Project 3 – Command Executor

Objectives:

In this project, you will implement a basic command executor which will help you gain experience in using Unix system calls.

Requirements

By now, you should be comfortable using the BASH shell to execute programs and utilities. You should also know by now how to use input, output and error redirection i.e. using the <, >, >> and >& symbols on the command line to get/send the input/output of a command to a file. Additionally, you should know how to use pipes (the '|' symbol) to send the output of one program to become the input of another program e.g.

```
$ ls | wc
```

The command above sends the output of the ls utility to the input of the wc utility. Only the output of the wc utility is shown on the terminal.

For this project, you have to implement a program which will execute a given command(s) as command line arguments to your program. You will need to support input, output and error redirection and pipes. However, instead of using the symbols used by the shell, you will use the following symbols:

```
LT          for         <
GT          for         >
GTGT        for         >>
GTAMP       for         >&
PIPE        for         |
```

The name of your program should be 'proj3'.

The first argument of your program should be either of the two words 'silent' or 'verbose' (their effect will be detailed below).

Some examples are given below (the corresponding shell command is given first, followed by the project 3 command which should give the same result):

Shell Command: `$ ls`
Project 3: `$ ./proj3 silent ls`

Shell Command: `$ ls -al`
Project 3: `$./proj3 silent ls –al`

Shell Command: `$ ls > out`
Project 3: `$./proj3 silent ls GT out`

Shell Command: `$ ls >> out`
Project 3: `$./proj3 silent ls GTGT out`

Shell Command: `$ wc < input > output`
Project 3: `$./proj3 silent wc LT input GT out`

Shell Command: `$ ls >& out`
Project 3: `$./proj3 silent ls GTAMP out`

Shell Command: `$ ls | wc`
Project 3: `$./proj3 silent ls PIPE wc`

Shell Command: `$ ls | wc | wc > out`
Project 3: `$./proj3 silent ls PIPE wc PIPE wc GT out`

A few of test runs of the reference implementation:

```
$  ./proj3 silent ls
 makefile  out  proj3  proj3.c  proj3.o

Command(s) execution complete.


$ ./proj3 silent ls PIPE wc
     5       5      35

Command(s) execution complete.


$ ./proj3 verbose ls PIPE wc
     5       5      35

ls        : 23533
wc        : 23534
Command(s) execution complete.
```

Notice in the last example, the verbose argument is used, which will cause the process id's of the created processes to be displayed right before your program ends.

Strategy/Tips:

1. The name of the executable must be proj3
2. Your program should execute the command(s) specified by the command line arguments, display any results and then exit (run the reference implementation to see how it works).
3. For each command that needs to be executed, fork a separate process to run it. The fork() and execvp() system calls will be useful.
4. For input, output and error redirection, you will need to create/open the file(s) as required and then redirect the stdin, stdout and stderr file descriptors (file descriptors 0, 1 and 2 respectively) as appropriate. The dup(), dup2(), open() and close() system calls will be useful.
5. You will need to create and use pipes to perform interprocess communication between piped commands. The pipe() system call will be useful.
6. You will need to keep a track of child processes' process ids to print them if needed. Also, The parent process will need to wait for the last child process to finish before it can exit/return. The waitpid() system call will be useful for this.
7. To make your work easier, your program will not be tested with more than 5 piped commands and each command can have a maximum of 5 arguments each.
8. It is recommended that you implement the required functionality step by step e.g. first try to execute one command like 'ls' and only go forward when you have this working. You will get partial credit, so always ensure that you have a running program at each step even if you cannot complete the entire project.

Due Date:

Saturday, 6th April, 2019. Submission instructions will be given to you later.