Yilin Yang

CSC345

Project 1

Part I: Child process creation

Child creation is handled in the *execute* function. The child is created and fed parameters used to execute a command using execvp(). These parameters are taken from what the user inputs and parsed through a parse function such that each standalone term is recognized (i.e. ps –ael is recognized as “ps” and “-ael”) and store into an argument array. The *parse* function uses white spaces, including tabs and enter keys, to delimit these terms before storing them in args. *Parse* and *execute* are called anytime a command is enacted, most frequently in *main* but in other functions as well.

Part II: Command history management

History management is handled in the *history* function. Up to 10 commands are remembered and stored in a hist array. This recorded as soon as the user enters an input in *main*. Exceptions to this is if the command is retrieving history, as this would redundantly inform they user they are trying to access history. The *history* function determines what specific command the user is retrieving by identifying the second character of the input. Inputs can include !! or !n where n is an integer between 1 and 10. The function retrieves the appropriate command based on the second character based on how many commands are currently stored. In the event more than 10 commands are entered, the oldest command in the first index of the hist array is overwritten. The function accounts for this by taking the modulus of the current index by 10.

Part III: Directory manipulation

The commands to create, rename/move, copy, and delete files are supported and should be recognized by execvp when instructed. These commands include vi, cp, mv, and rm.

Part IV: Arrow key input

History can be navigated through the use of arrow key inputs similar to Linux. The *arrow* function handles this by switching input style to noncanonical mode. The function begins by identifying the most recent command, searching through the hist array in reverse order. The index containing the command (or 0 if no command is entered) is marked as a threshold; commands cannot be retrieved after this index since they do not exist. From here, a while loop scans user input for up or down arrow keys, incrementing or decrementing the index respectively to print the corresponding command. The index cannot be decremented less than 0 or greater than the threshold (the most recent command). Once the user presses the enter key, the command is executed and the program returns to main in canonical mode.

Part V: Batch processing

Batch processing is supported within the *main* function. If a test file is requested, the function searches if such a file exists, and if so, reads it for commands, passing them to *parse* and *execute*. These commands may include examples such as echo or pwd.