UNIX Command Interpreter (Shell)

Assignment Overview:

Design and implement a C program that runs as a basic command-line interpreter (shell).

Assignment Specifications

1. Let PROG1 be the name of your executable. If the program is invoked as "PROG1 –c LIST" then the program executes the command given by LIST and terminates. For example "PROG1 –c ls –l" (without quotes) will execute the command "ls –l" and terminate. (In this option, built-in commands (listed below) are not allowed)
2. Without the –c option, the program repeatedly displays a prompt containing the sequence number of the current command (starting at 1) and the name of the machine. This information will be enclosed in the characters '<' and '>'. For example: <1 adriatic >. The sequence number and machine name will be separated by a singe space.
3. After displaying the prompt, the program will read one line of input from the user. An input line is defined as a sequence of zero or more tokens (character strings), separated by one or more delimiters (blanks and tabs), ending with a newline character. The characters '<', '>', ';', and '|' are defined to be separate tokens. There will be no more than 128 characters in a line. The first token may be preceded by any number of delimiters.
4. If the first token is the name of a built-in command (listed below), then the program will take the appropriate action. Otherwise, the program will assume that the command is the name of a file containing an executable program.
5. The program will recognize the following built-in commands:

Hist	Display the last 10 input lines
curPid	Display the process ID
pPid	Display the parent process ID
cd Dir	Move from the current directory to Dir, where that token may be an absolute or a relative pathname
quit	Terminate the current process

6. Command 'hist':
   Will display the most recent input lines entered by the user (along with the sequence number). An example of partial 'hist' output is as follows:
   197 hist
   198 cd executable

199 sample
200 curr
201 curPid

7. The history list will hold up to ten non-null input lines, including any lines with misspelled commands.
8. For external commands, the program will create a child process; the child process will use a member of the 'exec' family of system calls to execute the external command.
9. The program will perform appropriate error handling. It will display appropriate message if the user's command fails in any way.
10. Implementation of ";". A list of commands separated by ";" will be executed one after another. I.e., a command of the form "ls ; pwd ; pine" will result in execution of "ls".
    Once that terminates, pwd will be executed. Subsequently, "pine" would be executed. Command prompt will be printed after all commands are executed.
11. Pipeline Feature:
    The user may give commands in a pipeline, which is a sequence of 2 or more commands separated by the token '|'. For example: ls –l | grep lib | less. In this example, the output of 'ls –l' will act as the input to 'grep lib' command. In other words, the 'grep' command will search for 'lib' in the output of 'ls –l'. And, the output of the 'grep' command is the input to 'less'.
12. Input output redirection:
    User may specify input output redirection using < and > respectively. For example, 'ls > testfile' would save output into file 'testfile'

Assignment Deliverables

The deliverables for this assignment includes the following files:

1. prog01.*.c - the source code file(s) for your solution.
2. prog01.*.h - the interface file(s) for your solution
3. prog01.makefile - a makefile which produces the executable for your solution