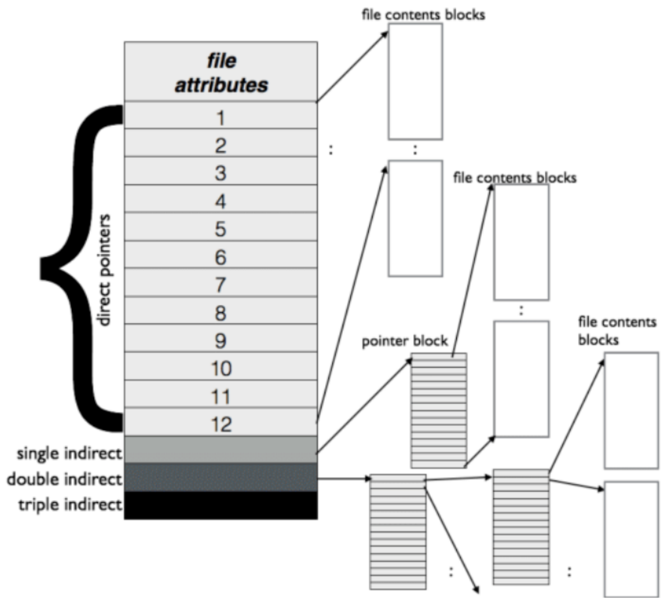
1. a) 1) 4 - inode blocks. 1 for the file `c`, and 3 for the directdories `/`, `a`, `b`
   2) 3 - directory blocks - one for root `/`, one for `a`, the other for `b`
   3) 1 - single indirect block as far as we know. The file definitely has more than 12 blocks (# of data blocks pointed by direct pounters), but less than 1036 (# of data blocks pointed by direct pointers and single indirect pointers). We are reading block 1034.
   4) 1 - data block for file `c`

   b) All of the above

Notes

- **Inode**



  - Is short form of **index node**
  - Describes a file system object such as file or data
  - Contains all information about a file/directory, including
    * File Type,
    * Size
    * Number of blocks allocated to it
    * Protection information
    * Time information (e.g time created, time modified)
    * Location of data blocks residing on disk

References

1) Wikipedia, Inode, link
2) Machanick, Philip. (2016). Teaching Operating Systems: Just Enough Abstraction. 642. 10.1007/978-3-319-47680-3_10., link

c) Size, the location of data blocks that reside on disk

Notes

- I wonder what information about blocks inode has. Is it total number of blocks both inode and data, or just data?
- I struggled a bit on this one. I should find an easier way to remember which information inode has

d) **Rough Work**

- **Creash Scenarios**
  - When only new data block is written to disk
    - This is fine in system's point of view
    - No inode points to it (it doesn't contain any information about file)
    - No bitmap points to it
    - Is as if write never occured
  - When only the updated inode is written to disk
    - There is no bitmap that's pointing to it
    - There is new inode where existing inode is
    - The data block `Db` hasn't been created
    - Reading data where `Db` is will return garbage data
    - there is a term for this. Is called **File-System inconsistency**
  - When only inode bitmap is written to disk
    - inode block pointed by bitmap is assumed to be allocated
    - But there is no desired inode where it's pointing
    - This is another example of **File-System-Inconsistency**
    - If left as is, then space cannot be used for future use (**inode leak**)
  - When only data bitmap is written to disk
    - data block pointed by bitmap is assumed to be allocated
    - But there is no desired inode where it's pointing
    - This is another example of **File-System-Inconsistency**
    - If left as is, then space cannot be used for future use (**data leak**)

Notes

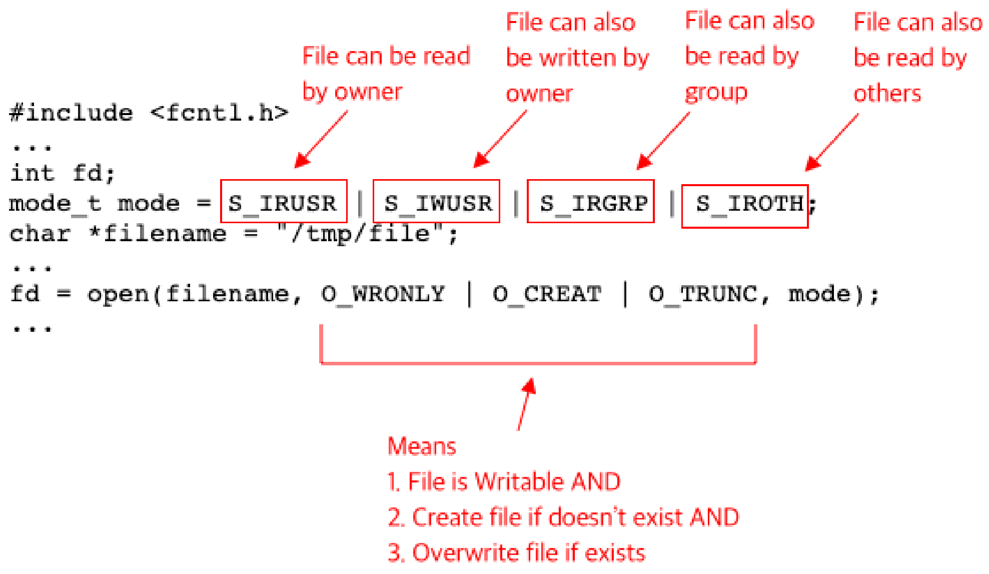
- I wonder how system call for reading file/directory works in UNIX. Does it check for bitmap?
- I wonder how system call for deleting file/directory works in UNIX
- I wonder how system call for creatubg file/directory works in UNIX
- **Creating Files**

- **Syntax:**

  ```
  int fd = open("foo". O_CREAT|O_WRONLY|O_TRUNC, S_IRUSR|S_IWUSR)
  ```

  * Is a system call
  * `O_CREAT` - Creates file "foo" if does not exist
  * `O_WRONLY` - Open file for writing only (default)
  * `O_TRUNC` - Overwrites existing file Need example/Clarification
  * Can have multiple flags
  * Returns **file descriptor** or `fd` for short
    · Is an integer
    · Is private per process
    · Can be used to `read()` and `write()` files

Example



```
#include <fcntl.h>
...
int fd;
mode_t mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;
char *filename = "/tmp/file";
...
fd = open(filename, O_WRONLY | O_CREAT | O_TRUNC, mode);
...
```

File can be read by owner
File can also be written by owner
File can also be read by group
File can also be read by others
Means
1. File is Writable AND
2. Create file if doesn't exist AND
3. Overwrite file if exists

–

- **Reading File**
  - **Syntax:**

    ```
    read (int fd, void *buf, size_t count)
    ```

    * Is a system call
    * `fd` - file descriptor (from `open()`)
    * buf -
- **Reading and Writing Files**
- **Reading and Writing Files**

- **Renaming Files**
- **Removing Files**
- **Making Directories**
- **Reading Directories**
- **Removing Directories**
- **Hard Links**
- **Symbolic Links**