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:

#define INPUT_LENGTH

Provides a base string length for all command processing.

char* m_Command;

char* m_Parameters;

void main();

This prompts the user for their desired command. If exit is entered, the application quits. Otherwise, if the first part of their input line is recognized as a command, the arguments are acquired and the appropriate command is executed alongside the arguments.

void GetArguments(char* commandName, char* commandLIne, char** arguments);

Parses user input following their accepted command. This allows for the creation of a commandInfo string, which contains all required arguments for the execution of commands. The initial string in the character array is also set to the command name, allowing it to be called following the argument acquisition. This utilizes strtok to parse the command line into the character array.

Testing:

- User Input
    - If not a known command
        - Returns to prompt
    - If ‘exit’ is entered
        - Quits the shell
    - If known command
        - Execute command
            - Command will handle error checking on parameters

cat X

Description:

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.

Notable Functions & Parameters:

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.

Testing:

- On execution
    - If no parameters
        - Error Message: No file provided
    - If parameters
        - Output error message if
            - X is directory
                - Error Message: Invalid file provided
            - More than one argument
                - Error Message: Multiple arguments entered

- File does not exist
    - Error Message: File does not exist
- File cannot be accessed / read
    - Error Message: File cannot be read

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:

- If no argument provided
  - Move user to home directory
- If parameters
  - Error message if
    - Directory is invalid
      - Error Message: No directory found
    - X is file
      - Error Message: Argument is not a directory
    - More than one parameter is provided
      - Error Message: Multiple arguments entered

df

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

Notable Functions & Parameters:

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.

Testing:

- Outputs an error message if no disk is mounted
    - Error Message: No disk mounted

ls (X)

Description:

If X is a file, this should list out the file name, extension, type FLC, and size. If X is a directory, this will list out the names of all files within the directory, also detailing file information. If X is empty, this will print out all file information for the current working directory.

Notable Functions & Parameters:

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.

Testing:

- If no parameters
  - Prints out current directory contents
- If parameters
  - If file
    - Prints out file information
    - If not a valid path to file
      - Error Message: Invalid file entered
  - If directory
    - Checks to see absolute / relative path

- If incorrect path to directory
    - Error Message: Invalid path

mkdir X

Description:

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

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:

- If parameter provided
    - If X exists
        - Error Message: Directory or File already exists
    - Else
        - Check for relative / absolute path
            - Create directory at path
- If no parameters
    - Error Message: Enter the path to the directory

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:

- If no parameters
    - Error Message: No file specified
- If multiple parameters
    - Error Message: Multiple arguments provided
- If one parameters
    - If file does not exist
        - Error Message: File does not exist
    - If argument is a directory
        - Error Message: Argument is a directory

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:

- If no parameters
  - Error Message: No directory specified
- If multiple parameters
  - Error Message: Multiple arguments provided
- If one parameters
  - If directory does not exist
    - Error Message: Directory does not exist
  - If argument is a file
    - Error Message: Argument is a directory
  - If directory is not empty
    - Error Message: 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:

- If one parameter
    - If X exists
        - Error Message: File already exists
    - If X does not exist
        - Check relative / absolute path
            - Create file X at path
- If no parameters
    - Error Message: No file specified
- If multiple parameters

- Error Message: Multiple arguments provided