Operating Systems

File Assignment

Assignment

- Disks (and almost all “storage devices”) offer several important features:

- The contents don’t disappear when there is no power (persistence)
- There is a way to name and find information (directory)
- Files may be of different sizes, and stored efficiently
- After a file is created it may grow larger or smaller, it may be deleted, or renamed

Assignment

- Disks (and almost all “storage devices”) offer several important features:

- There is a way to find out what is on the disk (list directory)
- There is some information about the files which is not a part of the file (meta-information), such as file size, sometimes time&date the file was created or modified, sometimes who has permissions to read it or modify it

- There is usually some overhead, such as knowing which part of the disk is being used (occupied), and which part is free (free space)

Assignment

- So there is a need to keep some information on the disk about the files and directories on that disk

- There needs to be a way to find that information, that is the layout or format of the disk (directories, files, free space) must be specified, at least partially (may expand later, want to change)

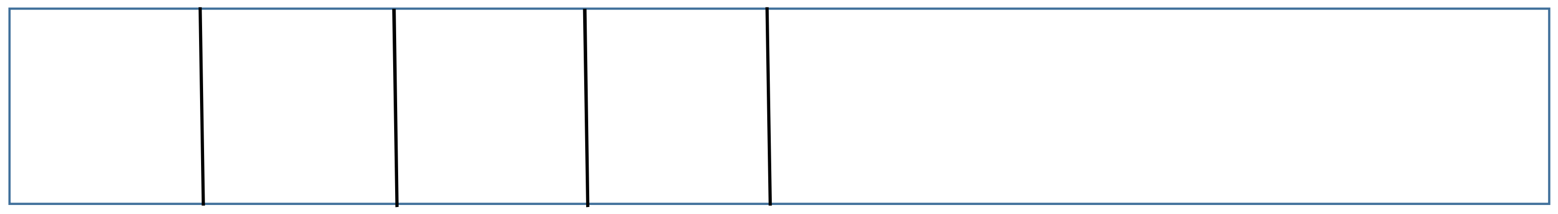
Assignment

- Maybe a top down approach might help

- There are disks, they are divided into fixed size sectors
- In this assignment you are using **a file** to ***emulate*** (to act as) a disk
- So you do not destroy the contents of your disk
- The name of the file, which acts like a disk is “**disk01**”
- That file is logically divided into **256 byte** sectors

Assignment

- Logically this looks like
- “disk01”



- 0 256 512 768 1024
- offset

Assignment

What goes into these sectors? (Of “disk01”)

The contents of files. (You are copying the contents of a **host** file into this file)

A simple (but not very useful) example is:

The Unix command

Assignment

What goes into these sectors? Of “disk01”

A simple (but not very useful) example is:

The Unix command:

```
cat file01 file02 file03 >disk01
```

This works on files of any type: jpg, text, executable files

But how can you later separate disk01 back into these separate files?

Assignment

What goes into these sectors? Of “disk01”?

But how can you later separate disk01 back into these separate files?

How do you know where file01 ends and file02 begins?

This is what directories do

They have meta information (additional info) that is not part of a files contents

Assignment

What goes into these sectors? Of “disk01”?

Directory:

Has file names:

File01 → location 0

File02 → location 512

File03 → location 768

Assignment

What goes into these sectors? Of “disk01”?

Directory:

Has file names:

File01 $\rightarrow$ location 0

File02 $\rightarrow$ location 512

File03 $\rightarrow$ location 768

(notice that these are on sector boundaries, an integer number of sectors)

Assignment

Directory:

File01 → location 0

File02 → location 512

File03 → location 768

File01	0
File02	512
File03	768

Assignment

- Directory

- How large is each file? (in bytes)? We know in sectors (maybe), not in bytes
- When was file created?

Assignment

Directory (split into simple parts):

File01 → inode number 5

File02 → inode number 3

File03 → inode number 8

File01	5
File02	3
File03	8

Assignment

Directory (split into simple parts):

5	Size: 480, July 4
3	Size: 10, July 1
8	….

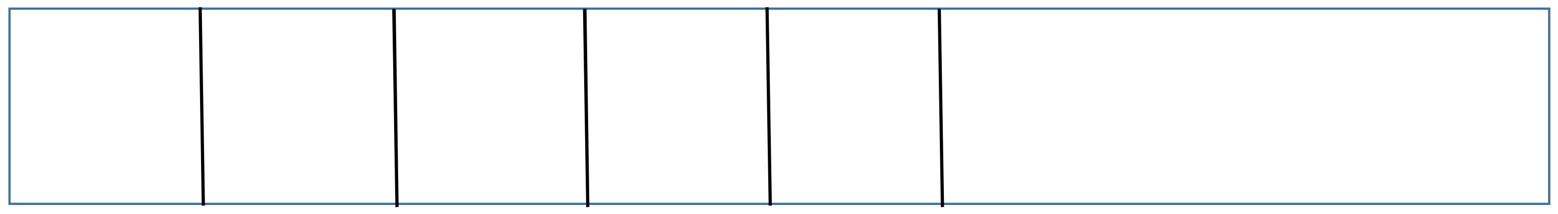
Assignment

Directory (split into simple parts):

5	Size: 480, July 4	where? Sector 11, sector 8
3	Size: 10, July 1	where? Sector 10
8	….	

Assignment

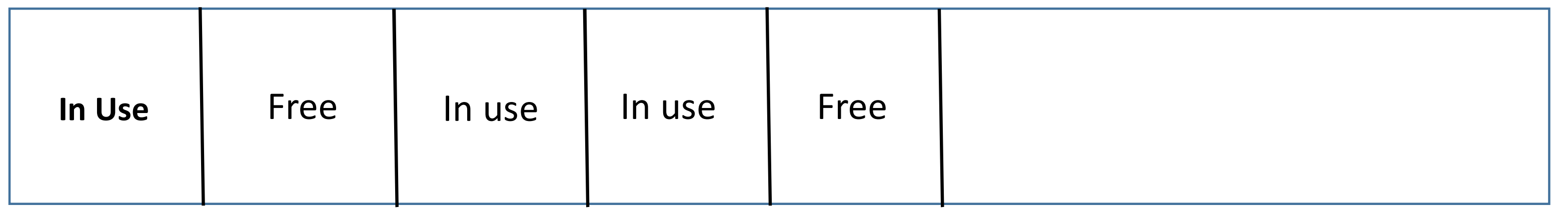
- Free Space ("free" or unused disk blocks)
- "disk01"



- 0 256 512 768 1024 1280

Assignment

- Free Space ("free" or unused disk blocks)
- "disk01"



- 0 256 512 768 1024 1280
- Bit Map of Free Space: 1 0 1 1 0, where 1 is in use, 0 is free

Assignment

- These files must be of type “binary”.
- https://stackoverflow.com/questions/979816/whats-a-binary-file-and-how-do-i-create-one
- In addition to the system calls: open, close, read,write (there are many varients) you may need to use lseek.

- You may look at the contents of any file (binary, text, jpg, anything)
- With the utility: “od”, such as “od –A x –c disk01”
- Can also look at “xxd”, Hexedit or Ghex

Assignment

- File names stored in a directory
- Only name and pointer to additional information
- Called FNT (File Name Table)
- Entries:
- Name (max 50 characters)
- A number (DABPT index, or pointer) this is an “inode”
- (that is all)

Assignment

- File names stored in a directory
- Only name and pointer to additional information
- Called FNT (File Name Table)
- Entries:
- Name (max 50 characters)
- A number (DABPT index, or pointer) this is an “inode”
- (that is all)

Assignment

- File meta information
- DABPT (Directory block pointer table)
- Contains
- Current file size (integer)
- Time and date (integer)
- User Name (string)
- Pointer to another table (Block Pointer Table)

Assignment

- File meta information
- Block Pointer Table
- These are “pointers” (integers) for “actual disk blocks”
- Which in this case are just 256 byte groups in memory
- Or on the disk
-

Assignment

- File meta information
- Why these tables with pointers?

- These pointers are NOT memory pointers, they are an index (0,1,2…)
- These (hopefully):
  - Simplify design (coding, debugging)
  - Avoid variable length strings, groups of pointers, etc
  - Variable length data is a pain, allocating max lengths is wasteful (and may not work anyway)

Assignment

- Some people think better (easier) with pictures
- Draw them
- Or create data structures in code

- This can/should be developed incrementally

Last

Last