CMPUT 379 Lab

ETLC E1003: Tuesday, 5:00 – 7:50 PM.

Tianyu Zhang, Peiran Yao

CAB 311: Thursday, 2:00 – 4:50 PM.

Max Ellis, Aidan Bush

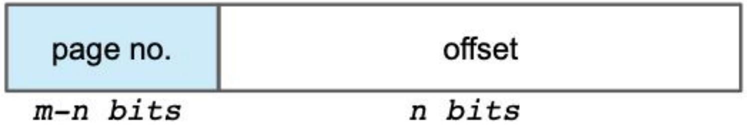
Today’s Lab

- More examples on memory management
- Assignment 3

Memory management examples

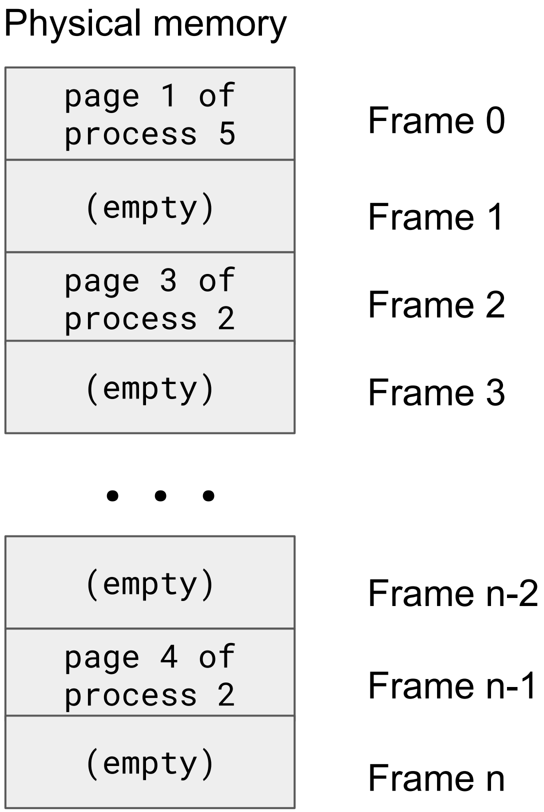
Recap - Paging

- The memory space of a process is divided into fixed-size pages.
- The logical address is divided into page number and offset



Recap - Paging

- The memory space of a process is divided into fixed-size pages.
- The logical address is divided into page number and offset
- The physical memory is divided into fixed-size frames that holds pages



Physical memory

page 1 of process 5 — Frame 0

(empty) — Frame 1

page 3 of process 2 — Frame 2

(empty) — Frame 3

. . .

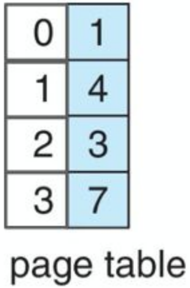
(empty) — Frame n-2

page 4 of process 2 — Frame n-1

(empty) — Frame n

Recap - Paging

- The memory space of a process is divided into fixed-size pages.
- The logical address is divided into page number and offset
- The physical memory is divided into fixed-size frames that holds pages
- The page table of each process stores in which frame each page locates



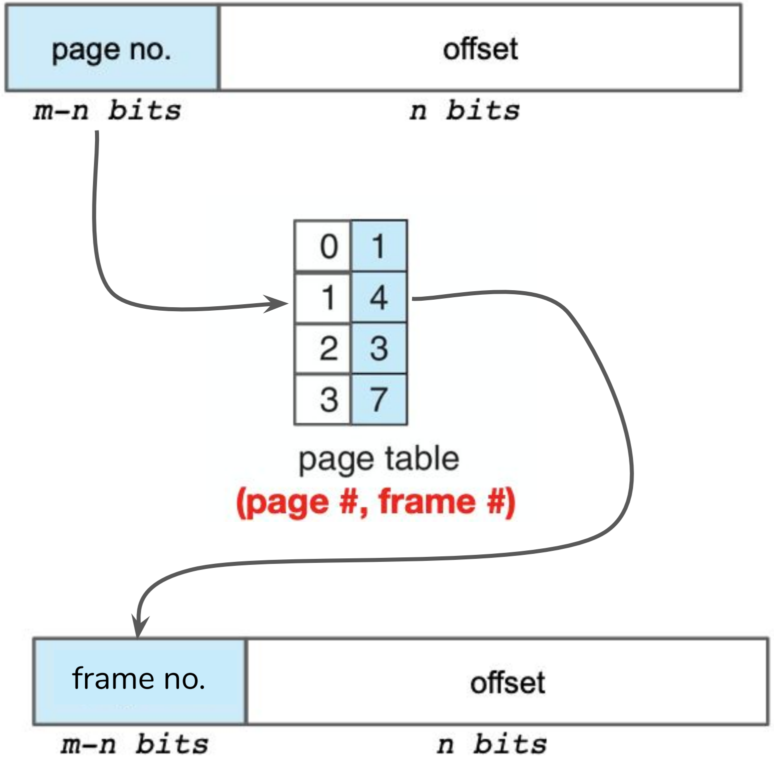
0	1
1	4
2	3
3	7

page table

(page #, frame #)

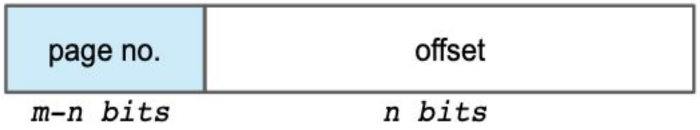
Recap - Paging translation

1. Read the **page no.** and **offset** from a VA
2. Look up the **frame no.** of that **page no.** in the process' page table
3. Replace the **page no.** in the VA with **frame no.** to get the **PA**



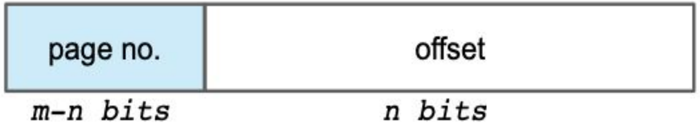
0	1
1	4
2	3
3	7

Page table size



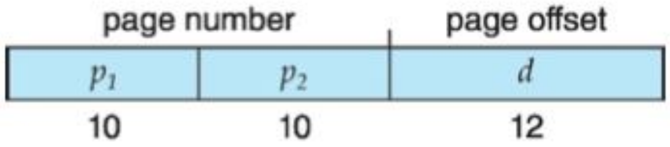
- Assuming we can address 1 byte increments
- Page / frame size: 2^n bytes
- Number of pages: 2^(m-n)
- Large page will cause internal fragmentation
- Small page will incur large page table … 😒

Page table size



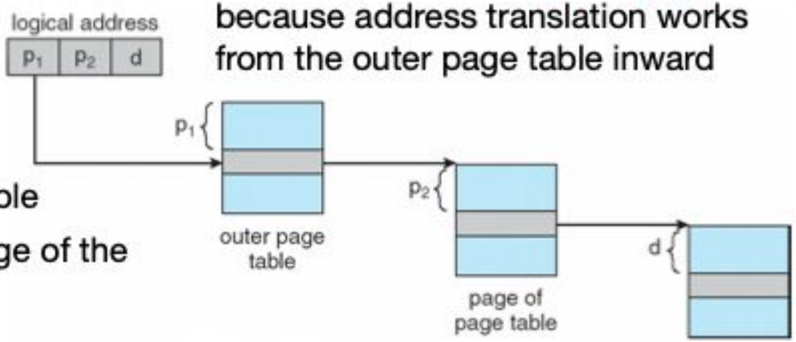
- Example:
  - 32-bit address space
  - 4KB (2^12 byte) page size
  - 4-byte page-table entry
- Number of page table entries: 2^32 / 2^12 = 2^20
- Page table size: 2^20 * 4 bytes = 4MB (per process)
- My laptop currently has 499 running processes
- RAM for storing page tables: ~2GB
- How to reduce page table size?

Multi-level page tables



- Look up the outer page table to get the frame number for the inner page table
- Not all inner page tables need to be allocated at first - save space!
- Aliases
  - Outer page table - page directory
  - Inner page table - page table

known as forward-mapped page table because address translation works from the outer page table inward



p_1 is an index into the outer page table

p_2 is the displacement within the page of the inner page table

d is the page offset

Multi-level page tables - have a try

- Download the example from eClass
- Go to **HW-Paging-MultiLevelTranslate** folder
- Run **paging-multilevel-translate.py** to get example page directory, page table, memory dump and some VAs
- Try to locate the page table for each VA (by looking up the page directory), translate VA to PA (by looking up the page table), and check your answers

Page replacement policy

- RAM are fast but small
    - Physical memory < memory used by processes
    - Virtual memory: move some pages to the disk
- Which one to replace? (cache replacement policies)
    - First in first out (FIFIO)
    - Least recently used (LRU)
    - Least frequently used (LFU)
    - Most recently used (MRU)
    - etc.

Page replacement policy - have a try

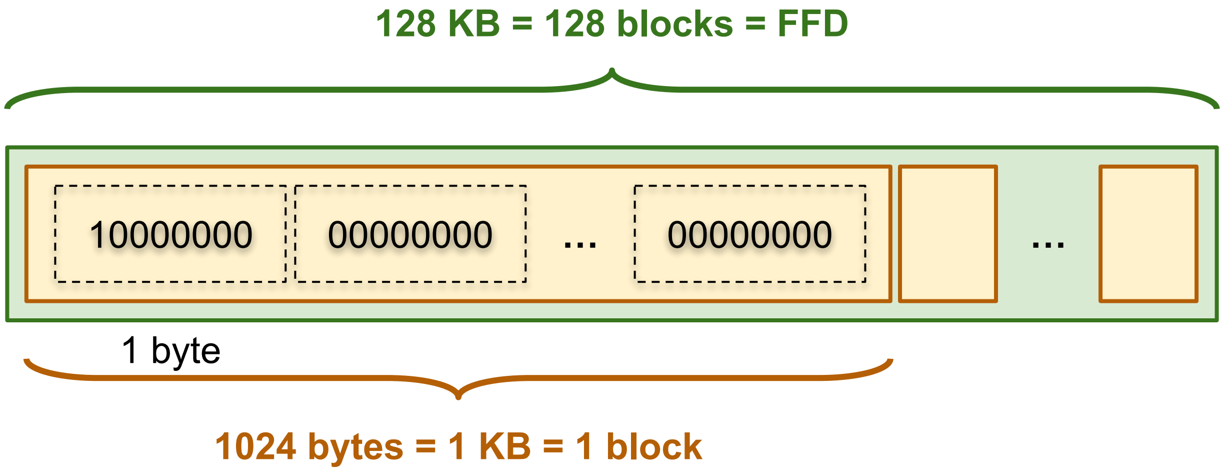
- Download the example from eClass
- Try **./paging-policy.py --addresses=0,1,2,0,1,3,0,3,1,2,1 --policy=LRU --cachesize=3 -c** with different policies, and compare the number of misses

UNIX-like File System

Assignment 3
Implementing a Unix-like FS

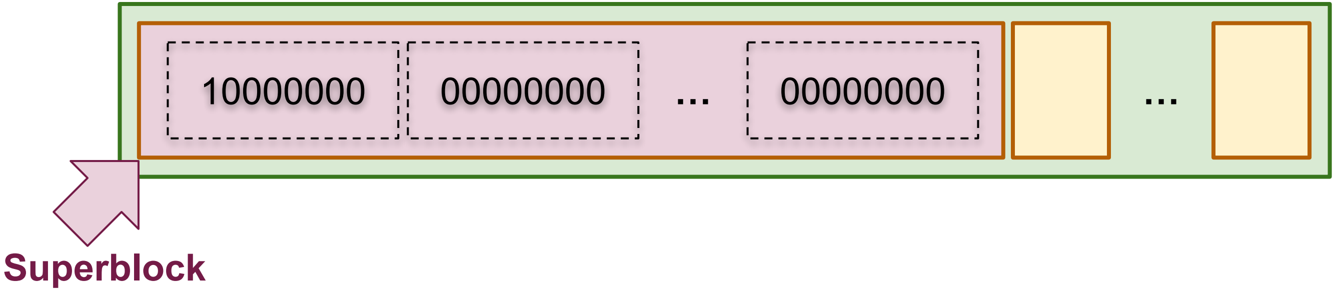
File System Simulator (FS-Sim)

- Our toy **FFD** (fake file disk) has a capacity of **128 KB**, which contains only one partition and one simulated file system.
- **Blocks** on our simulated file system are **1024 bytes (or 1 KB)** in length.



File System Simulator (FS-Sim)

- The first block is the **superblock**, that stores free-space list and inodes.

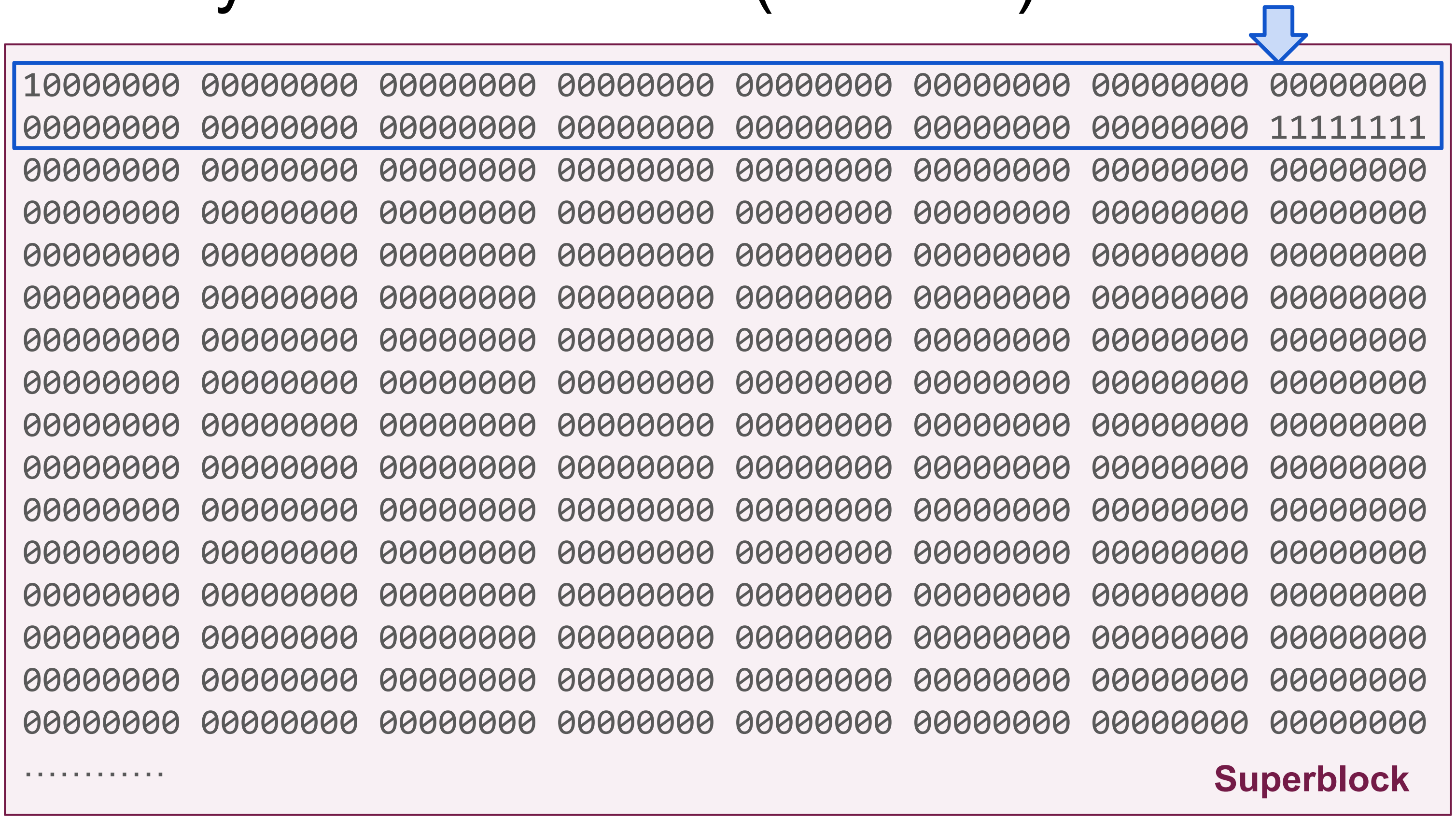


File System Simulator (FS-Sim)

```
10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    1 -    8 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 11111111    9 -   16 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   17 -   24 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   25 -   32 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   33 -   40 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   41 -   48 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   49 -   56 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   57 -   64 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   65 -   72 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   73 -   80 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   81 -   88 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   89 -   96 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   97 -  104 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  105 -  112 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  113 -  120 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  121 -  128 bytes
............                                                             128 -1024 bytes
```

Superblock

File System Simulator (FS-Sim)

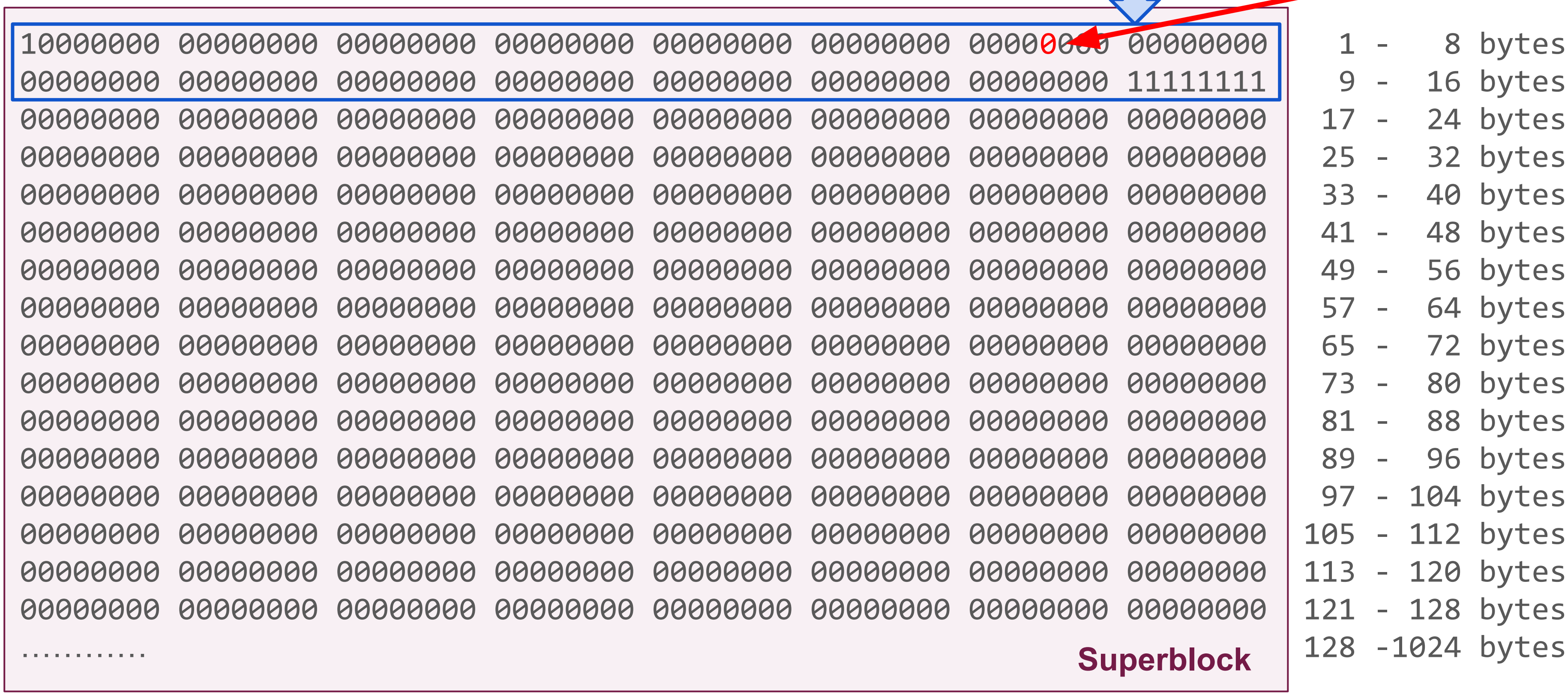


Free-block list

```
10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     1 -    8 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 11111111     9 -   16 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    17 -   24 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    25 -   32 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    33 -   40 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    41 -   48 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    49 -   56 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    57 -   64 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    65 -   72 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    73 -   80 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    81 -   88 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    89 -   96 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    97 -  104 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   105 -  112 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   113 -  120 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   121 -  128 bytes
............                                                              128 -1024 bytes
```

Superblock

File System Simulator (FS-Sim)



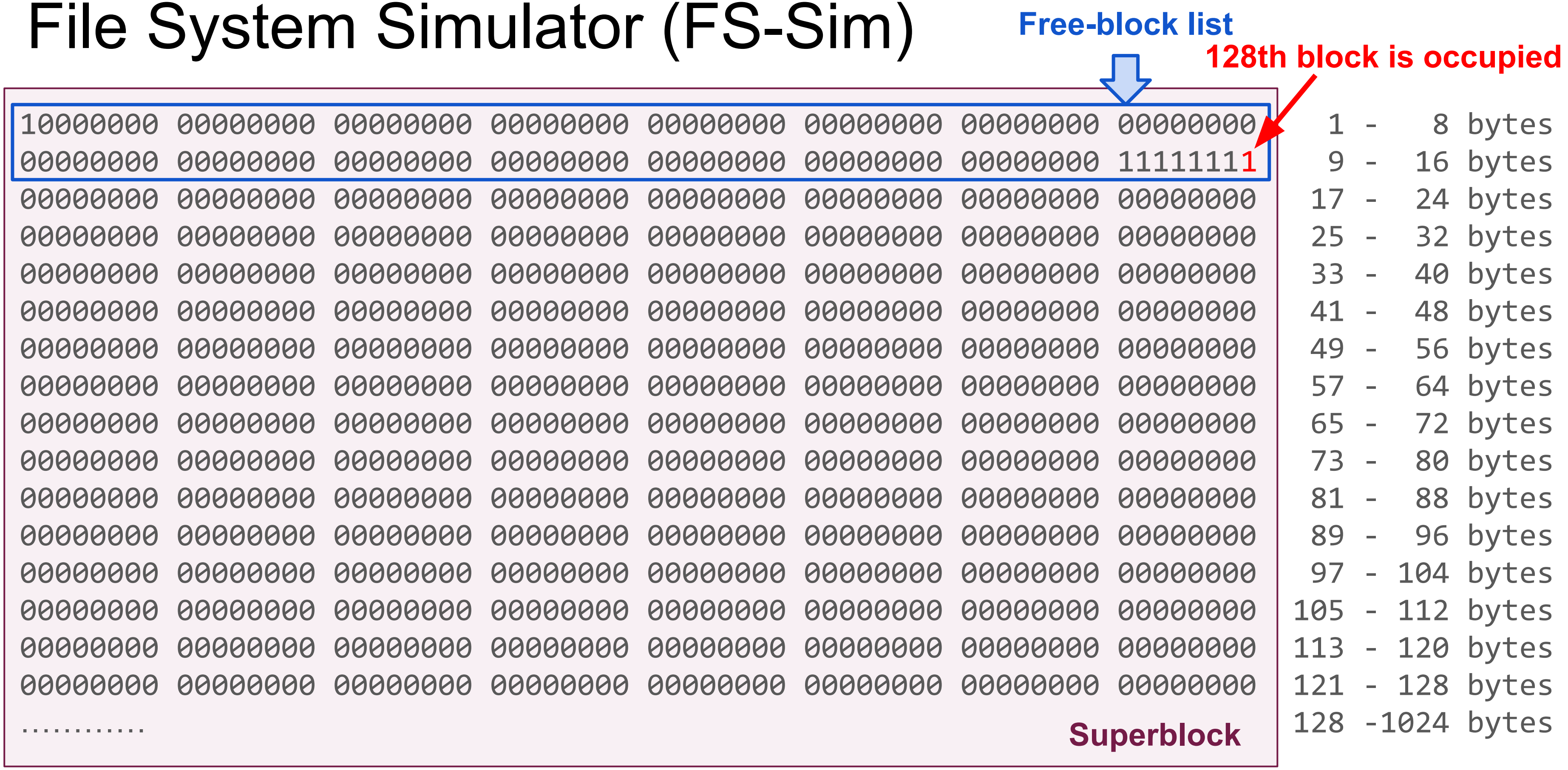
Free-block list

53rd block is available

```
10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000      1 -   8 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 11111111      9 -  16 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     17 -  24 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     25 -  32 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     33 -  40 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     41 -  48 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     49 -  56 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     57 -  64 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     65 -  72 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     73 -  80 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     81 -  88 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     89 -  96 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     97 - 104 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    105 - 112 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    113 - 120 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    121 - 128 bytes
............                                                               128 -1024 bytes
```

Superblock

File System Simulator (FS-Sim)



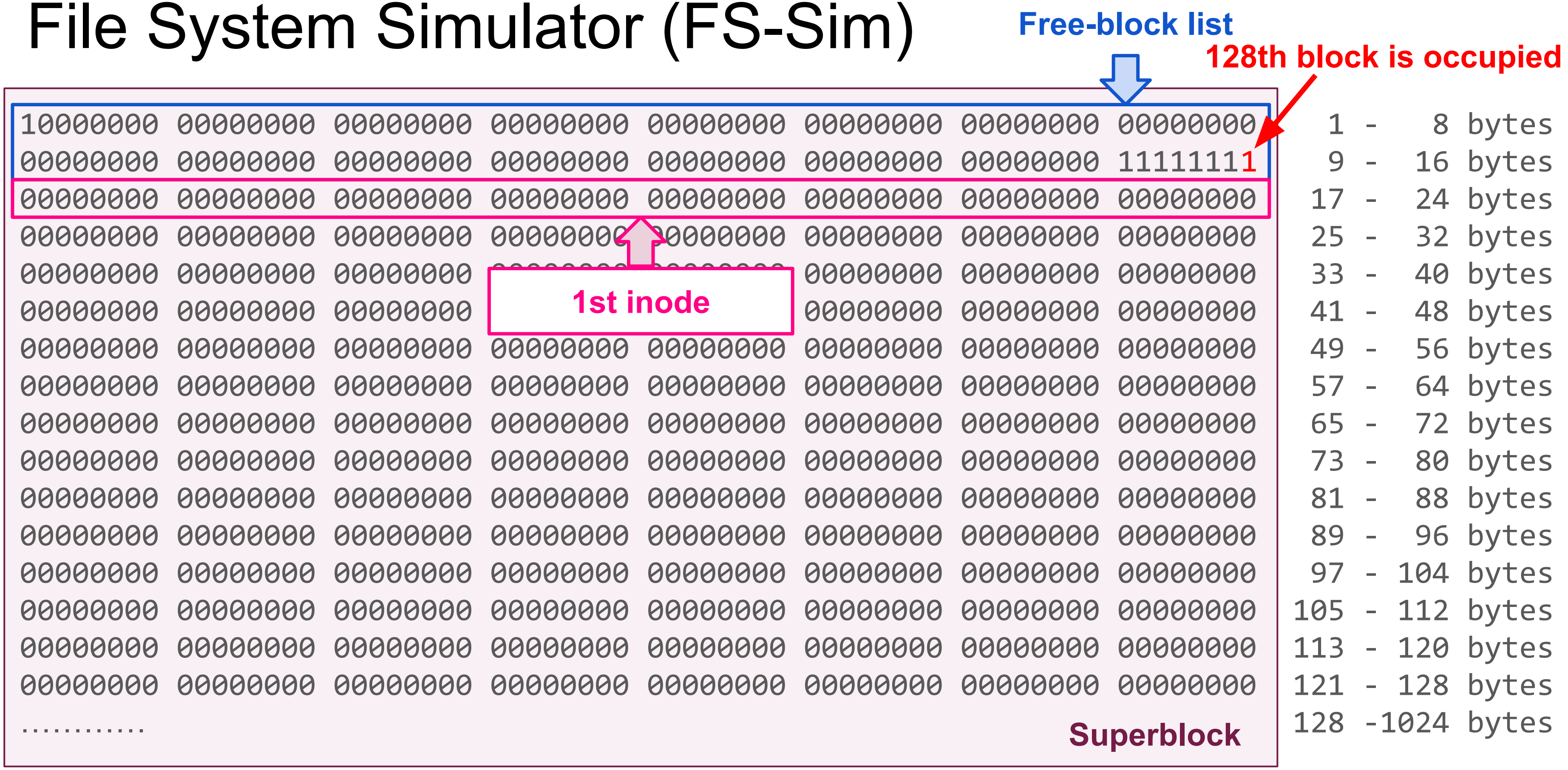
Free-block list

128th block is occupied

```
10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     1 -    8 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 11111111     9 -   16 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    17 -   24 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    25 -   32 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    33 -   40 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    41 -   48 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    49 -   56 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    57 -   64 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    65 -   72 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    73 -   80 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    81 -   88 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    89 -   96 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    97 -  104 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   105 -  112 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   113 -  120 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   121 -  128 bytes
............                                                              128 -1024 bytes
```

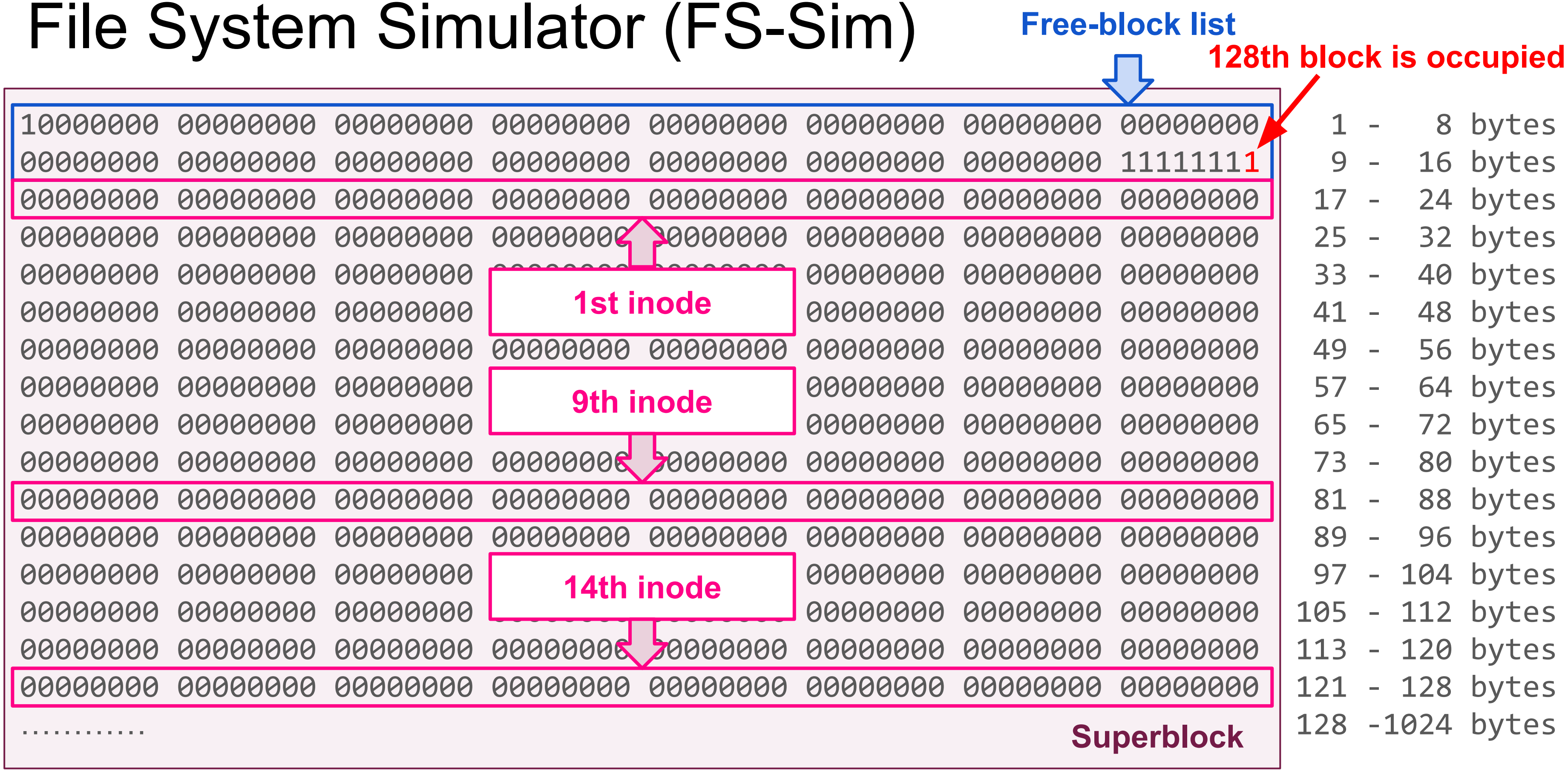
Superblock

File System Simulator (FS-Sim)



```
10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     1 -    8 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 11111111     9 -   16 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    17 -   24 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    25 -   32 bytes
00000000 00000000 00000000                   00000000 00000000 00000000    33 -   40 bytes
00000000 00000000 00000000                   00000000 00000000 00000000    41 -   48 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    49 -   56 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    57 -   64 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    65 -   72 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    73 -   80 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    81 -   88 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    89 -   96 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    97 -  104 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   105 -  112 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   113 -  120 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   121 -  128 bytes
............                                                              128 -1024 bytes
```

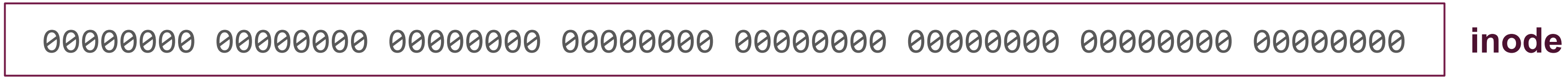
File System Simulator (FS-Sim)



```
10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000     1 -    8 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 11111111     9 -   16 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    17 -   24 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    25 -   32 bytes
00000000 00000000 00000000                   00000000 00000000 00000000    33 -   40 bytes
00000000 00000000 00000000                   00000000 00000000 00000000    41 -   48 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    49 -   56 bytes
00000000 00000000 00000000                   00000000 00000000 00000000    57 -   64 bytes
00000000 00000000 00000000                   00000000 00000000 00000000    65 -   72 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    73 -   80 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    81 -   88 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    89 -   96 bytes
00000000 00000000 00000000                   00000000 00000000 00000000    97 -  104 bytes
00000000 00000000 00000000                   00000000 00000000 00000000   105 -  112 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   113 -  120 bytes
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   121 -  128 bytes
............                                                              128 - 1024 bytes
```

File System Simulator (FS-Sim)

- The **inode** (short for index node) contains the metadata for each file or directory in the file system, such as name, size, etc.

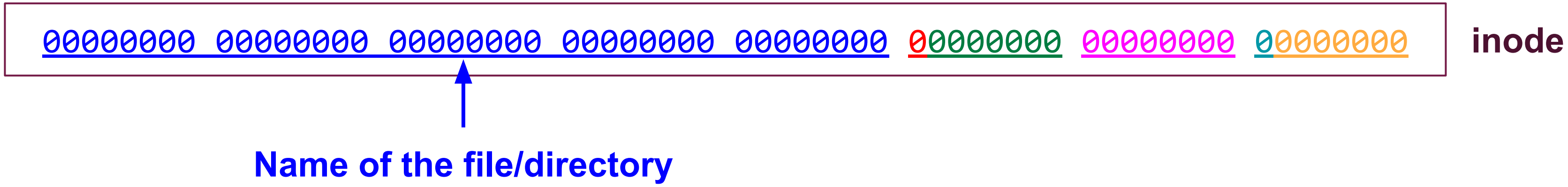


```
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

inode

File System Simulator (FS-Sim)

- The **inode** (short for index node) contains the metadata for each file or directory in the file system, such as name, size, etc.



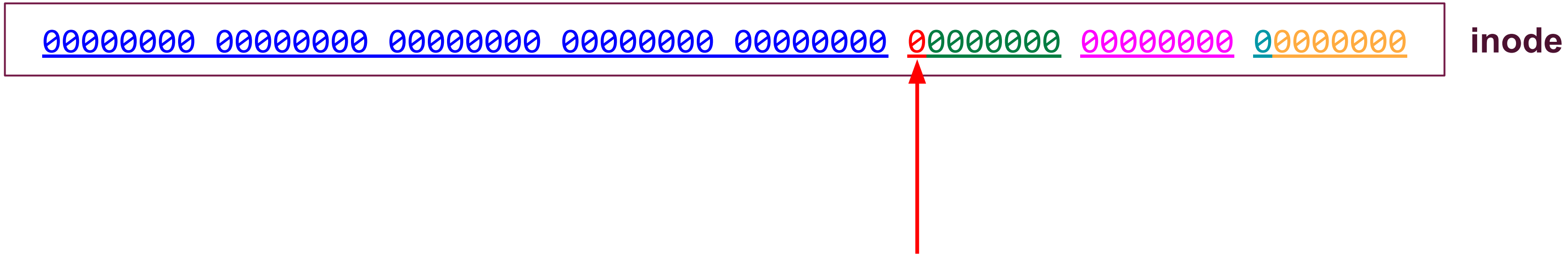
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

inode

Name of the file/directory

File System Simulator (FS-Sim)

- The **inode** (short for index node) contains the metadata for each file or directory in the file system, such as name, size, etc.

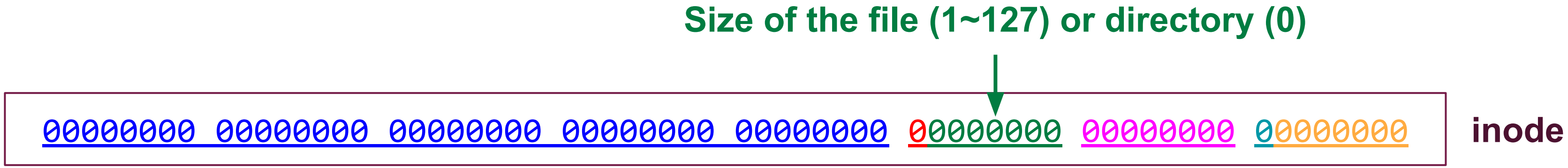


00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 **inode**

This inode is in use (1) or free (0)

File System Simulator (FS-Sim)

- The **inode** (short for index node) contains the metadata for each file or directory in the file system, such as name, size, etc.



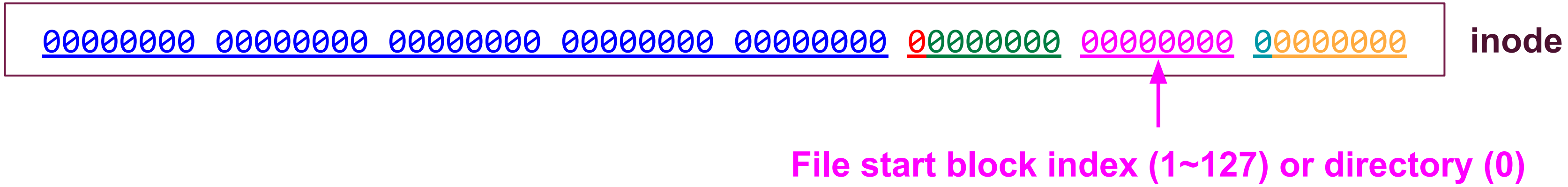
Size of the file (1~127) or directory (0)

```
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

inode

File System Simulator (FS-Sim)

- The **inode** (short for index node) contains the metadata for each file or directory in the file system, such as name, size, etc.



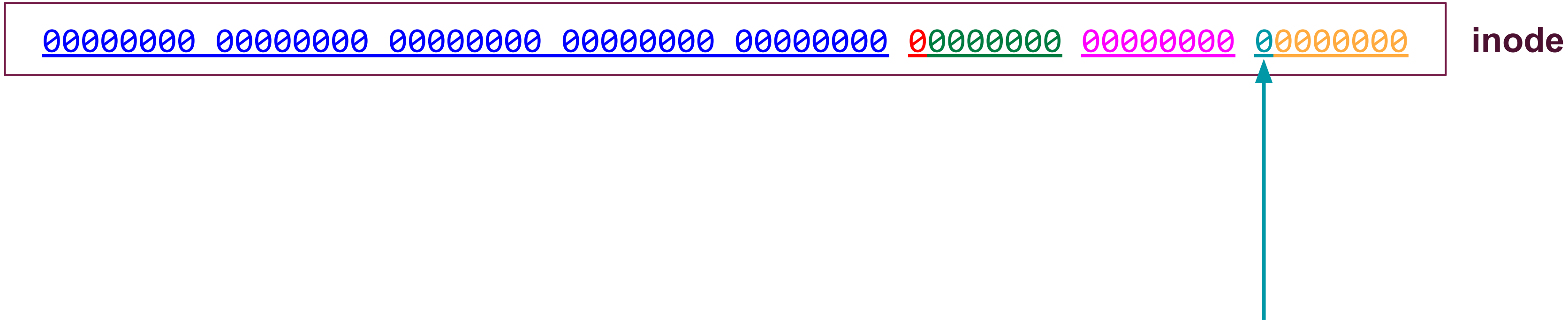
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

inode

File start block index (1~127) or directory (0)

File System Simulator (FS-Sim)

- The **inode** (short for index node) contains the metadata for each file or directory in the file system, such as name, size, etc.



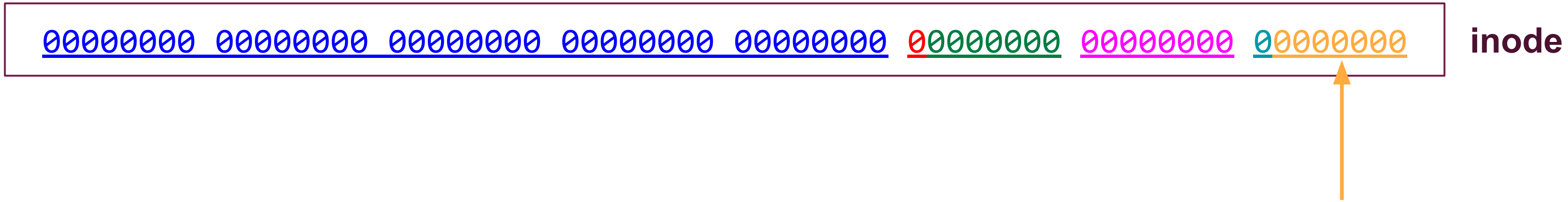
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

inode

This inode is directory (1) or file (0)

File System Simulator (FS-Sim)

- The **inode** (short for index node) contains the metadata for each file or directory in the file system, such as name, size, etc.



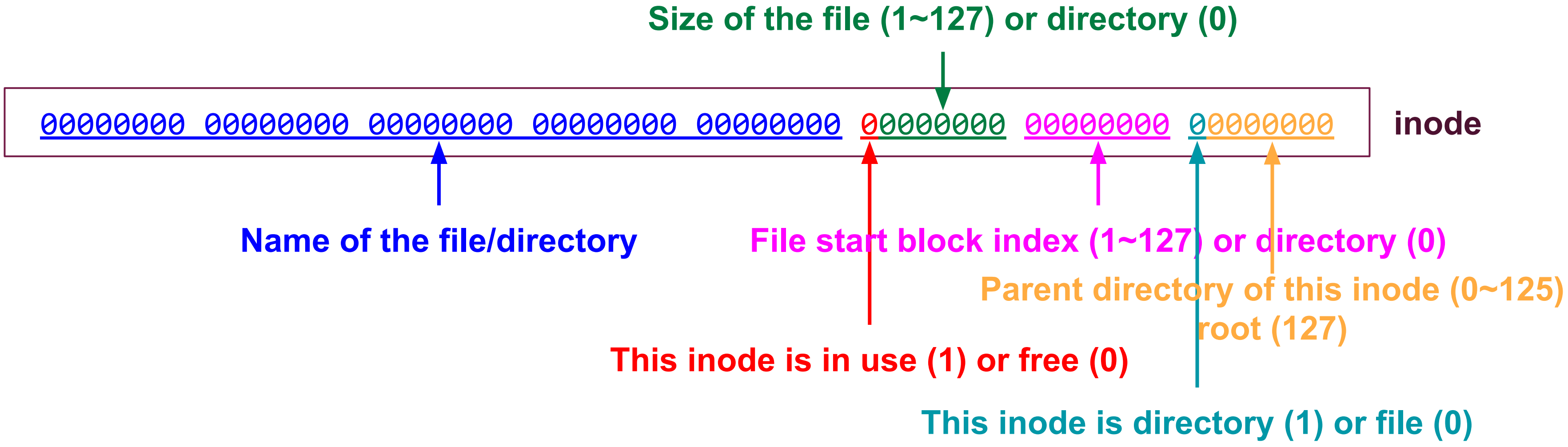
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

inode

Parent directory of this inode (0~125)
root (127)

File System Simulator (FS-Sim)

- The **inode** (short for index node) contains the metadata for each file or directory in the file system, such as name, size, etc.



Size of the file (1~127) or directory (0)

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

inode

Name of the file/directory

File start block index (1~127) or directory (0)

Parent directory of this inode (0~125)
root (127)

This inode is in use (1) or free (0)

This inode is directory (1) or file (0)

File System Simulator (FS-Sim)

- The first block is the **superblock**, that stores free-space list and inodes.
- **Free-space list** occupy the first 128 bits (16 bytes) in the **superblock**, each bit represents the corresponding block is available or occupied.
- Each **inode** (short for index node) occupy 8 bytes in the **superblock**. We have **126 inodes** in the superblock, which occupy 126 × 8 = 1008 bytes in the **superblock** in total.

In superblock: 16 bytes (free-block list) + 1008 bytes (inodes) = 1024 bytes = 1 KB

Your program also have a buffer, which is 1 KB. Zero-out it at the beginning.

File System Simulator (FS-Sim)

```c
void fs_mount(char *new_disk_name);
void fs_create(char name[5], int size);
void fs_delete(char name[5]);
void fs_read(char name[5], int block_num);
void fs_write(char name[5], int block_num);
void fs_buff(uint8_t buff[1024]);
void fs_ls(void);
void fs_resize(char name[5], int new_size);
void fs_defrag(void);
void fs_cd(char name[5]);
```

File System Simulator (FS-Sim)

```
void fs_mount(char *new_disk_name);
```

- Mount FFD to your program. Remember we are mounting FFD with our simulated FS, so we are not mounting a real disk with real FS, which means you should not use system call “mount”.
- Do not forget to check the consistency of the file system. In this assignment, we only care about 6 potential errors in the FFD caused by unknown reasons. You need to check them one by one and follow the given order.
- If the FFD passes the test, then mount it into the program by reading the superblock of this FFD into the memory, and unmount the previous FFD. Otherwise, keep using the last FFD.
- A success fs_mount will change the current working directory to root.

File System Simulator (FS-Sim)

```
void fs_create(char name[5], int size);
```

- Create a file on current mounted FFD with given name and a fixed size.
- When assign blocks to the file, the file must be allocated a number of contiguous blocks.
- A size of 0 means the user wants to create a directory.
- Remember that you cannot have two files with the same name under the same path.
- Name . and .. are reserved and cannot be used.
- In this assignment, user can have symbols in the name as well (such as ...). We will only test the characters that is on your keyboard (i.e. will not test ‘\n’, ‘\r’, etc.).

File System Simulator (FS-Sim)

```
void fs_delete(char name[5]);
```

- Delete the file on mounted FFD by its name.
- If a directory is selected, then remove the directory and all files/directories in it.

File System Simulator (FS-Sim)

```
void fs_read(char name[5], int block_num);
```

- Put 1 KB data in the specified block to buffer.

```
void fs_write(char name[5], int block_num);
```

- Put 1 KB data in buffer to the specified block.

```
void fs_buff(uint8_t buff[1024]);
```

- Put user input to buffer.

File System Simulator (FS-Sim)

```
void fs_ls(void);
```

- Same thing as when you type ls in your terminal.
- List files and directories based on the order they stored in inodes.
- Format in description.

File System Simulator (FS-Sim)

```
void fs_resize(char name[5], int new_size);
```

- Change the size of the file into new_size.
- If you have to shift the file, remember to clean-up data in the old position. We do not want any dummy data in your FFD.

File System Simulator (FS-Sim)

```
void fs_defrag(void);
```

- Organize the contents of all files in FFD.
- You should shift all files to the lowest start block index, in the meantime maintain the order of those files (i.e., if file1 locate physically after file4, then after defragmentation, file1 should still physically after file4).

File System Simulator (FS-Sim)

```
void fs_cd(char name[5]);
```

- Change the current working directory into the new directory.
- Same as cd in your terminal.

File System Simulator (FS-Sim)

General comments:

1. Concepts are relatively easy. This is a coding assignment, and the implementation is straight forward.
2. Test your code with errors. We listed all errors you need to handle, and we assigned priorities to those errors.
3. You are encouraged to make your test cases and submit them.
4. Start early.