FAT System Design

Robin Shafto, Alex Taxiera

October 4, 2016

1 Shell Description

The shell will read and parse user input, if "exit" was input it exits, otherwise it executes the command given and loops for more input.

2 New Functions

2.1 mapFree

Prototype: char* mapFree()
Input: None
Returns: A free-space bitmap

This function creates a free-space bitmap to track the free space.

2.2 updateBitmap

Prototype: bool updateBitmap(char* operation, int bits)
Input: operation is whether space was added or removed, and bits are how many bits were altered.
Returns: True if successful

This function will update the bitmap with what space is free.

2.3 getCurrentDirectoryContents

Prototype: string getCurrentDirectoryContents()
Input: None
Returns: An array of strings containing all files and directories in current directory.

This function will check all the contents of the current directory, providing a reference for commands like cd, cat, and touch.

2.4 getDirectoryContents

Prototype: string getDirectoryContents(char* directory)
Input: Directory to read
Returns: An array of strings containing all files and directories in directory

This function will check all the contents of the given directory, it will call isDirectory first.

2.5 fileExists

Prototype: bool fileExists(char* fileName)
Input: File that you wish to check.
Returns: True if the file exists.

This function checks the existence of a file by calling getCurrentDirectoryContents and comparing.

2.6 isDirectory

Prototype: bool isDirectory(char* fileName)
Input: File that you wish to check.
Returns: True if the file is a directory.

This function checks the type of a file by calling fileExists.

2.7 printFile

Prototype: void printFile(string fileName)
Input: Name of file to print contents
Returns: Nothing

This function will print the contents of a file.

2.8 error

Prototype: void error(int errorNumber)
Input: Number of error to print.
Returns: Nothing

This function will be sent an integer value when there should be an error thrown and will then print the corresponding string.

2.9 getFileSize

Prototype: int getFileSize(char *filename)
Input: Name of file to get size.
Returns: Size of file.

This function will find the size of a file.

2.10 checkRange

Prototype: bool checkRange(int x, int y)
Input: Range of FAT entries to check.
Returns: True if valid inputs, otherwise false.

This function verifies range when looking to read FAT entries.

2.11 readFAT12Table

Prototype: char* readFAT12Table(char* buffer)
Input: Which FAT table to read.
Returns: A buffer with the desire FAT table.

This function reads a FAT table for us.

2.12 freeSector

Prototype: bool freeSector(int sectorNumber)
Input: The data sector of the target file.
Returns: True if it was successful.

This function frees sectors.

2.13 findFree

Prototype: int findFree(char* directory)
Input: Find free entries in the directory
Returns: The free entries

This function finds free entries.

2.14 allocateSector

Prototype: bool allocateSector(char* directory)
Input: Allocate another sector to the directory
Returns: True if successful

This function allocates more sectors for file creation.

3 Command Processes

3.1 cat *x*

1. Call isDirectory
2. If the file exists and it is not a directory, call printFile
3. Else return an error

3.2 cd *x*

1. Call isDirectory
2. If the directory exists then move to it
3. Else return an error

3.3 df

1. Print the free-space bitmap.

3.4 ls *x*

1. If more than one argument return error
2. If no arguments call getCurrentDirectoryContents
3. Print directory contents
4. If one argument call isDirectory
5. If true call getDirectoryContents
6. Print directory contents
7. If false call fileExists
8. If true display file details
9. If false return error

3.5 mkdir *x*

1. If more than one argument return error
2. Call isDirectory
3. If true return error
4. If false call findFree
5. If no free entries call allocateSector
6. If false return error
7. If true create the directory and call updateBitmap

3.6 pwd

1. Print absolute path to the current directory

3.7 rm *x*

1. If more than one argument return error
2. Call fileExists
3. If false return error
4. Call isDirectory
5. If true return an error
6. if false remove the file
7. Call freeSector and updateBitmap

3.8 rmdir *x*

1. If more than one argument return error
2. Call fileExists
3. If false return error
4. Call isDirectory
5. If false return error
6. If true call getDirectoryContents
7. If array is not empty return error
8. If array is empty then remove entry
9. Call freeSector and updateBitmap

3.9 touch *x*

1. Command should ignore all arguments except first one
2. Call fileExists
3. If true return error
4. If false call findFree
5. If no free entries call allocateSector
6. If false return error
7. If true create the file and call updateBitmap

4 Test Plans

The shell must be completed first so that the other components may be tested. It will be tested for compatibility with the provided functions, then with stubs of the commands, then with the supplementary functions and working commands.

4.1 Test Cases

4.1.1 cat *x*

1. **For a file:** prints file contents
2. **For a directory:** prints error message
3. **For incorrect number of arguments:** prints error message
4. **For unspecified:** prints error message

4.1.2 cd *x*

1. **For a specified directory:** changes to specified directory
2. **For unspecified:** changes to home directory

4.1.3 df

1. **For all inputs:** creates free-space bitmap mapping each cluster to a bit

4.1.4 ls *x*

1. **For a filename:** lists name, extension, type, FLC, and size of file
2. **For directory:** lists names of entries with their extensions, types, FLC, and sizes as well as the current and parent directories
3. **For unspecified:** for the current directory, lists names of entries with their extensions, types, FLC, and sizes as well as the current and parent directories
4. **For two or more arguments:** returns error message

4.1.5 mkdir *x*

1. **For existing directory:** prints that x already exists
2. **For insufficient memory:** print appropriate message
3. **For a number of arguments not equal to one:** returns error message
4. **For a new directory x in an existing target directory with space:** creates x in target directory

4.1.6 pwd

1. **For all cases:** prints path to current directory

4.1.7 rm *x*

1. **For nonexistent file:** prints that file does not exist
2. **For a directory:** print error message
3. **For a number of arguments not equal to one:** prints error message
4. **For a file x in a directory:** removes x from parent directory, optimizes parent directory storage, frees data sectors of target file, updates data structures

4.1.8 rmdir *x*

1. **For nonexistent directory:** prints that file does not exist
2. **For a file:** print error message
3. **For a number of arguments not equal to one:** prints error message
4. **For non-empty directory:** prints error message
5. **For a directory x:** removes x from parent directory, optimizes parent directory storage, frees data sectors of target file, updates data structures

4.1.9 touch *x*

1. **For existing file x:** prints that x already exists
2. **For insufficient memory:** print appropriate message
3. **For a number of arguments not equal to one:** returns error message
4. **For a new file x, given sufficient space:** creates x in target directory