Design Document for Project1(CS\_551) Developers: Imran Ali, Goutham Kannan, Cheithan Bapatla

**Flow Control Diagram**

Run the executable to launch a new shell

Running the .profile file to export the PATH variable and to set Timer

Display the prompt to get user inputs

If input is #Alias \*

Yes

Save the Alias command in a local file

No

If the input of format If..then..else..fi

No

Parse the statement and execute the condition and action accordingly.

Yes

No

Decode the Alias to Minix comand

If the command is an alias

Yes

No

Execute the Command

If command Blocks

Initialize Block Timer

No  
   
 Yes

Continue Execution

Block Execution

Discontinue…?

**Program structure:**

To build our own shell, the following files/functionalities have been created:

1. .profile file:

The file will be used to set the initial parameters of the shell, like Timer value or the PATH variable. It will read at the beginning when the shell invokes, before displaying the command prompt.

1. main C file

In our case this file is main.c . This file contains the source code of the working of the shell and all of its functionalities. It consists of the following functions :

* 1. get\_timer():

This function reads the .profile file and finds the value of the “TIMER” variable and returns it.

* 1. read\_cmd\_line() :

This function reads the command from the command line and returns it.

* 1. split\_cmd\_line() :

This function splits the command and the command line arguments

* 1. run\_exec() :

This function executes the command using execvp() function, if the command blocks, run the timer and ask the user if it wants to terminate the blocking command. If the commands doesn’t block, return to the command prompt.

1. Alias file:
   1. User requested Aliases are stored under the file Alias\_list.txt , This file contains the Key – Value combination.
   2. Eg. Alias list=’ls’ command gets stored in the file as
      1. list:ls

So many new Alias requests gets appended to the next line and so on.

1. my\_shell file :

This is a shell script which is used to initialize all the variables written in the .profile file, except the TIMER variable which is initialized by the get\_timer() function.

1. test.c file

This file contains an infinite loop and it never returns. This file is for the testing purpose of the shell. It blocks when it is compiled and run on shell.

**Exception handling mechanisms:**

1. If a process doesn’t fork: Collect return value of the fork function, display the error message accordingly, ask the user to retry and return to the command prompt.

2. If the execvp() returns unsuccessfully: Collect return value of the execvp function, if it appears less than zero, retry exec again else return to the command prompt.

3. If a process blocks later than expected: Initialize the timer, if the execution time exceeds, ask the user to terminate the process. If user agrees terminate the process, collect the core dump.

4. In case of If-then-else-fi parsing each of the commands are tightly matched , incase of mismatch appropriate error messages are provided.

**Test Plan and Test Cases :**

The following areas have been tested.

1. .profile file : The file should be read and initialized properly by the shell:
   1. TIMER : Change the TIMER values in the .profile file, see if it works as expected.
      1. If the TIMER=0 the timer should switch off.
      2. If timer is on, run ‘./test’ , wait for timer’s prompt”[Y/N]”, and say ‘Y’. ‘test’ should terminate generating a core dump.
   2. PATH : Change the PATH value to “/bin” , then run a commend from “/sbin” like “chown”, it should produce an error.
2. Alias command : Run the following commands on shell prompt :
   1. >Alias ls1=’ls’

>ls1

It should work similar to ‘ls’.

* 1. >Alias ls1=”cat”

>cat main.c

The ‘cat’ should run, not the ‘ls’.

* 1. The Aliasing is made work even after restarting the shell , The Alias names are stored under Alias\_list.txt and retrieved as when required. Alias cease to exist till the Alias\_list.txt file is deleted.

1. Special Alias case >Alias **ls=”ls1”**

It should print an error saying ‘ls’ is an existing command, it cannot be aliased. As ls is an existing binary command, it can’t be used an Alias name for another command.

eg. If the below mentioned Alias statements are executed , it will cause led to improper functioning of ‘ls’ , So such cases are avoided.

Alias ls1=’pwd’

Alias ls=’ls1’

1. The if-then-else-fi command :
   1. Any input of the format If [string1]; then [string2]; else [string3]; will be parsed;
      1. String1 will be executed as the If condition statement.
      2. If the condition statement [string1] passes then [string2] will be executed.
      3. If the condition statement [string1] fails then [string3] will be executed.
   2. Eg: If ls; then pwd else date; fi
      1. If command ls got executed properly then pwd command will be executed.
      2. If ‘ls’ fails then date command gets executed.
   3. If-then-else-fi command are not implemented to evaluate expressions, this command is designed for shell usage.
2. Basic Shell testing : Run the following commands on shell and see if you get expected results:

>ls

>cat main.c

>echo “hello”

>exit -> This addition shell command is supported to quit the current execution.

1. Negative Testing : See if the shell detects the erroneous commands :
   1. >abcd

Error message should appear.

* 1. In ‘.profile’ file, type TIMER=abc

The error message should appear saying wrong value for TIMER, setting to default.

**New Shell functionality**

Running in background using ‘&’ : Run the following commands on shell :

>./test &

The prompt should appear again.

>./ps ax

You should see “./test” process

>./kill -9 <pid of ./test>

>./ps ax

The “./test” process should disappear from process list.