during demo.
- **20 points deduction for any unknown bug of any kind caught during/after the demo**
- 5 points deduction for each warning message.
- Reporting a bug in the sample executable gets extra 5 pts.

DEADLINES AND MATERIALS TO BE HANDED IN

- **Sept. 13**. Demo (10 mins) + soft and hard copy of your programs.

MISCELLANEOUS

All programs will be checked by an automatic software plagiarism detection tool.