CS 377: Operating Systems Lab

Assignment 3

This handout concludes Assignment 3. Today, you will extend on the JASH you built last time. It has been specifically split into two parts, since the parts are quite distinct. **Please read the entire assignment (both parts) and the coding and submission directives before you begin.**

You were told to read about certain functions and commands. If you have not done so, we recommend that you do so before you begin the assignment. You are expected to complete the assignment, as usual, in teams of two. And importantly, observe the deadline. **We will be strictly NOT accepting any late submissions for any of the parts.** The submission deadline for Part B is 6.00 pm and that for Part C will be midnight. Try not to keep the submission part till the last minute; Moodle is not really reliable for submitting your assignment 30 seconds before the deadline.

PART B: The Shell Strikes Back

30 January 2015

This part aims to implement redirecting IO streams for processes. Besides, all commands and utilities mentioned in Part A need to work as well.

I/O Stream Redirection

You need to redirect input/output streams from/to files as is done on bash using appropriate symbols. The *jash* should have the following built-in tokens: **<**, **>**, **>>** and `|`. The commands are to be executed as per details given.

1. The **<** token indicates that the STDIN stream should be redirected from a user specified file. Display an error message if this file does not exist or the redirection is done more than once, and discard the entire input line.
2. The **>** token indicates that the STDOUT stream should be redirected to a user specified file. If the file does not already exist, then you should create it; if the file does already exist, then you should destroy its contents and rewrite the file from scratch. Display an error if this token occurs more than once in a command or occurs with **>>**.
3. The **>>** operator indicates that the STDOUT stream be redirected in ***append*** mode. Display an error if this token occurs with **>** or occurs more than once in a command. Again, if the file does not exist you should create it. However, if it exists, you should contents written to the STDOUT stream should be appended to the existing contents.
4. The `|` operator comes in between two separate commands. More details in the next section under piped executions

You may assume that the tokens and files names and commands will be separated by whitespace characters. So, you will not be given an example as "`ps >file.out`". The example will be something like "`./a.out < input.txt > output.txt`".

Other examples include: "`ls > out.txt`", "pwd `>> out.txt`" etc.

You may assume that the tokens **<**, **>**, **>>** will appear at the end of the command after all the arguments and in between

Piped Execution

This mode requires two commands to be supplied. Both of these commands are to be executed concurrently with the STDOUT stream of the first command piped to the STDIN stream of the second command. Assume that this operator will be present only once in a command; so you do **NOT** have to handle multiple pipes like **<command 1>** | **<command 2>** | **<command 3>**. The shell's input and output streams are held by the first and second programs respectively, if no redirection was requested. Assume that *jash* built-in commands will not be present within the pipe commands. Though piped execution commands may be present within batch files given to "run" command, as well as among the commands for sequential or parallel execution. (See combined commands for more details.)

For both IO redirection and piping find out how to use ***dup*** (or its variant ***dup2***) to manipulate input and output streams.

Try these examples to see if your code works: "`ls | sort`", "`man fork | wc`", "`pwd | less`"

Combined Execution (BONUS)

Again if your *jash* program can execute commands compounded together you would get added credit. For combined commands the IO redirection operators **<**, **>**, **>>** gain (*highest*) **precedence** over pipe |, which gains precedence over **:::** used for ***sequential*** and ***parallel*** execution.

Example: "`parallel ls | sort > out.txt ::: ./a.out < in.txt | less`" would be parsed as *(parallel (ls* | *(sort > out.txt)) ::: ((./a.out < in.txt)* | *less))*

Attempt bonus sections only if you've done all previous parts.

Exiting

Again there are no background processes. All child processes forked by *jash* should run in the foreground. The *jash* must wait for foreground process to complete before looping with the **$** prompt. However, the user must be able to kill the current child process by sending it a SIGINT (Ctrl+C) signal. SIGINT should not kill *jash* itself.

The "exit" command or EOF input (Ctrl+D) to the *jash* must just like the previous part exit jash.

Submission Instructions

1. Create a single tar-zipped archive of your folder (which you have to submit) —
tar zcvf <RollNumber1>_<RollNumber2>_lab3.tgz <RollNumber1>_<RollNumber2>_lab3

2. The submission deadline is **6 pm**. Submission on the Moodle link is compulsory. Even if your assignment is incomplete, please submit it for the (6 pm) deadline. We will not accept any late submissions via Email or otherwise.

PART C: The Return of the Shell

30 January 2015

The final part is intended to implement background processes and the cron utility.

Background Processes

The **&** operator indicates the process needs to be run in the background. It will occur at the end of the command (even after the IO redirections). When a command contains the **&** operator, you have to run the command in the background. The I/O redirections should be as per the other operators. *jash* should not wait for the command to finish and should immediately resume its interactive loop and display the **$** prompt.

Whenever a process running in the background ends, the main (parent) *jash* program should display a message along with its PID before the next prompt display. (Somewhat like bash). The message should contain **PID** of the exited process, and **status** it exited in, and if possible the name of the command itself.

Your *jash* program should support the **&** operator along with the IO redirection operators. (This is required and is NOT for bonus credit.)

CRON Utility

Read the man page of ***cron*** if haven't already.

Syntax: *cron <filename>*

CRON command is used to execute commands at predefined time. It takes a file as argument. The file contains the information about when a particular command is to be executed. When it is time for a command to be executed, *JASH* should create a new process in the background and run the command.

Each line in the file contains one job. The job is CRON expression, followed by a command to execute.

```
# *   *  command to execute
# |   |
# |   |
# |   └────────────── hour (0 - 23)
# └──────────────────────────── min (0 - 59)
```

'*' in a particular field denotes that there is no restriction on that field. A value in that field means a restriction is there in that field. For the command to be executed at a particular time, all the fields should satisfy that time instant.

Example 1: 15 * ls
'ls' command will be executed every hour at 15th minute.

Example 2: 5 1 pwd
'pwd' command will be executed at 1:05 AM everyday.

Example 3: * 10 echo Hello
'echo Hello' command will be executed every minute from 10.00 AM to 10.59 AM.

You will get additional credit if instead of polling every minute to check if all restrictions are met, you find how much time is left for the next job and suspend or sleep the process until then.

A CRON process always runs in the background. Adding a **&** operator after it should have no effect on its execution. The CRON process should exit only when the *jash* itself is exited.

Exiting

The *jash* must wait for only the foreground process, and NOT background processes, to complete before looping with the **$** prompt. However, the user must be able to kill the foreground child process by sending it a SIGINT (Ctrl+C) signal. SIGINT should not kill any background process or the *jash* itself.

When the *jash* itself is exited (using "exit" or EOF), it must first kill all background processes running (and also print their PIDs in the format mentioned above) and only then exit. No unnecessary zombie processes must remain hanging. All processes spawned by *jash* must be killed before *jash* exits.

Combined Execution (BONUS)

Again if your *jash* program can execute commands compounded together you would get added credit. For combined commands the IO redirection operators **<**, **>**, **>>** gain (*highest*) **precedence** over pipe |, which gains precedence over **:::** used for ***sequential*** and ***parallel*** execution, and the **&** operator (for background execution) gets the least precedence. So, the **&** operator may occur at most once within a command input line at the end.

Example: "`parallel ls | sort > out.txt ::: ./a.out < in.txt | less &`" would be parsed as *((parallel (ls | (sort > out.txt)) ::: ((./a.out < in.txt) | less)) &)*

Attempt bonus sections only if you've done all previous parts.

Submission Instructions

1. Create a single tar-zipped archive of your folder (which you have to submit) —
tar zcvf <RollNumber1>_<RollNumber2>_lab3.tgz <RollNumber1>_<RollNumber2>_lab3

2. The submission deadline is **midnight**. Submission on the Moodle link is compulsory. Even if your assignment is incomplete, please submit it before the deadline. We will not accept any late submissions via Email or otherwise.

Coding Guidlines and Directives

1. You may code in C or C++ only. You may have multiple source and/or header files. You must adhere to the following **directory structure**:

   <RollNumber1>_<RollNumber2>_lab3
   - [*all source/header files*]
   - Makefile
   - README.txt
   - **test** (optional)

2. **README.txt** file must contain your details (names and roll numbers) , implementation status, additional documentation and anything else you need to tell us. You may optionally add a "**test**" folder containing test cases which worked for your program. This will help us grade.

3. **Makefile** is given to you. It is designed to handle code from multiple source and header files. You may use the given file, or modify it as needed. But a Makefile is needed. Moreover it should have the targets "***all***" (which compiles all source files into the target called '*jash*') and "***clean***" (which clears all temporary, object and executable files).

4. *jash* should handle an **error cases** gracefully by rejecting the line and writing a descriptive error message to the STDERR stream. Try to use the ***errno*** variable ***perror*** function for this. The *jash* program must NOT exit unnecessarily or result in a Segmentation Fault.

5. Your program should contain no **memory leaks**. For every call of ***malloc*** (or ***new***), eventually there should be a call of ***free*** (or ***delete***).

6. Assume that no input line, either from STDIN or a batch file, would contain more than 1000 characters, the newline character included.

7. Apart from error messages and the **$** prompt, make sure your final version does not print any other messages (debugging statements etc.). Keep the output clean.

8. **Documentation is NECESSARY**. Sorry to bother you, but it is. Each function definition should have a comment stub above it to describe what it does. As a healthy programming practice, you should have ample number of comments inside function definitions to guide someone who is reading your code for the first time.

Note that failure to adhere to these directives will result in loss of marks.