**1. Introduction to Architecture:**

Usually a shell is divided into four levels viz process, shell, command line, user.

**User level:** Deals with input and output of commands of user. There is no exception handling at this level

**Command Interpreter service :** The input from user is processed and examined to be presented to the next level. This is the layer where exception handling can take place. In case of an error or deviation of regular flow of command, exception handler will be called. If there are no errors, then program will continue with its normal flow.

**Shell Service:** This level is filled with data structures, algorithms, input commands, existence of commands, action to be taken next etc. If commands do not exist or further action is undecidable, the exception handler is called. This is where out tab function going to work. It shall find commands and fill it with link table.

**Process Service Level:** It is this level where its decided whether fork shall handle the command or pipe. Its where data from shell is processed and responded back to user.

User Service

Command Interpreter Service

EXCEPTION HANDLING

Shell Service

Process Service

**2. Function Descriptions**

Our shell consists of the following functions and description for each of them is given below

a) Function Name : set\_profile,set  
Using this file the environment variable are to our shell. This can be tested by giving the built in commands such as ls, ps, etc. also the home directory is set as needed by the ourselves. This can be verified by giving pwd as it will point to the required file. Used setenv command to set the value.

b) Function Name : cmd\_exec  
Description : This function accepts pointer to array of strings and returns the status or error code. It begins by checking if the input string has been previously stored or not. It shall fork to parent and child. Child calls execvp command through which new process image replaces current one. The return value for this is only to check whether an error has occurred or not and is directly related to errno. In this function, our requirement of alarm after 5 seconds has been set. When executed under circumstances, a prompt shall be received by user asking whether he/she wants to terminate program or not.

c) Function Name : parsein  
Description : This is a preliminary check function which is used to parse all the input arguments. First step of this is to check if the input string is ‘echo’ and if it is then increment position integer. For this if statement the else if checks for /r. If none of above condition is satisfied then it calls cmd\_exec function  
Exceptions Handled : For input commands echo and /r

d) Function Name: get\_line ,split\_line  
Description : This function is used to get the input from the user and split it and send it to the parser for processing

e) Function Name : History  
Desctiption : This function is repeatedly checked when user is giving any input. Algorithm goes as follows: Check the command string given by user, when pressed tab this function is invokes and command table checks if this command was previously given by user. If given by user it gets command form table else no action is taken. If a completely new command is entered, an entry is made into the command table for future reference.  
Exceptions Handled : If no entry found then function should not return any value

**3.Exception Handling:**

Exception is something that disrupts normal flow of program. Exception handling has a broad classification and plays crucial role in determining the robustness of system. An ideal system should be able to handle all possible exceptions. In our program we had written exception handling function at most of the places to produce a reliable output.

# Test case Description

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Description | Expected Outcome | Actual Outcome |
| 1 | cat .profile | Execute shell command | Success |
| 2 | cd to any forwarding directory | Change current dir to the given one | Success |
| 3 | echo $HOME | Display home directory | Success |
| 4 | prompt | => | Success |
| 5 | Press the enter key i.e. null input shell | Handled well created the line and did not throw any error | Success |
| 6 | alarm = 5; | Set alarm to 5s. Override the alarm timer from the profile file | Success |
| 7 | sleep 10 | Alarm triggered. User prompted if wish to exit command | Partial Success. For response = no, the parent process does not wait for the child |
| 8 | alarm = 0;  sleep 5 | No alarm generated | Success |
| 9 | History feature implementation | History buffer was created | Partial success using h we can display the commands used |
| 10 | Invalid command execution | Throwed error | Success |