FAT12 Project
Testing Report
Team 6

2/2/05

Assume ls and util.c functions to be already tested and working.

Test cases for util.c

parse_path

1)Called with a path that does exist
Returns the correct FLC and index of the path

2)Called with a path that does not exist
Returns an negative integer error code

3)Called with ".", "..", and /
If called with ".", returns the FLC and index for the current directory
If called with "..", returns the FLC and index for parent directory
If called with / returns the FLC of the root directory

get_file_info

1)Called with an existing FLC/index pair
Returns the data stored at the location of the FLC/index pair

get_file_type

1)Called with a fileinfo struct that is for a file
Returns a 1

2)Called with a fileinfo struct that is for a subdirectory
Returns a 2

3) Called with a non-initialized or otherwise invalid fileinfo struct
Returns a negative error code

get_dir_list

1)Called with a directory FLC that begins and ends in the same cluster
Correctly reads up to the 15 possible directory entries

2)Called with a directory FLC that requires a read from the FAT table to read completely
Correctly reads all directory entries across all FAT entries used by the directory

3)Called with an FLC that does not point to a valid directory entry
Returns a negative error value

4)Called with a value for max that will hold all the directory listings returned
Returns all directory listings in the *list* data space. The integer return value indicates how many values were returned, and will be less than or equal to the length of the *max*.

5)Called with a value for max that will not hold all the directory listings returned
Returns *max* number of directory entries in *list*. The integer return value indicates how many entries were actually present in the directory.

set_wd

1)Called with a path and an FLC
Correctly sets the path and FLC in shared memory space

get_wd

1)Called with buffer to hold the path
Correctly returns the path and FLC stored in shared memory space

get_free_space

1) Called when some of the blocks are used
Correctly returns the number of blocks used

2)Called when no blocks are used
Returns all blocks as being free

3)Called when all blocks are used
Returns all blocks as being full

drive_usage

1)Called with a filestructure reads size from fileinfo struct
Correctly reads the filesize field from the fileinfo structure and returns the value

get_free_block

1)Called when there are blocks free
Correctly returns the first free block available

2)Called when the device is full
Returns a negative error value indicating the device is full

allocate_block

1)Called with a root_FLC that has a free index to place the fileinfo in
Places fileinfo structure into the root_FLC indicated and initializes FAT entry. Returns root_FLC in *FLC* and the index where the entry was placed.

2)Called with a root_FLC that requires a FAT jump to another block to store fileinfo
Follows FAT table to jump to block with free index entry, places structure into directory, and returns the FLC of the directory entry in *FLC* and the index in *index*.

3)Called with a root_FLC that requires a new block be created to store the fileinfo
Allocates and links a new block for a new directory entry, follows the FAT table to that block, places the structure there, and returns the FLC and index where the filestructure was placed.

write_data (optional function, not tested now)

unlink_entry

1)Called with a directory with files in it
Returns a negative error code.

2)Called with a directory with only “.” and “..” in it
Removes directory entry in FAT table (assuming removing one/multiple blocks works correctly) and removes entry from parent directory.

3)Called with a directory that is contained in one block
Removes the directory from the FAT table and removes entry from parent directory.

4)Called with a directory that is contained in multiple blocks
Follows the FAT table removing each block that is associated with the directory.

5)Called with a file that is contained in one block
Removes the entry for the block from the FAT table.

6)Called with a file that is contained in multiple blocks
Follows the FAT table removing all blocks associated with the file.

read_file

1)Called with a file that spans only one block
Output up until the end of the block is returned

2)Called with a file that spans multiple blocks
Output from all blocks is returned

check_filename

1)Called with a legal FAT12 filename
Returns a success code

2)Called with an invalid FAT12 filename
Returns a negative error code

Other test cases (not util.c)

mkdir

Given the ls output below used for each case:

```
floppy> ls
Name           Type      File size      FLC
.              Dir       512            10
..             Dir       512            2
Dir1           Dir       512            21
help.txt       File      35             22
```

Case 1: Argument already exists (directory or file)

```
floppy> mkdir Dir1
Specified file/directory already exists.
```

Case 2: Argument is greater than one

```
floppy> mkdir Dir1 help.txt
Please list only one directory.
```

Case 3: There are no arguments

```
floppy> mkdir
Please list a directory to create.
```

Case 4: Argument is a non-existent directory to be placed in the current directory

```
floppy> mkdir Dir2

floppy> ls
Name           Type      File size      FLC
.              Dir       512            10
..             Dir       512            2
Dir1           Dir       512            21
help.txt       File      35             22
Dir2           Dir       512            23
```

Case 5: Argument is a non-existent directory not in the current directory

(give an absolute path as well as parent directory to check here), should get same results as case 4 only in different directory)

pwd

Case 1: There are one or more arguments given

```
floppy> pwd Dir1
/Dir (should ignore arguments given)
```

Case 2: No arguments given

```
floppy> pwd
/Dir
```

Case 3: Immediately after start up of system

```
floppy> pwd

(must return root directory)
```

cat

Output of ls is given below will be used for all cases:

```
floppy> ls
Name           Type      File size      FLC
.              Dir       512            10
..             Dir       512            2
Dir1           Dir       512            21
help.txt       File      35             22
```

Case 1: Argument is a file

```
floppy> cat help.txt

(contents of help.txt printed here)
```

Case 2: Argument is a directory

```
floppy> cat Dir1
Argument is not a file.
```

Case 3: More than one argument is given

```
floppy> cat help.txt Dir1
Please list only one file.
```

Case 4: No arguments are given

```
floppy> cat
Please specify a file.
```

Case 5: File given does not exist

```
floppy> cat hereIam.txt
"hereIam.txt" does not exist.
```

Touch

```
>> if(-e testFile) then rm testFile
>> touch testFile
```

Should create a file named "testFile" in the current working directory.

```
>> touch /testFile
```

Should create a file named "testFile" in the root directory.

```
>> touch ../testFile
```

Should create a file named "testFile" in the parent directory.

```
>> mkdir ./myFolder
>> touch ./myFolder/testFile
```

Should create a file named "testFile" in the folder "myFolder" that lives inside the current working directory.

```
>> touch testFile
```

Should do nothing as "testFile" already exists.

```
>> touch ../noFolder/testFile
```

Should error assuming that "noFolder" doesn't exist.

Shell

For testing purposes, it will output the string representation of each program it calls.

Case 1: Shell is able to run a program without command line arguments

```
floppy> ls
Name           Type      File size      FLC
help.txt       File      35             22
foo.bar        File      23             23
```

Case 2: Shell is able to run a program with command line arguments

```
floppy> ls help.txt
Name           Type      File size      FLC
help.txt       File      35             22
```

Case 3: Shell properly initializes the shared memory containing the current working directory and the stream for the floppy image.

Case 4: Shell stores the root directory as the current working directory at startup

```
csh # ./shell floppy1
floppy> pwd
/
```

cd

```
>> ls
```

Should list current working directory.

```
>> cd .
>> ls
```

Should again list current working directory.

```
>> cd ..
>> ls
```

Should list parent directory.

```
>> cd /
>> ls
```

Should list root directory.

```
>> cd ./child
>> ls
```

Should list contents of the subdirectory “child.”

```
>> cd prog.c
```

Should error since “prog.c” is not a directory.

```
>> cd /class/cs/csse230
>> ls
```

Should list contents of the cs230 folder.

ls

Case 1: Argument is a file (relative path)

```
floppy> ls ../help.txt
Name           Type      File size      FLC
help.txt       File      35             22
```

Case 2: Argument is a file (absolute path)

```
floppy> ls /team6/os/help.txt
Name           Type      File size      FLC
help.txt       File      35             22
```

Case 3: Argument is a directory (relative)

```
floppy> ls ../
Name           Type      File size      FLC
.              Dir       512            10
..             Dir       512            2
Shell.exe      File      78             20
Dir1           Dir       512            21
help.txt       File      35             22
```

Case 4: Argument is a directory (absolute)

```
floppy> ls /team6/os/
Name           Type      File size      FLC
.              Dir       512            10
..             Dir       512            2
Shell.exe      File      78             20
Dir1           Dir       512            21
help.txt       File      35             22
```

Case 5: Argument is a file or directory that doesn't exist

```
floppy> ls /team6/0s/
The directory or file you have specified cannot be found.
```

Case 6: No argument is given

```
floppy> ls
Name           Type      File size      FLC
.              Dir       512            10
..             Dir       512            2
MoreHlp.txt    File      78             56
```

Case 7: Argument is "." or ".."

```
floppy> ls ..
Name           Type      File size      FLC
.              Dir       512            10
..             Dir       512            2
Shell.exe      File      78             20
Dir1           Dir       512            21
help.txt       File      35             22
```

Case 8: More than one argument given

```
floppy> ls / /team6/os/
Usage: ls [file | directory]
```

rm

Given the following results of an ls execution (a command assumed to have been previously tested) before *each* case:

```
floppy> ls
Name            Type      File size      FLC
.               Dir       512            10
..              Dir       512            2
Shell.exe       File      78             20
Dir1            Dir       512            21
help.txt        File      35             22
```

Case 1: Argument is a file (relative path)

```
floppy> rm Shell.exe
floppy> ls
Name            Type      File size      FLC
.               Dir       512            10
..              Dir       512            2
Dir1            Dir       512            21
help.txt        File      35             22
```

Case 2: Argument is a file (absolute path)

```
floppy> rm /team6/os/Shell.exe
floppy> ls
Name            Type      File size      FLC
.               Dir       512            10
..              Dir       512            2
Dir1            Dir       512            21
help.txt        File      35             22
```

Case 3: Argument is a directory (relative)

```
floppy> rm Dir1
"Dir1" is not a file.
Usage: rm file

floppy> ls
Name            Type      File size      FLC
.               Dir       512            10
..              Dir       512            2
Shell.exe       File      78             20
Dir1            Dir       512            21
help.txt        File      35             22
```

Case 4: Argument is a directory (absolute)

```
floppy> rm /team6/os/Dir1
"/team6/os/Dir1" is not a file.
Usage: rm file

floppy> ls
Name           Type      File size      FLC
.              Dir       512            10
..             Dir       512            2
Shell.exe      File      78             20
Dir1           Dir       512            21
help.txt       File      35             22
```

Case 5: Argument is a file that doesn't exist

```
floppy> rm /team6/os/Shll.exe
"/team6/os/Shll.exe" is not a file.
Usage: rm file

floppy> ls
Name           Type      File size      FLC
.              Dir       512            10
..             Dir       512            2
Shell.exe      File      78             20
Dir1           Dir       512            21
help.txt       File      35             22
```

Case 6: No argument is given

```
floppy> rm
Please specify a file
Usage: rm file

floppy> ls
Name           Type      File size      FLC
.              Dir       512            10
..             Dir       512            2
Shell.exe      File      78             20
Dir1           Dir       512            21
help.txt       File      35             22
```

Case 7: Argument is "." or ".."

```
floppy> rm .
"/team6/os" is not a file.
Usage: rm file

floppy> ls
Name           Type      File size      FLC
.              Dir       512            10
..             Dir       512            2
Shell.exe      File      78             20
Dir1           Dir       512            21
help.txt       File      35             22
```

rmdir

Given the following results of an ls execution:

```
floppy> ls
Name           Type      File size      FLC
.              Dir       512            10
..             Dir       512            2
Shell.exe      File      78             20
Dir1           Dir       512            21
help.txt       File      35             22
Dir3           Dir       512            23
```

Case 1: Argument is a file (relative path)

```
floppy> rmdir Shell.exe
"Shell.exe" is not a directory.
Usage: rmdir directory

floppy> ls
Name           Type      File size      FLC
.              Dir       512            10
..             Dir       512            2
Shell.exe      File      78             20
Dir1           Dir       512            21
help.txt       File      35             22
Dir3           Dir       512            23
```

Case 2: Argument is a file (absolute path)

```
floppy> rmdir /team6/os/Shell.exe
"Shell.exe" is not a directory.
Usage: rmdir directory

floppy> ls
Name           Type      File size      FLC
.              Dir       512            10
..             Dir       512            2
Shell.exe      File      78             20
Dir1           Dir       512            21
help.txt       File      35             22
Dir3           Dir       512            23
```

Case 3: Argument is an empty directory (relative)

```
floppy> rmdir Dir1

floppy> ls
Name           Type      File size      FLC
.              Dir       512            10
..             Dir       512            2
Shell.exe      File      78             20
help.txt       File      35             22
Dir3           Dir       512            23
```

Case 4: Argument is an empty directory (absolute)

```
floppy> rmdir /team6/os/Dir1

floppy> ls
Name           Type      File size      FLC
.              Dir       512            10
..             Dir       512            2
Shell.exe      File      78             20
help.txt       File      35             22
Dir3           Dir       512            23
```

Case 5: Argument is a directory that doesn't exist

```
floppy> rmdir /team6/os/Dir2
"/team6/os/Dir2" is not a directory.
Usage: rmdir directory

floppy> ls
Name           Type      File size      FLC
.              Dir       512            10
..             Dir       512            2
Shell.exe      File      78             20
Dir1           Dir       512            21
help.txt       File      35             22
Dir3           Dir       512            23
```

Case 6: No argument is given

```
floppy> rmdir
Please specify a directory
Usage: rmdir directory

floppy> ls
Name           Type      File size      FLC
.              Dir       512            10
..             Dir       512            2
Shell.exe      File      78             20
Dir1           Dir       512            21
help.txt       File      35             22
Dir3           Dir       512            23
```

Case 7: Argument is a non-empty directory

```
floppy> ls Dir3
Name           Type      File size      FLC
.              Dir       512            10
..             Dir       512            2
File1.exe      File      45             15

floppy> rmdir Dir3
"Dir3" is not empty.
Usage: rmdir directory

floppy> ls
Name           Type      File size      FLC
.              Dir       512            10
..             Dir       512            2
Shell.exe      File      78             20
Dir1           Dir       512            21
help.txt       File      35             22
Dir3           Dir       512            23
```

Case 8: Argument is "." or ".."

```
floppy> rmdir .
"/team6/os" is not empty.
Usage: rmdir directory

floppy> ls
Name           Type      File size      FLC
.              Dir       512            10
..             Dir       512            2
Shell.exe      File      78             20
Dir1           Dir       512            21
help.txt       File      35             22
Dir3           Dir       512            23
```