COP290: Assignment 2

Writing a basic shell

Write a program in C to act as a command interpreter (Shell). The shell will give a prompt for the user to type in a command (from a set of commands), take the command, execute it, and then give the prompt back for the next command (i.e., actually give the functionality of a shell). Your program should do the following:

- Give a prompt "shell>" for the user to type in a command
- Implement the following *builtin* commands:
    - **cd "dir"** : changes the directory to "dir"
    - **pwd** : prints the current directory
    - **mkdir "dir"** : creates a directory called "dir"
    - **rmdir "dir"** : removes the directory called "dir"
    - **exit** : exits the shell

  The commands are the same as the corresponding Linux commands by the same name. Do "man" to see the descriptions. You can use the standard system calls chdir, getcwd, mkdir, rmdir etc. to implement the calls (standard C library functions are available for these; look them up). These commands are called *builtin* commands since your shell program will have a function corresponding to each of these commands to execute them; no new process will be created to execute them.
- Any other command typed at the prompt should be executed as if it is the name of an executable file. For example, typing "a.out" should execute the file *a.out* . The file can be in the current directory or in any of the directories specified by the PATH environment variable. The file should be executed after creating a new process and then exec'ing the file onto it. The parent process should wait for the file to finish execution and then go on to read the next command from the user.
- Should be able to redirect the output of a program to a file using ">" and read the input of a program from a file using "<". For example, typing "a.out > outfile" should send whatever was supposed to be displayed on the screen by a.out to the file *outfile* . Similarly, typing "a.out < infile" should make a.out take the inputs from the file *infile* instead of the keyboard.
- Should be able to redirect the output of one command to the input of another by using the "|" symbol. For example, if there is a program *a.out* that writes a string "abcde" to the display, and there is a program *b.out* that takes as input a string typed from the keyboard, counts the number of characters in the string, an displays it, then typing "a.out | b.out" at your shell prompt should display 5 (the output "abcde" from *a.out* was fed as input to *b.out* , and 5, the number of characters in "abcde", is printed). Use the pipe command.

To run your shell, write another C program that will create a child process and call an appropriate form of exec to run the program above. The parent process simply waits for the child to finish (execute the "exit" command), after which it also exits.
Name the file containing your C code for the shell <your roll no>_sh.c (for example, 06CS1004_sh.c). Submit only this C file.