Robert Bethune
Evan Schipellite
CSI 385 - 02
1 February 2015

FAT Design

Shell Structure

Description:

Runs in a loop, waiting for user input. When the user logs out or types an exit key, the loop will end and the shell will terminate. Anything that the user types will then be processed within the shell. The shell will attempt to validate the command with any of its listed commands, allowing for a child process of that command to be created if valid. Otherwise, the shell will either return a statement noting an incorrect command or it will give a prompt for the proper amount of parameters for the desired command.

Notable Functions & Parameters:

char* m_Command;

char* m_Parameters;

void run();

Attempts to choose a valid command that compares with the current m_Command. This will then pass the parameters into the command, creating a child process that will attempt to execute the command. On failure, the command will generate an appropriate error output. If the command doesn't exist, an error will be output with the command information. Resets character arrays to empty at the end of the function.

void parse(char* userInput);

Parse user input, separating input into its command and parameter structure. This sets m_Command and m_Parameters. M_Paremeters may remain empty if the user does not provide parameters.

Testing:

- ➢ Handle all forms of user input
    - o No input
    - o Too much input for parameters
    - o Wrong command

cat X

<u>Description:</u>

Prints out the current file information to the screen. The user can specify a file path, allowing them to designate whether the file is within a relative or absolute file path.

<u>Notable Functions & Parameters:</u>

char* m_File;

char* m_FileContents;

char* m_CurrentDirectory;

void readFile(char* filePath);

Utilizes filePath to open up the file. This will allow a loop through all of its contents, copying the character information to the m_FileContents.

Void printFile();

This outputs the m_FileContents to the screen, attempting to keep formatting.

<u>Testing:</u>

- Error message
    - If directory
    - If more than one argument
        - Could also just utilize first argument
    - If file does not exist
    - If file cannot be accessed or read properly

cd X

Description:

Changes the current directory to the designated directory. If no argument is available, places the user at the home directory.

Notable Functions & Parameters:

char* m_Directory;

void setDirectory();

Set the m_CurrentDirectory in the shell, if applicable. Previous checks determine if the directory provided is valid, therefore allowing the shell to now acknowledge the current directory to apply additional commands within.

Testing:

- Error message
  - If directory is invalid
  - If more than one parameter is provided

df

<u>Description:</u> Prints out the free space on the current disk mounted. If the user provides any parameters, these are ignored.

<u>Notable Functions & Parameters:</u>

int m_DiskSpace;

void checkDiskSpace();

Checks the current mounted disk to uncover the free logical blocks. This could utilize functionality from the pbs command, allowing it to access cluster and sector information. Each time this function is called, it will reexamine the mounted disk to ensure it is examining the current disk, and to allow for allocation changes to be noted.

<u>Testing:</u>

- Outputs an error message if no disk is mounted

ls (X)

<u>Description:</u>

Prints out the current directory if no arguments are listed, and it prints out the directory of the argument or file with extension.

<u>Notable Functions & Parameters:</u>

char* m_TargetChar

void ReadCurrentDirectory():

this function if no arguments are sent in will go through the current directory and write to the target char pointer.

void PrintChar():

This will take the current directory char and print it out so the user will see from the targetChar

void FindFile(char* argument):

This will go through the current directory and only give files that contain the argument array in full. This will be saved to the targetchar and then be ready for print function.

<u>Testing:</u>

- Checks if the argument given exists at all

mkdir X

Description:

This will create a new directory with the X name in the current working directory or path if one is given.

Notable Functions & Parameters:

int DoesExist()

This will take in the X and check if that directory already exists. If it does it will send out a 0 if it doesn't exist and a 1 if it does exist.

int CreateDirectory()

This will create the directory for the user and output again a 0 or 1 depending on if it worked or failed. It will also send out an appropriate error message if it fails for a reason. Example char name is way too large to create the directory.

Testing:

- Test if the directories are being saved to disk and not just written to current instance than lost

pwd

Description: Prints out the absolute path of the current directory.

Notable Functions & Parameters:

printAbsoluteDirectory();

Acquires the current directory from the Shell and outputs it to the screen. If the m_CurrentDirectory is empty, this means the user is in the root directory.

Testing:

rm X

Description: Attempts to remove the specified file, if it exists. Can have an absolute or relative path.

Notable Functions & Parameters:

char* m_File

removeFile();

Executed after error checking. Removes the file from the disk. This removes it from the parent directory and adjusts allocations within the disk. This will also free clusters, possibly updating the df command.

Testing:

- Print an error message
    - If the file does not exist
    - If the file is a directory
    - If there is more than one argument

rmdir X

Description: Removes the provided directory from the disk. Can have an absolute or relative path.

Notable Functions & Parameters:

char* m_File;

int checkIsEmpty();

After error checking, this checks to see if the existing directory has any files. If so, this returns 1 (true) and continues with the command. Otherwise, this returns 0 (false) and exits.

void removeDirectory();

This removes the directory from the parent directory, adjusting space allocations. This also frees up data clusters, potentially updating other commands, such as df.

Testing:

- Prints an error message
    - If the directory does not exist
    - If the arguments is more than one
    - If the argument is a file
    - If the directory is not empty

touch X

Description:

This function will look at X and see whether or not it exists already, if it does it will return that it does. If it does not exist it will look and try to create the new file or directory in either the name or file path location

Notable Functions & Parameters:

int Search()

this will search the current directory or filepath to see if the file can be created or not. This will return an int which will determine whether or not it moves forward.

void Create(char* filePath)

this function will change current working directory and save the old location or steps to get back. Than it will create the new file in the location and step back out to the current directory the user was on.

Testing:

- Make sure it was actually created and saved properly.