**CSCI430 File System Specification: J-SpaceMaster**

**Joshua J. Evers**

**INTRODUCTION:**

The usage of this file system is to maintain as much usable space, removing as much slack space as possible. The system will be slow, deletes and updates will be very slow, whenever a file is updated/deleted it will need to be moved. The only required space for usage is the set space for file details. The files are never separated, if there is space a file is always added to the end of the disk. When a file is deleted that is not at the end, all files move left if possible. This means disk defragmentation will have no effect on this file system.

Start of File system contains

<VolumeName> String Cannot be longer than 8 characters

<TotalSpace> long 8 bytes

<FilesOnDisk> int 4 bytes

20 bytes total initial data

**Volume Methods**

**ALLOCATE**

**Allocate<VolumeName> <Size>**

Allocates a file of the specified space and given name. This method writes the first data to the disk containing the volume name, size of the disk, and running total of file on the disk.

**DEALLOCATE**

**Deallocate<VolumeName>**

This method deletes the volume that is passed into it

**TRUNCATE**

**Truncate<VolumeName>**

Takes in the volume name and writes \0 to all spaces after the initial data set.

**DUMP**

**Dump<VolumeName>**

Writes the volume to the console one byte at a time.

**MOUNT**

**Mount<VolumeName>**

Opens the volume to be used for file methods

**UNMOUNT**

**Unmount<VolumeName>**

Closes the volume that is mounted or returns volume not mounted.

**Vinfo**

**Vinfo<>**

Volume information method. Prints the volume name, size and total files to the screen.

**FILE METHODS**

**FileInfo**

**FileInfo<FileName>**

File information method. Prints the file name, size and properties to the screen.

**createFile**

**CreateFile<FileNAME>Creates a new empty file on the volume.**

1. Checks for space requirement of file of the total drive
2. Reads the number that tells the total files on disk
3. Gets size of the last file
4. Writes filename and properties

Stores filename with the attributes following it directly

Such:

<FileName> - String 16 bytes

<Size> - int – 4 bytes

<ReadOnly> - boolean – 1 byte

<DateCreated> - short - 8 bytes ddmmyyyy

<TimeCreated> -int - 4 bytes hhmm

<DateLastModified> - short – 8 bytes ddmmyyyy

<TimeLastModified> - int – 4 bytes hhmm

45

**volSize**

**volSize<>**

Retrieves the size of the volume from the datasets at the start of the disk

**totalFiles**

**totalFiles<>**

Returns the value of the totalFiles value from the start of the disk

**Search**

**Search<FileName>**

Returns true or false if it finds the file given within the currently mounted volume.

**fileSize**

**fileSize<FileName>**

searches the mounted volume for the file, if found scans to the files size value and returns.

**fStart**

**fStart<FileName>**

This adds up all of the sizes of the files before the file that is specified. Then it returns the total size file found.

**spaceRemain**

**spaceRemain<>**

goes through the size values of all files on the volume and returns their total size

**fileWrite**

**fileWrite<FileName><Data>**

This method takes in the name of the file you would like to append data to, and the data to be appended.

**fileDelete**

**Delete<FILENAME>**

Delete moves all files to the right of the deleted file left and overwrites the file, or if no files after the initial file by removing the name and start location of the file it will be overwritten when that space becomes used.

1. Checks for file name

2. if found goes to data and begins moving files left

3. as it moves, calls write command to move and update files

**fileLoc**

**fileLoc<FileName>**

Given a filename returns that file’s starting location

**filesRight**

**filesRight<FileName>**

Finds the total number of files to the right of the file given. Used for the filemigration and fileWrite commands to know how many files to move so that files are not overwritten.

**Filemigrate**

**Filemigrate<FileName><end>**

Given a filename there are three paths through migration. The first (end == 0) moves all files right of the FileName to the end of the disk. The second (end == 1) moves all files at the end of the disk to the right up to the FileName given’s total size. The third (end == 2) is used for deletion and overwrites any data by moving all files right of the FileName’s location overtop of itself.

**fileRead**

**fileRead<FileName>**

Given the FileName searches the disk for the file, and data location, then prints it to the screen