COSC 3750
Linux Programming
Homework 10

## 1 Basic info

This is Step III of writing our shell program . We go from executing programs with arguments to I/O redirection including pipes.

Notes:

1. There can be at most one standard output redirection, one standard error redirection and one standard input redirection.
2. That also means that the left side of a pipe must not also redirect output. The right size of a pipe must not also redirect input.
3. The order of redirection is important. The command line

   ```
   command 2>1 > filename
   ```

   redirects *command*'s standard output to *filename* but its diagnostic output to standard output. On the other hand

   ```
   command > filename 2>1
   ```

   redirects *command*'s standard output to *filename* and its diagnostic output will also go to *filename*.
4. $2 > 1$ redirects standard error to wherever standard output is currently directed.
5. You should make SURE that you do not call *parse_line()* after getting an EOL. It will just continue to return EOL giving an endless loop. Make sure that you download the LATEST versions of the scanner code to avoid a segfault in this situation.

## 2 What you will need

For this version you will need your program from Homework 10. Additionally, you will need the files *wyscanner.c* and *wyscanner.h*. Make sure you get the versions from Homework 10.

You will need to include "wyscanner.h" in your program, add wyscanner.c to the compilation.

```
gcc -Wall -std=gnu99 -ggdb wyshell.c wyscanner.c -o wyshell
```

Please look in *wyscanner.h* for the return values from *parse_line().* You will also see **extern** declarations of *parse_line*, the buffer *lexeme* which will contain a valid string if the return value is WORD and the character *error_char* which contains the offending error character if the return value is ERROR_CHAR. Note that the defined values for the tokens are such that the error values are less that EOL (end of line) and the valid tokens are all greater than EOL.

## 3 What you will create

Your job is to write a program named **wyshell.c** that

1. Prints a prompt, "`$> `". There is one space after the `>`.
2. Reads the lines of user input.
3. Handles program execution like the last assignment (HW10) **plus**
   (a) open files for input, output, and diagnostic output;
   (b) dup2() these files to standard input, output, error as appropriate;
   (c) create pipes to connect two processes and dup2() the standard input/output of those processes as needed;
   (d) close excess files and pipes.
4. After the process(es) execute(s), print the prompt again, just like the "real" shells.
5. If the command line ends with the ampersand (&), do NOT wait on the process(es) to finish before redisplaying the prompt. But make sure, that at some point in time as soon as practicable, your shell waits on them in order to "kill the zombies."
6. After a QUOTE_ERROR or ERROR_CHAR return or detection of a grammar error, that line of text is abandoned and the next line is processed.
7. After SYSTEM_ERROR, the program exits.
8. Reasonable error messages will be printed for syntax errors and for character or quotation errors.
9. When the read from `stdin` returns End-Of-File, the program will exit.

## 4 Turn in

You will upload a tar archive, named **hw10** that contains **wyshell.c** and any other files you may have that need to accompany it, to the assignment on WyoCourses. This will of course include a **Makefile** but will NOT include *wyscanner.c* or *wyscanner.h*. Make sure that all the required files are there. I will not ask you to email me any missing files this time. If the archive is NOT compressed the extension will be **tar** and if it is compressed will be **tgz**. So upload either **hw10.tar** or **hw10.tgz**.