Design Document for Q1

Name: Samina Shiraj Mulani Name: Jithin Kallukalam Sojan

ID: 2018A7PS0314P ID: 2017A7PS0163P

Given problem statement is to build a bash-like shell with a few requirements.

Code Files:

i. shell.c: Execute gcc shell.c

Assumptions:

i. No combination of I, II, III pipe operators can exist in a single command.
ii. Multiple I pipe operators can be present in a command, but only one II and ||| pipe operator can be present in a command.
iii. File redirection operators(<,>,>>) can be present anywhere in the command, but should ideally be placed only at the start and end of the command. For example: **ls –l > file | grep -i r** is valid, but the input to grep -i r is empty.
iv. The maximum size of any command given to the shell is 100 characters(Can be altered using the MAX macro).
v. The maximum number of processes that can pipelined is 10(can be altered using the MAXProcesses macro).
vi. Paths for the commands given are searched for in the PATH environment variable.

The design of the shell can be mainly divided into the following parts:

i. Parser: This module is used to parse the command given to the shell. The parser is flexible, accounts for varying number of spaces, tabs, commands interleaved with file redirection operators, multiple redirection operators, etc. It also checks the validity of the commands based on the type of pipe operators, number of commas, etc. The main functions involved in this part are:
    a. parseCommand()

b. checkCommandPipelineType()
c. checkValidity()

ii. Execution: The parent goes through each path and searches for an executable with the given name using the access() call. It also handles the ordering and execution of the processes that are part of the command. The parent also prints the pids and statuses of each of the processes, the pipe fds/file fds used and the order of execution.
    a. In the case of a single pipe(|), the output of the process to the left is pipelined into the input of the process to the right.
    b. In the case of multi-pipe operators(||,|||), the output of the process on the left is taken from a pipe into a buffer at the parent and then written into pipes as the input to each of the process on the right.

    The main functions involved are:

    a. searchForPathAndExecute()
    b. executeProcess()

iii. SIGINT: On receiving the SIGINT signal, the parent prints the last 10(or less) commands and the pids and statuses of the processes in each of those commands if the commands were valid, else shows that the commands were invalid. The signal handler is:
    a. sigHandlerINT()

iv. SIGQUIT: On receiving the SIGQUIT signal, the parent asks if the user wants to exit and takes action accordingly. The signal handler is:
    a. sigHandlerQUIT()