Project 2 Phase 2: User Manager

For this phase, you will implement a user manager (UM) to manage multiple simulated users using the system. The executable for UM should be "UM.exe". Each user has a $\boldsymbol{userid}$, which is of type integer. Since UM reads its commands from one window, we need to attach $userid$ for every command. The five UM commands are "$\boldsymbol{userid}$ login", "$\boldsymbol{userid}$ logout", "$\boldsymbol{userid\ shshcmd}$", "$\boldsymbol{userid}$ pre", and "0". The userid value starts from 1 and 0 in the userid field means to terminate the UM. UM prints out command prompt "ucmd> " and reads and handles user commands.

In Unix systems, a shell process is created upon each user login. Similarly, when a user logs in to UM, UM uses fork() to create a Shsh process (U4) for the user. UM also creates pipes to communicate with each user's Shsh (U1). For each "$userid\ shshcmd$" command, UM simply passes the shshcmd (without userid) to the Shsh process of the corresponding user via one pipe and waits for its response from the other pipe (U6*). Note that you need to change your Shsh program to run as functions within the UM. If you use exec() to run the Shsh process, the pipe and current directory settings cannot be passed. (You can use named pipe if you use exec(). But we will not explore it in this project.)

To avoid garbling the output of different users, UM will create one file for each user to store the outputs of the user (U2). For a user with userid = uid, its output should be written to "$\boldsymbol{uid}$.out" by UM.

A shshcmd may require multiple rounds of user-Shsh interactions (e.g., shshcmd = "cmd ./a.out" and a.out needs to read input data). Also, one shshcmd may produce multiple output objects. For the first case, the user would know what Shsh wants and can send the subsequent input data. To safely pass the inspection of the UM, we use the "pre" command to indicate that the message is following the previous shshcmd. In the second case, UM should read the pipe from Shsh continuously till it receives the next Shsh command prompt "cmd> " (U6**).

Note that each user of the UM should have a consistent view of her/his own Shsh environment. For example, if user $u1$ creates a file "xyz.c", "xyz.c" should not appear in user $u2$'s environment. To achieve this, we let each user have a home folder and assume that each user will always work within her/his own home folder and its subfolders. When UM creates a Shsh process for a user with userid = uid, it also creates a subdirectory "$\boldsymbol{uid}$.dir" as her/his home folder (U3).

A user's Shsh process will have the same current working directory (cwd) as the cwd of the UM at the Shsh process creation time. Thus, before handling any shshcmd, the Shsh process needs to change its cwd to the home folder of the user using the chdir() system call (S2). Thus, upon creating a user's Shsh process, the UM should send user's userid (integer) and the home folder path of the user (string) to the user's Shsh process (U5) via the corresponding pipe and the Shsh process should read the information from the pipe (S1). Just like Unix login, the user starts from her/his home folder. But unlike Unix, the user will not work beyond her/his home folder.

After all the initial steps, Shsh should read commands from its input pipe, executes them like before, and writes the response to its output pipe. To ensure that the output of the "cmd" commands and the last shcmd in a pipe command) are delivered to the output pipe of the Shsh process, we need to map stdout of the Shsh process to the output pipe (dup2). Similarly, we need to map stdin of the Shsh to the input pipe in order for a command that requires inputs from stdin (like running ./a.out) to properly get its inputs. These have to be done before any "system()" system call is invoked (S3).

When getting the logout command, UM translates it to "exit" and sends it to the corresponding Shsh process for termination. You need to close the corresponding pipes accordingly. Also, your Shsh printouts should now be “Shsh process ($\boldsymbol{userid}$, $\boldsymbol{pid}$) has been created by process $\boldsymbol{ppid}$” and “Shsh process ($\boldsymbol{userid}$, $\boldsymbol{pid}$) terminates”. The printouts defined in the previous phase regarding each shshcmd received and processes and pipes created for each pipe command stay the same. Note that the printouts mentioned above are system information and should be printed out by Shsh, only appear on the monitor, not to be sent to UM and printed in the user file. All other outputs (related to the execution of shshcmd, besides those within the pipe command) should be delivered to UM and printed to the user file by UM.

We assume correct user behaviors. Each user (same $userid$) will not make multiple logins before logging out. The commands issued by the users are correct and conforming to the required format. No commands with non-explicit interactions (e.g., text editors) will be issued.