Homework 4

Due Date: March 4, 1998 at 11:59PM **Points**: 100
Late Date: March 6, 1998 at 11:59PM (20% penalty)

1. (*100 points*) In the *Puzzle for the Day* for February 23, the function *mpopen* provides a safe environment for the execution of the given command. Specifically, it first deletes the caller's environment, and sets **only** the following environment variables: **PATH** (to "/bin:/usr/bin:/usr/ucb:/usr/sbin:/sbin") **IFS** (to " \t\n"), **SHELL** (to the current value of the **SHELL** environment variable, or to the Bourne shell if **SHELL** is not set) and **TZ** (to "PST8PDT"). It closes all file descriptors that will not be used. It sets all signals to their default value, and *umask* to 077, the effective UID to the real UID, and the effective GID to the real GID. It then spawns the shell named in **SHELL** to execute the given command.

   The stream opened by *mpopen* is closed by *mpclose*.

   The interface to *mpopen* and *mpclose* is identical to that of *popen*(3) and *pclose*(3), as is their functionality.

   You are to write these two library functions.

   Hint: The process environment should be changed *after* the *fork*(2). if you change it before the *fork*, you will change the parent process' environment, which is not correct!

   Note: you may use any language you like, but the result ***must*** be a compiled library that can be called from a C program. We will supply our own driver for grading!

Extra Credit

2. Implement a variant of the *system*(3) command that uses the same protections as the *mpopen*/*mpclose* library functions, above.
3. Implement a set of library functions to enable the caller to control the environment set in *mpopen*:
   a. *int le_set*(*char *env*) takes an environment variable and associated value in the form "VAR=value". Subsequent calls to *mpopen* will add that environment variable and value to the subprocess. If *env* is "VAR=", the variable is defined to have the empty value. If *env* is "VAR", the value from the current environment is used. This function returns 1 if the argument was successfully added, and 0 if not.
   b. *int le_unset*(*char *env*) deletes an environment variable from the environment of subsequent calls to *mpopen*. The argument *env* names the variable, so to delet "VAR", say "le_unset("VAR")". This function returns 1 if the argument was successfully deleted, and 0 if not.
   c. *void le_clobber*(*void*) forces the next call to *mpopen* to use the standard *mpopen* environment.
   d. *int le_umask*(*int umask*) changes the value of *umask* as indicated in the environment of subsequent calls to *mpopen*. This function returns 1 if the change succeeds, and 0 if not.