Homework 3 - Simple Shell

Due by 4:00 p.m. Wednesday, 3/3/16

For this assignment you will implement your own shell that runs on top of the regular command-line interpreter for Linux. Your shell should read lines of user input into a 1024 byte buffer, then parse and execute the commands by forking/creating new processes. For each command, your shell should call `fork()` followed by `execvp()`. Following each command, your shell should wait for its child process to complete, and then reprint the command prompt. The user should be able to specify the command to execute by giving a path to the executable file (*e.g.* `/bin/ls`) or by using path expansion to locate the executable file (*i.e.* searching each directory in the `PATH` environment variable). (Note that the `execvp()` function perform this processing automatically; you do not need to program it yourself.)

If your shell encounters an error while reading a line of input it should report the error and exit. If your shell encounters EOF while reading a line of input, it should exit gracefully without reporting an error. Ensure that you do not overflow your 1024 byte buffer when fetching the line of input (functions that do not accept the size of your buffer are not able to prevent overflows whereas functions that do accept a size generally do; be sure to check the manpage of any function you use carefully). You do not need to report an error if the user's input line is larger than the 1024 byte buffer; just use the truncated input as the command.

Before your shell forks a new process to call `execvp()`, it should parse the input string and separate it into a collection of substrings representing the executable file and any command-line arguments. If the user entered an empty line, report an error and fetch a new line of input. Your code must handle up to four command-line arguments (not including the executable file itself). If the user enters a command with more than four arguments, report an error and fetch a new line of input.

You should store pointers to the substrings in an array (similar to the "`argv`" array passed to `main()`) and pass this array of arguments to `execvp()`. Note that the number of command-line arguments is variable; this is indicated in the array by including a `NULL` pointer in the array after the last substring. (This means that if the user specifies *N* substrings, your array must hold *N + 1* pointers where the last pointer is `NULL`.) If the user enters the `exit` command, your shell should terminate (returning to the regular shell).

Here is a sample execution:

```
> /bin/ls -l /dev/null
crw-rw-rw-  1 root  wheel    3,   2 Feb 21 15:26 /dev/null
> echo these are some args
these are some args
> echo this is too many args
Too many arguments
>
Empty command
> exit
```

You should submit your source code file(s) and a short writeup in PDF format that includes a description of what you did and the compilation and execution output from your program. Your execution output should include commands with command-line arguments. Then use the exit command to exit your program and show the output of the same commands in the regular command-line interpreter for that machine to ensure they match. Submit everything to the regular submission link on iLearn, and then submit just the writeup to the TurnItIn link to generate an originality report.