CS 341 Assignment

Our Team Members

Hareesh Reddi - 150101051
M. Sai Krishna Prateek - 150123025
Nikhil Matcha - 150123021

Our command shell supports all the basic commands such as

1. pwd - Gives present working directory.

2. cd <dirname> - Changes directory name if possible.

3. ls - List all the Files/Directories in the current directory.

4. rm commands

(a) rm -r <filenames/Directorynames> Removes all the filenames and Directory names mentioned if they are present in the current directory.
(b) rm <filenames> / rm -f <filenames> Removes all the filenames mentioned if they are present in the current directory (does not remove directories).
(c) rm -v <filenames> Removes all the filenames mentioned if they are present in the current directory(does not remove directories) and gives details of all the files removed and no. of files removed.

5. history n (most recent n commands) / history (all commands)

6. exit - Exits from Command Shell.

7. issue n - Issues the nth command in the history once again.

Also supports features like

8. rmexcept <filenames> - Removes all the files except the filenames mentioned in the current directory(does not remove directories) and gives details of all the files removed and number of files removed.

9. <program_name> - Creates a child process to run <program_name>. Supports > and < to redirect the input and output of the program to indicated files.

Description of the Code :

(1) We assumed maximum length of a command can be of 256 characters(MAX_LINE), count denotes total number of commands executed until now and maximum no of arguments in a command can be 129 (MAX_LINE/2 + 1).

(2) main(void) function initializes inputBuffer to store the current command and args[] array for parsing the current command into arguments. It starts a child process which executes the commands the user enters in the prompt and we assumed parent process waits for child process to complete.

(3) formatCommand(char inputBuffer[],char* args[]) function uses size_t read(int files, void *buf, size_t nbytes) function to take input command and buf stores the input command. It also creates another buffer iB to store the current command and will be parsed using parse function and will be executed through execute function.

(4) To execute history command, we used a buffer history[1000000][MAX_LINE] to store all the commands(maximum 1000000 commands can be executed) and Commands will be checked in the function execute(char* args[],char* iB) where iB denotes the input buffer and args denotes the arguments of the current command which are updated for each command using parse(char *line, char *args[],int length) function where parse functions takes input as line(current command) and length(length of the line) and stores the arguments into args[] char array.

(5) void displayHistory(int num) function is used for displaying the history according to number of recent commands needed i.e num(by default prints all commands issued by the user.) and numbers are assigned based on recency of the command (EX: if (history 3) commands outputs (1.ls 2.pwd 3.cd Desktop); then ls is assumed to be most recent command and assigned number 1)

(6) In the execute(char* args[],char* iB) function,buffer history is always updated and count is increased by 1 for each command entered by the user. Each command is checked using strcmp(str1,str2) function for execution using if--else if--else block.

Some of the important Library functions used are as follows –

(a) For executing cd command ::

chdir(directory)---Library/Header file(unistd.h)
char* getcwd(char* buffer,size_t size)---Library/Header file(unistd.h)

(b) For executing all types of rm commands ::

First,file is opened using DIR* opendir(const char *name) function, then read using struct dirent readdir(DIR* dir) function and removed files/directories according to the given command using int remove(const char *filename) function and finally closed the file using int closedir(DIR* dir) function.Libraries/Header files (dirent.h,sys/types.h) are included for using the builtin library functions.

(c) For Creating a child process to run <program_name> and supporting the redirection operators > and < to redirect the input and ouput of the program to indicated files,example command::(a.out > in.txt < out.txt) a.out is executed and in.txt is given as input and output is written to out.txt.This is done using inbuilt functions such as dup2(),open(),close() and used atoi() and numbers_only() function for command <program_name> m to check for number m and and alarm(timer) and SIGALRM are used to abort the process if it does not complete its operation in m seconds.

Contributions:

1)M S Krishna Prateek – Took care of basic commands like pwd ,exit and helped in debugging.

2)Nikhil Matcha – Implemented basic functions to support execution.

3)Hareesh Reddi – Took care of entire code and lead the team to complete the exercise and implemented all the important functions and referred linux book for implementing each of the commands.