Step #1

1. Prompt the user for string until they type “exit” or press Control-D (or Control-Z on Windows)
2. Separate the each string by pipes
3. Output each set of words on a line by themselves

Step #2

For each non-pipe string, use fork() and execvp() to execute the program where the first word is the program and each successive word is an argument

Examples: `ps aux | ls`

Hint: You may find `wait()` or `waitpid()` helpful.

Step #3 - Redirect Input/Output

If the string contains:

- `>` – redirect the stdout of the final program to the file

Example: `ls -al > file.txt`

- `<` – read the file after the `<` and send it as input to the first program

Example: `grep root < file.txt`

Step #4 - Tricky

For each non-pipe string, use the pipe() function to connect the stdout of the first process to the stdin of the second.

Examples: `ps aux | grep root`

Step #5 - Hard...

Implement each signal on the previous slide:

- SIGINT - stop the program
- SIGTSTP - suspend the program
  - If “bg” is then typed, run the process in the background
  - If “fg” is typed, bring the process to the foreground and continue running it