CSE 312
Operating Systems

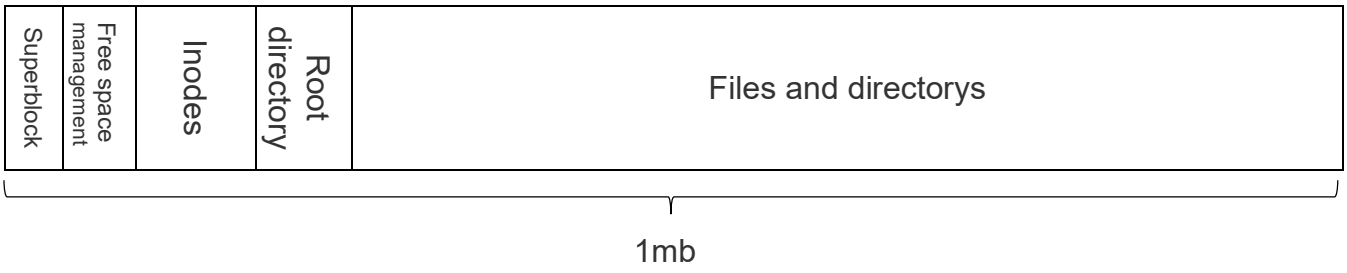
#midterm_project

Jwan hussein

151044078

Disk partition:

The system file has been divided into five parts :



- **the superblock partition** :

  located in the file first block, with the size of block size and holds information about the system as follow

  5 bytes : contain the block size
  4 bytes : contain the number of the blocks
  4 bytes : contain the number of free blocks
  4 bytes : contain the number of the inodes
  4 bytes : contain the number of free inodes
  4 bytes : contain the address of the free space management
  4 bytes : contain the address of the inodes block
  4 bytes : contain the address of the root directory block
  4 bytes : contain the address of the file and directory block
  8 bytes : contain the address of the single indirect linkes
  8 bytes : contain the address of the double indirect linkes
  8 bytes : contain the address of the triple indirect linkes

- **The free space management block :**

  Located in the second block of the the system file ,
  Contain the free block map and the free inodes map

  The block map :

0	1	2	3	4	5	6	7		Block_number
0	1	1	1	1	1	0	0	……	0

  The inode map map :

0	1	2	3	4	5	6	7		inode_number
1	0	0	1	0	1	0	0	……	0

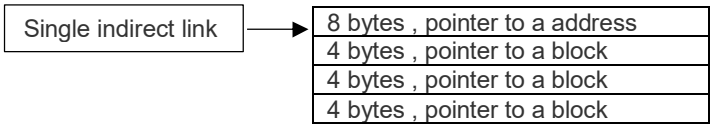
- **The inodes blocks**:
  The inode structure:

attributes

• Size of the file • Point to a file or directory block • The file name • The file extension (not used) • The last modification time • The last modification date • The number of link used	• 8 bytes • 1 byte • 32 bytes • 6 bytes • 8 bytes • 10 bytes • 4 bytes
Direct link 1	4 bytes
Direct link 2	4 bytes
Direct link 3	4 bytes
Direct link 4	4 bytes
Single indirect link	8 bytes
double indirect link	8 bytes
triple indirect link	8 bytes

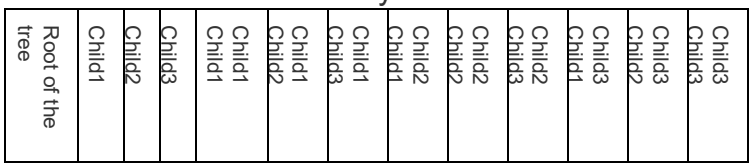
- Size of the file: contain the size of the file that the inode points to
- Point to a file or directory: its one byte either 0 or 1
  If 1 => the inode points to a file
  If 0 => the inode points to a directory
- File name: contain the file name and extension
- File extension: it was declared to hold the file extension but it is unused because directory structure has no extension field.
- Last modification time
- Last modification date
- Number of links used: contain the number of links used to save the file blocks addresses.
  this number can be in maximum 43 and here where the first problem was raised, if the file size is bigger than (34*block size) then there will be no space to keep its block addresses in the indoe structure so it is ignored

- Direct links: point directly to the data blocks
- Single indirect link :



  Point to a memory block of size 12 bytes that is split into 3 part each part contain address to block of the system file
- Double indirect link :

  Pointer to address of the system file which contain a tree



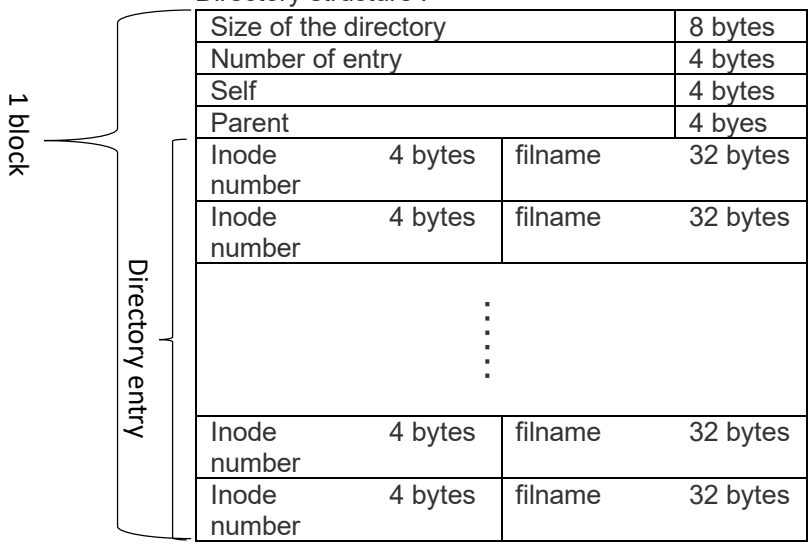
  The second problem raised here the root of the tree and the roots three children was represented as 4 byte long so they could not point to physical address in the file , this problem was noticed in the last day and could not be fixed due to lack of time
- Triple indirect link :

  a tree of height 3 was designed to hold the address of the data blocks but the same problem as in double indirect link occurred again .

due to this problem a file of size bigger than (7 * block size) cannot be written into the system file, so the program writes the first 7 blocks of the input file and ignore the rest of the file and notify the user.

➢ Root director block :

Directory structure :



The root directory is the created first when the system file is created
It can not be removed

- Size of the directory : not used
- Number of entry : the number of file and folders in the directory
- Self : point to the inode which point to the directory
- Parent : point to the inode which point to the parent directory
- Directoy entry : contain file name and the number of the inode the points to the file .

- **Functions:**

```
int op_dumpe2fs(char *Sfilename);
int op_mkdir(char* Sfilename,char* path);
int op_rmdir(char* Sfilename,char* path);
int op_ls(char* Sfilename,char* path);
int op_write(char *Sfilename,char *path,char *filename);
int op_read(char *Sfilename,char *path,char *filename);
```