

---

# FAT32 Utility Operations Guide

---

Classes COP4610 / CGS5765  
Florida State University

---

# Outline

- Directory entries
  - Short-name and long-name entries
- *Read-only* project 3 operations
- Other *non-read-only* project 3 operations

---

# Directory Entries

---

---

# Directory Entries

- Lists names of files and directories in a directory
- Types
  - ❑ Short-name directory entry
  - ❑ Long-name directory entry

---

# Short-name Directory Entry

- Limits name size to 8 bytes with additional 3 bytes after “.”
- Compatible with previous FAT versions
- 32 bytes total size
- Holds important information about file or dir:
  - Attributes, timestamp, last access date, first cluster number, size

# Short-name Directory Entry

## ■ FAT32 Directory Entry Structure

Name	Offset (byte)	Size (bytes)	Description
DIR_Name	0	11	Short Name
DIR_Attr	11	1	File Attributes (More on it later)
DIR_NTRes	12	1	Reserved for Windows NT
DIR_CrtTimeTenth	13	1	Millisecond stamp at file creation time
DIR_CrtTime	14	2	Time file was created
DIR_CrtDate	16	2	Date file was created

# Short-name Directory Entry

## ■ FAT32 Directory Entry Structure

Name	Offset (byte)	Size (bytes)	Description
DIR_LstAccDate	18	2	Last access date
DIR_FstClusHI	20	2	High word of this entry's first cluster number
DIR_WrtTime	22	2	Time of last write
DIR_WrtDate	24	2	Date of last write
DIR_FstClusLO	26	2	Low word of this entry's first cluster number
DIR_FileSize	28	4	32-bit DWORD holding this file's size in bytes

---

# Short-name Directory Entry

- Byte 11: DIR\_Attr

Bit	7	6	5	4	3	2	1	0
Attribute	Reserved Set to 0		Archive	Directory	Volume ID	System	Hidden	Read- only



---

# Short-name Directory Entry

- Check page 23 on FAT32 Spec document for detailed descriptions
- For the correct implementation of this project, setting *DIR\_name*, *DIR\_Attr*, *DIR\_FstClusHI*, *DIR\_FstClusLO*, *DIR\_FileSize* correctly is essential
- You may lose a point or two if you don't set the other fields correctly

---

# Short-name Directory Entry

- If `DIR_Name[0] == 0xE5`, then the directory entry is free (no file or directory name in this entry)
- If `DIR_Name[0] == 0x00`, then the directory entry is free (same as for `0xE5`), and there are no allocated directory entries after this one

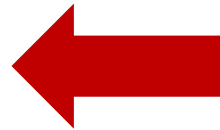
---

# Long-name Directory Entry

- Backwards-compatible way to allow longer names to be displayed
- Each long-name directory entry is 32 bytes
  - A long file name can cover a **set** of long-name directory entries
- Each set of long-name directory entries must correspond to a short-name directory entry
  - Long-name entries must immediately precede corresponding short-name entry

---

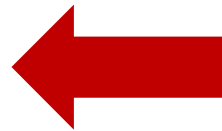
# Long-name Directory Entry



In this example case,  
two long-name  
entries are needed to  
hold the file name

---

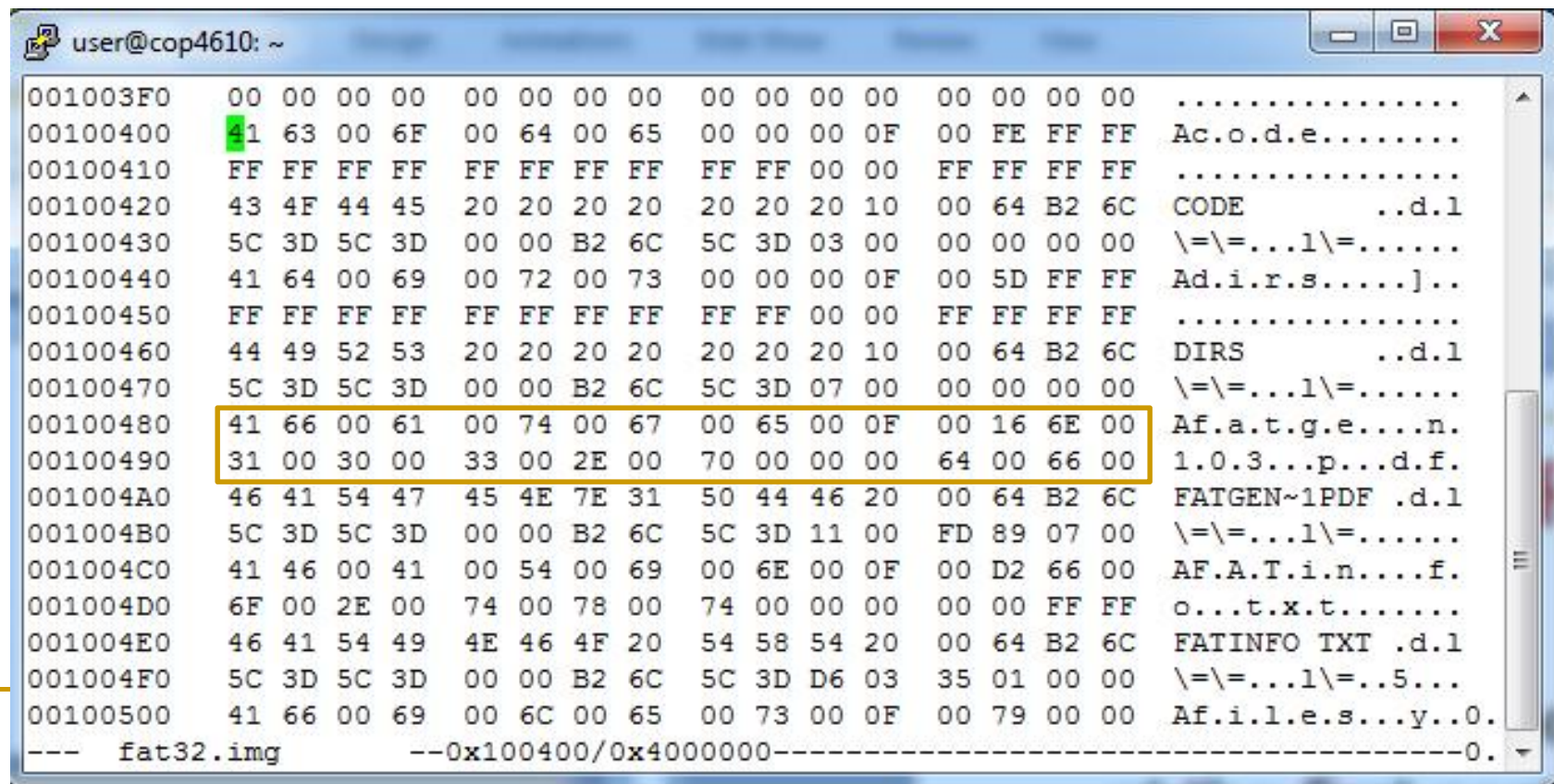
# Long-name Directory Entry



Short name entry for the file must exist too, and it immediately follows the long name entry(s)

# Directory entries

- Long-name entry for “fatgen103.pdf”



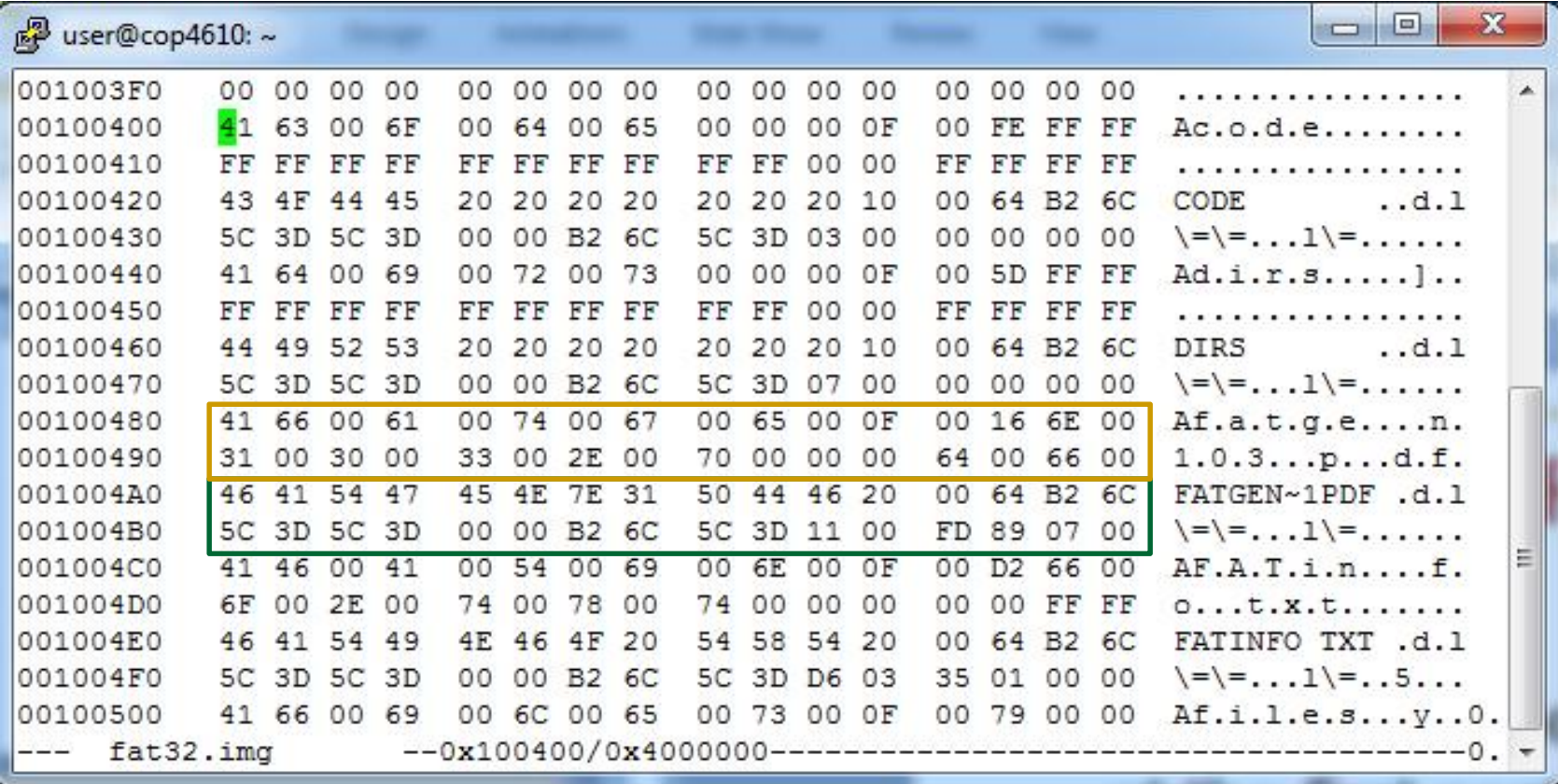
The screenshot shows a hex editor window titled "user@cop4610: ~". It displays a list of directory entries from a FAT32 file system. The entries are listed in hexadecimal format, with the first column showing the entry's address and the subsequent columns showing the entry's data. The entry for "fatgen103.pdf" is highlighted with a yellow box. The entry's data is as follows:

Address	Entry Data (Hex)	Entry Name (ASCII)
001003F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00100400	41 63 00 6F 00 64 00 65 00 00 00 0F 00 FE FF FF	Ac.o.d.e.....
00100410	FF FF FF FF FF FF FF FF FF FF 00 00 FF FF FF FF	.....
00100420	43 4F 44 45 20 20 20 20 20 20 20 10 00 64 B2 6C	CODE ..d.l
00100430	5C 3D 5C 3D 00 00 B2 6C 5C 3D 03 00 00 00 00 00	\=\=...l\=.....
00100440	41 64 00 69 00 72 00 73 00 00 00 0F 00 5D FF FF	Ad.i.r.s.....]
00100450	FF FF FF FF FF FF FF FF FF FF 00 00 FF FF FF FF	.....
00100460	44 49 52 53 20 20 20 20 20 20 20 10 00 64 B2 6C	DIRS ..d.l
00100470	5C 3D 5C 3D 00 00 B2 6C 5C 3D 07 00 00 00 00 00	\=\=...l\=.....
00100480	41 66 00 61 00 74 00 67 00 65 00 0F 00 16 6E 00	Af.a.t.g.e....n.
00100490	31 00 30 00 33 00 2E 00 70 00 00 00 64 00 66 00	1.0.3...p...d.f.
001004A0	46 41 54 47 45 4E 7E 31 50 44 46 20 00 64 B2 6C	FATGEN~1PDF .d.l
001004B0	5C 3D 5C 3D 00 00 B2 6C 5C 3D 11 00 FD 89 07 00	\=\=...l\=.....
001004C0	41 46 00 41 00 54 00 69 00 6E 00 0F 00 D2 66 00	AF.A.T.i.n....f.
001004D0	6F 00 2E 00 74 00 78 00 74 00 00 00 00 00 FF FF	o...t.x.t.....
001004E0	46 41 54 49 4E 46 4F 20 54 58 54 20 00 64 B2 6C	FATINFO TXT .d.l
001004F0	5C 3D 5C 3D 00 00 B2 6C 5C 3D D6 03 35 01 00 00	\=\=...l\=..5...
00100500	41 66 00 69 00 6C 00 65 00 73 00 0F 00 79 00 00	Af.i.l.e.s...y..0.

At the bottom of the window, the text "fat32.img" and "--0x100400/0x4000000" is visible, indicating the file being edited and the current offset.

# Directory entries

- Short-name entry for “fatgen103.pdf”



```
user@cop4610: ~
001003F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00100400  41 63 00 6F 00 64 00 65 00 00 00 0F 00 FE FF FF Ac.o.d.e.....
00100410  FF FF FF FF FF FF FF FF FF FF 00 00 FF FF FF FF .....
00100420  43 4F 44 45 20 20 20 20 20 20 20 10 00 64 B2 6C CODE ..d.l
00100430  5C 3D 5C 3D 00 00 B2 6C 5C 3D 03 00 00 00 00 00 \=\=...l\=.....
00100440  41 64 00 69 00 72 00 73 00 00 00 0F 00 5D FF FF Ad.i.r.s.....]..
00100450  FF FF FF FF FF FF FF FF FF FF 00 00 FF FF FF FF .....
00100460  44 49 52 53 20 20 20 20 20 20 20 10 00 64 B2 6C DIRS ..d.l
00100470  5C 3D 5C 3D 00 00 B2 6C 5C 3D 07 00 00 00 00 00 \=\=...l\=.....
00100480  41 66 00 61 00 74 00 67 00 65 00 0F 00 16 6E 00 Af.a.t.g.e....n.
00100490  31 00 30 00 33 00 2E 00 70 00 00 00 64 00 66 00 1.0.3...p...d.f.
001004A0  46 41 54 47 45 4E 7E 31 50 44 46 20 00 64 B2 6C FATGEN~1PDF .d.l
001004B0  5C 3D 5C 3D 00 00 B2 6C 5C 3D 11 00 FD 89 07 00 \=\=...l\=.....
001004C0  41 46 00 41 00 54 00 69 00 6E 00 0F 00 D2 66 00 AF.A.T.i.n....f.
001004D0  6F 00 2E 00 74 00 78 00 74 00 00 00 00 00 FF FF o...t.x.t.....
001004E0  46 41 54 49 4E 46 4F 20 54 58 54 20 00 64 B2 6C FATINFO TXT .d.l
001004F0  5C 3D 5C 3D 00 00 B2 6C 5C 3D D6 03 35 01 00 00 \=\=...l\=..5...
00100500  41 66 00 69 00 6C 00 65 00 73 00 0F 00 79 00 00 Af.i.l.e.s...y..0.
--- fat32.img --0x100400/0x4000000-----0.
```

---

# Long-name Directory Entries

- You can ignore the long directory entries
  - Can just display the short names
  - This makes the project easier



---

# Long-name Directory Entries

- How to know a directory entry is a long-name entry?
  - Byte 11: DIR\_Attr
    - ATTR\_READ\_ONLY | ATTR\_HIDDEN | ATTR\_SYSTEM | ATTR\_VOLUME\_ID → ATTR\_LONG\_NAME

---

# “Dot” Entries

- All directories (except root directory of entire system) have “.” and “..” directory entries
- “.” means “this directory”
- “..” means “the parent directory”
- Why do you think the root directory does not have these entries?

---

## Sub-directories

- ATTR\_Directory flag is set in the directory entry
- Treated just like a file in terms of cluster allocation
- Clusters contain 32 bytes directory entries, for the files and directories under this directory

---

# Utility Operations

---

---

# FAT32 Utility Operations

Utility recognizes the following built-in commands:

- open
- close
- create
- rm
- size
- cd
- ls
- mkdir
- rmdir
- read
- write

---

# A Few Definitions

- ***Read-Only Operations*** – can be completed without modifying file system image
  - ***Write Operations*** – must modify file system image to complete
  - *Hint:* Do the read-only operations first since they should not corrupt your image
-

---

# FAT32 Utility Operations

## Classified

### Read-Only

- open
- close
- ls
- size
- cd
- read

### Write

- create
- rm\*\*
- mkdir
- rmdir\*\*
- write

*\*\*Will go over rm and rmdir next week*

---

# Read-Only Operations

---



---

# Read-Only Precautions

- File or directory must ***exist*** before performing operations on it
- File must be ***open*** and flagged for reading before you attempt to read from it
- Be sure you are reading from the right location
  - ❑ Off by 1 byte can throw whole project off

---

## Read-Only Operation: **open**

1. Check if the file is already open
  2. Check that the mode-specifiers are valid (r, w, rw, or wr)
  3. Check that the provided file name exists in the requested directory
  4. If it exists, add the file to your open file table (or some similar data structure) with mode-specifiers
-

---

# open Use Cases

## ■ Successful open

```
/] open FATINFO.TXT rw  
/]
```

## ■ Unsuccessful open

```
/] open FATINFO.TXT rw  
Error: already open!  
/]
```

---

---

# open Use Cases

## ■ Unsuccessful open

```
/] open NOTHERE.TXT rw  
Error: does not exist  
/]
```

## ■ Unsuccessful open

```
/] open DIRS rw  
Error: cannot open a directory  
/]
```

---

---

# open Use Cases

- Unsuccessful open

```
/] open FATINFO.TXT z
```

```
Error: incorrect parameter
```

```
/]
```

---

## Read-Only Operation: `close`

1. Check that the file name provided exists in your open file table
2. If it does, remove that entry from your open file table

---

# close Use Cases

## ■ Successful close

```
/] close FATINFO.TXT  
/]
```

## ■ Unsuccessful close

```
/] close NOTHERE.TXT  
Error: file not in open file table  
/]
```

---

---

## Read-Only Operation: `ls`

1. Make sure that provided directory name is directory and exists in requested directory
  2. Seek first data cluster
  3. Iterate through and print each directory entry in the cluster
  4. If more directory entries left than first cluster can hold, seek next cluster and repeat 3
-



---

# ls Use Cases

## ■ Successful ls

```
/DIRS/] ls .
```

```
.    ..   A    B    C    D
```

```
/DIRS/]
```

---

## Read-Only Operation: **size**

1. Check that provided file name exists in the requested directory
    - Can be accomplished by seeking through the clusters of the requested directory
  2. If it does, extract the size information
    - Pay attention to endianness!
-

---

# size Use Cases

## ■ Successful size

```
/FILES/] size CONST.TXT
```

```
45119
```

```
/FILES/]
```

## ■ Unsuccessful size

```
/FILES/] size NOTHERE.TXT
```

```
Error: file does not exist
```

```
/FILES/]
```

---

---

## Read-Only Operation: `cd`

1. Check that provided directory name is a directory and exists
  2. Alter your current working directory to reflect the change
    - ❑ For ease of debugging and use, you may want to alter your prompt to show current working directory
-

---

# cd Use Cases

## ■ Successful cd

```
/] cd FILES  
/FILES/]
```

## ■ Unsuccessful cd

```
/] cd FATINFO.TXT  
Error: not a directory  
/]
```

---

---

# cd Use Cases

- Unsuccessful cd

```
/] cd NOTHERE
```

```
Error: does not exist
```

```
/]
```

---

## Read-Only Operation: **read**

1. Make sure file name provided is in open-file table and flagged as read-capable
  2. Check that the provided position is valid
  3. Check that the requested number of bytes is valid
  4. Seek to data cluster corresponding to the requested start position and begin reading
  5. If more data to be read, seek the next clusters and repeat 4
-

---

# read Use Cases

## ■ Successful read

```
/FILES/] read CONST.TXT 0 15
```

Provided by USC

```
/FILES/]
```

## ■ Unsuccessful read

```
/FILES/] read EMPTY.TXT 45 99
```

Error: attempt to read beyond EoF

```
/FILES/]
```

---



---

# Write Operations

---

---

# Write Precautions

- File must be ***open*** and flagged for writing before you attempt to write to it
- Make sure an entry name is not a directory before you try to write to it
- Check how much space is left in a cluster when writing a new string
  - ❑ Don't want to overwrite other pre-existing data

---

# Write Operations

- Many write operations may involve allocating a new cluster

---

# Allocating a New Cluster

1. Search the FAT table for any free clusters
    - ❑ If none, return an out of space error!
  2. Set the previous cluster to point to the new cluster number
    - ❑ Watch out, there may be more than one FAT to update
  3. Set the new cluster to EoC (end of cluster chain)
-

---

# Write Operations

- Many write operations involve creating a new directory entry

---

# Creating a New Directory Entry

- Just create a short-name directory entry
  - All new directories will be of length 8 characters or less

---

## Write Operation: `write`

1. Check that the parameters passed are valid (as for the read operation)
  2. Seek the data cluster position requested by the operation
  3. Write as much data as you can fit starting at the requested position up until the end of a given cluster
  4. If a cluster fills up, allocate a new cluster
  5. Repeat 3-4 until the write is complete
-

---

# write Use Cases

## ■ Successful write

```
/FILES/] open EMPTY.TXT rw
/FILES/] write EMPTY.TXT 0 10 "Not empty!"
/FILES/]
```

## ■ Unsuccessful write

```
/FILES/] open EMPTY.TXT r
/FILES/] write EMPTY.TXT 0 10 "Not empty!"
Error: File is not open for writing
/FILES/]
```

---



---

# write Use Cases

## ■ Unsuccessful write

```
/FILES/] write EMPTY.TXT 0 10 "Not empty!"  
Error: File not found in open file table  
/FILES/]
```

---

# Write Operation: create

1. Make sure the requested file name does NOT exist in the requested location
  2. Create new directory entry for the file
    - ❑ If there is enough room in the current cluster, write it there
    - ❑ If there is not enough space left in the cluster, allocate a new cluster and write it in the new cluster
-

---

# create Use Cases

## ■ Successful create

```
/FILES/] create HELLO.TXT  
/FILES/] ls  
. .. CONST.TXT EMPTY.TXT HELLO.TXT  
/FILES/]
```

## ■ Unsuccessful create

```
/FILES/] create EMPTY.TXT  
Error: File already exists  
/FILES/]
```

---

## Write Operation: `mkdir`

- Similar to `create`, except give the directory entry the proper directory attribute

---

# **mkdir Use Cases**

## ■ Successful mkdir

```
/DIRS/] mkdir NEW
```

```
/DIRS/] ls
```

```
.      ..      NEW      A      B      C      D
```

```
/DIRS/]
```

---

# **mkdir Use Cases**

## ■ Unsuccessful mkdir

```
/DIRS/] mkdir A
```

```
Error: directory already exists
```

```
/DIRS/]
```

---

# Next Week

- Operations rm and rmdir
- Answering any more questions