Design Document

Assignment 3 – Shell Implementation

Manavjeet Singh, 2018295

# Implementation

**Sample Input:** /bin/ls | /usr/bin/sort | /usr/bin/uniq

## Pseodocode

Create a pipe variable int fd[2] using pipe() function.

**Step 1)** Break the **input string** into Array of strings (\*args[]) with “ “ as the point of partition, Until an “|” is found.

**Step 2)** Add NULL in place of “|” in \*args[] and set the flag found\_pipe to 1.

**Step 3)** Add the leftover string to \*args[] without splitting.

**Step 4)** Check if pipe is found using found\_pipe flag.

**Step 5)** If pipe is not found:

**Step 6)** Fork the current process.

**Child Process**

**Step 7)** Check for redirection formats and implement them appropriately

**Step 8)** Use execvp(args[0], args) to run the input command

**Parent Process**

**Step 9)** Wait for the child process to finish

**Step 10)** Return and ask for more input

**Step 11)** If pipe is found

**Step 12)**  Create a pipe variable int fd2[2] using pipe(). Create a child process using fork();

**Child Process**

**Step 13)** If it is the first segment in a pipe

**Step 14)** Check for redirection input and implement it appropriately

**Step 15)** Set standard output to pipe fd2.

close(fd2[0]);

close(1);

dup(fd2[1]);

close(fd2[1]);

**Step 16)** Use execvp(args[0],args) to execute the current segment in the pipe.

**Step 17)** If it is the last segment of the pipe

**Step 18)** Check for redirection output and implement it appropriately

**Step 19)** Set standard input to pipe fd.

close(fd[1]);

close(0);

dup(fd[0]);

close(fd[0]);

**Step 20)** Use execvp(args[0],args) to execute the current segment in the pipe.

**Step 21)** If it is the neither last segment nor first segment of the pipe

**Step 22) Take input from fd and give standard output to fd2**

close(fd[1]);

close(0);

dup(fd[0]);

close(fd[0]);

*//give output to fd2*

close(fd2[0]);

close(1);

dup(fd2[1]);

close(fd2[1]);

**Parent Process Process**

**Step 23)** close(fd1) so that no one can write to it while child is executing

**Step 24)** Wait for child to execute

**Step 25)** fd[0]=fd2[0], fd[1]=fd2[1]. So that output stream of a process is input stream for the next process.

**Step 26)** set input string to leftover string after the position of current “|”

**Step 27)** if input string NOT EMPTY then iterate again from step 1.

## Test Run

**Step 1)** Fork andExecute “/bin/ls” and put output to fd2.

**Step 2)** Close input to fd. Wait for the child. Set fd = fd2

**Step 3**) Fork and Set fd as standard input, and fd2 as standard output. Execute “/usr/bin/sort”.

**Step 4)** Close input to fd. Wait for the child. Set fd=fd2

**Step 5)** Fork and set input to fd. . Execute “/usr/bin/uniq”.

**Step 6)** Close input to fd. Wait for the child. Set fd = fd2

**Step 7)** Exit