Usage

./<shell>

Shell uses > as a prompt. SIGINT (Ctrl+C) shows history, SIGQUIT (Ctrl+\) allows user to exit. If shell waits without showing prompt, press ENTER to get prompt.

Implementation

In situations with multiple pipes between sub-commands (eg ls|wc|wc|wc), we use recursion. An execute function is called, that sets up a pipe between the first sub-command and the rest of the command, then forks and executes each independently.

The base case of this recursion is a simple command, this is run on a child using the execv call. Each path in the PATH variable is checked for the existence of the command. When found, the command is added to the path to obtain the file, and an execv call is made.

Redirection is also implemented with pipes, by overwriting the stdin or stdout file descriptor with pipes. This is handled while parsing the relevant sub-command. Further, in cases where redirect operators contradict pipes, redirections are given precedence. For example, if our input was ls > output.txt | wc, output.txt will get the file list, while wc directly gets EOF.

To handle double (||) and triple (|||) pipes, we construct our own version of the tee command - the data flows to a separate tee process, which copies it to pipes to each of the destinations. This process terminates when the incoming pipe is closed.

We keep track of the history within the main procedure, this procedure keeps track of all the command that the user has entered so far inside of an array. For each command being executed, the procedure waits on the return value of that command and stores it in a separate array, along with the command itself.