Operating Systems Practice

Project #3 – Advanced xv6 File system

Soyoung Han

(zsally@dgist.ac.kr)

Before we start…

- Clean-up your Git repository for this project
  - `git fetch && git reset --hard && git clean -fdx && git checkout fs`
    - Your modifications will be deleted with this command!

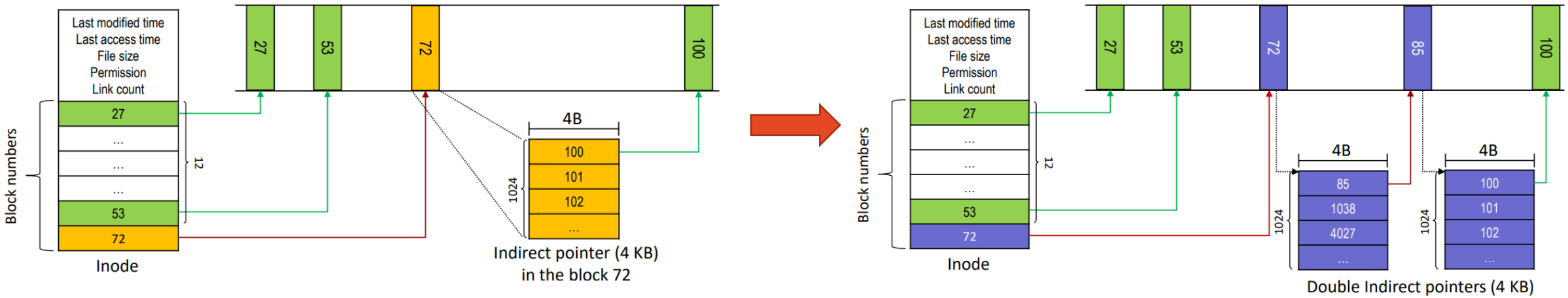
- If an error occurs after **usertests/big/integrity** tests, just do '**make clean**' before make qemu-nox -j

Overview – Advanced xv6 File system

- In the lecture, we learned about a very simple file system…
- The goal is to understand the file system structure and improve the xv6 file system (especially the inode and block pointer)

Overview – Advanced xv6 File system

- In the lecture, we learned about a very simple file system…
- The goal is to understand the file system structure and improve the xv6 file system (especially the inode and block pointer)



Overview – Advanced xv6 File system

- In the lecture, we learned about a very simple file system…
- The goal is to understand the file system structure and improve the xv6 file system (especially the inode and block pointer)

	Before	After
Max # of files	< 5000	> 500,000
Max file size	70 KB	10 MB

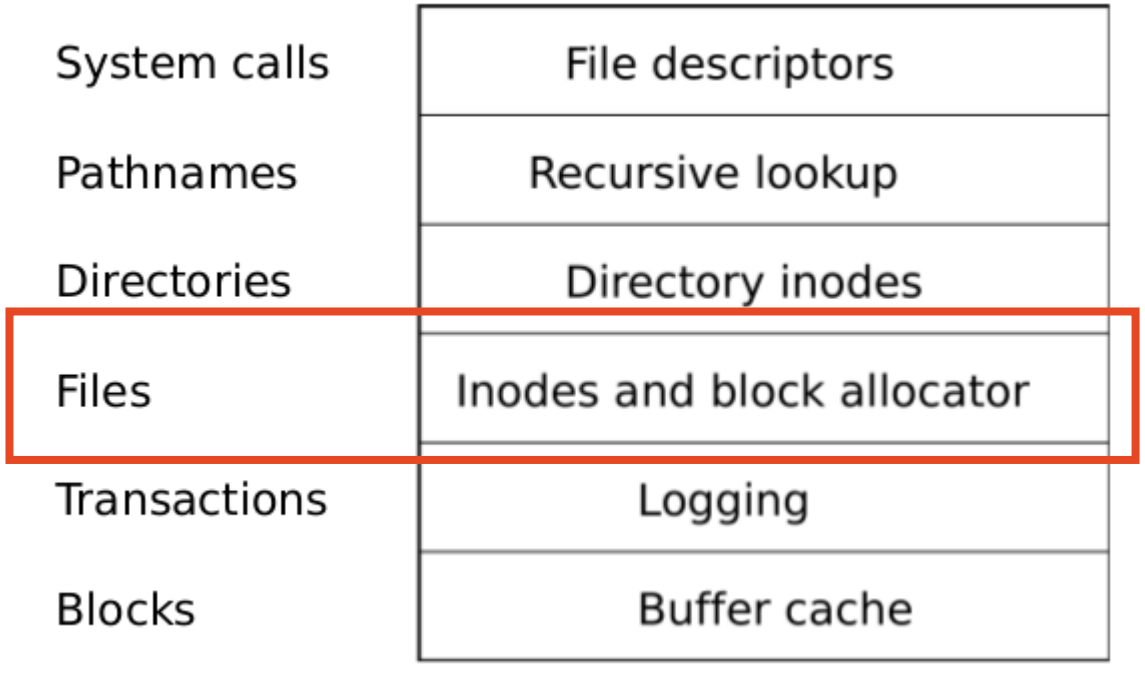
Xv6 – File system

- Layers of the xv6 file system

System calls	File descriptors
Pathnames	Recursive lookup
Directories	Directory inodes
Files	Inodes and block allocator
Transactions	Logging
Blocks	Buffer cache

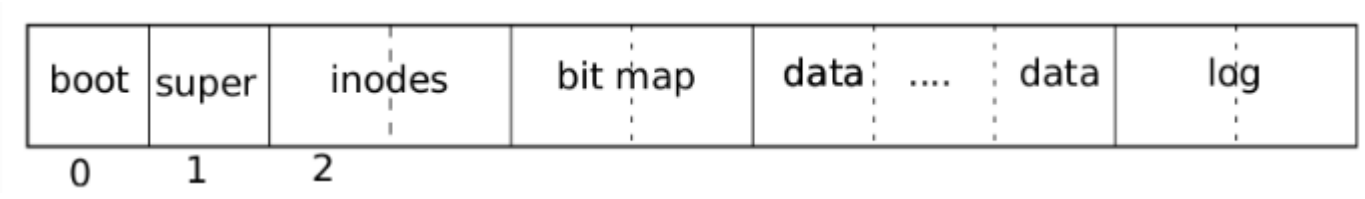
Xv6 – File system

- Layers of the xv6 file system

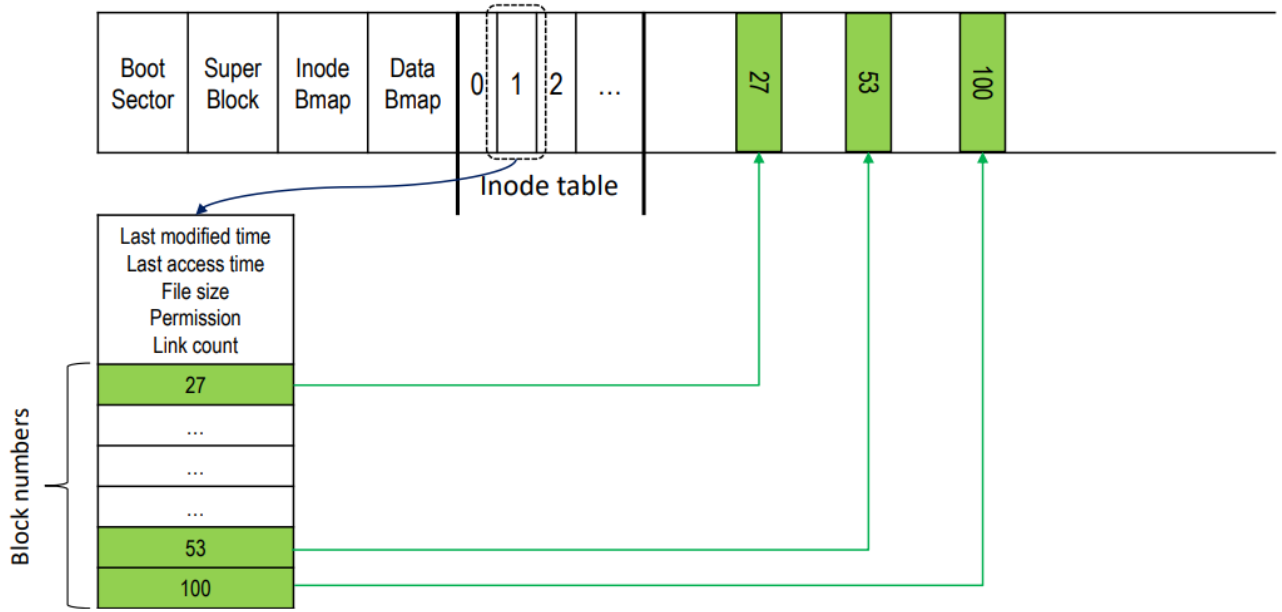


Xv6 – File system

- Structure of the xv6 file system

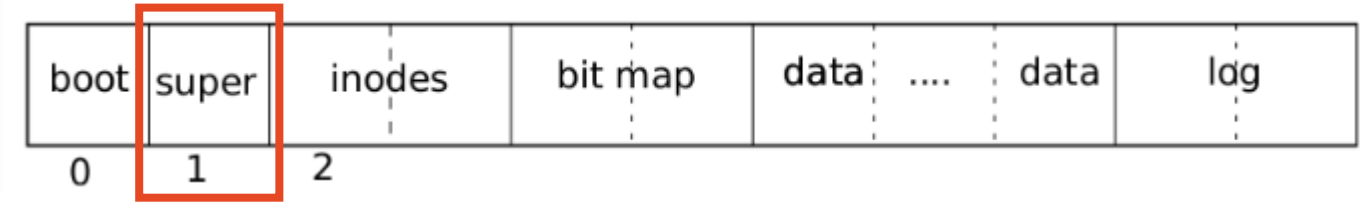


- (In the lecture) Structure of simple file system



Xv6 – File system

- Structure of the xv6 file system

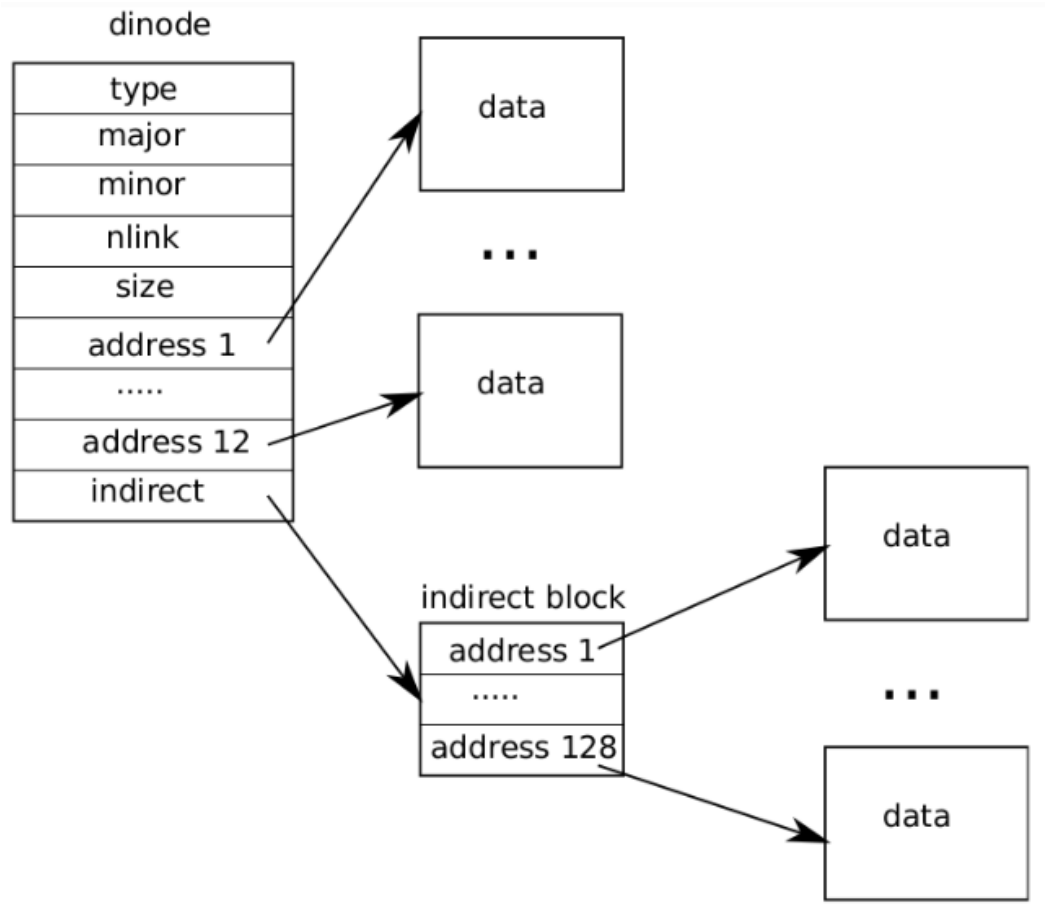


- The header fs.h contains *constants* and *data structures* describing the exact layout of the file system

```
struct superblock {
  uint size;         // Size of file system image (blocks)
  uint nblocks;      // Number of data blocks
  uint ninodes;      // Number of inodes.
  uint nlog;         // Number of log blocks
  uint logstart;     // Block number of first log block
  uint inodestart;   // Block number of first inode block
  uint bmapstart;    // Block number of first free map block
};
```

Xv6 – File system

- The representation of a file on disk



fs.h

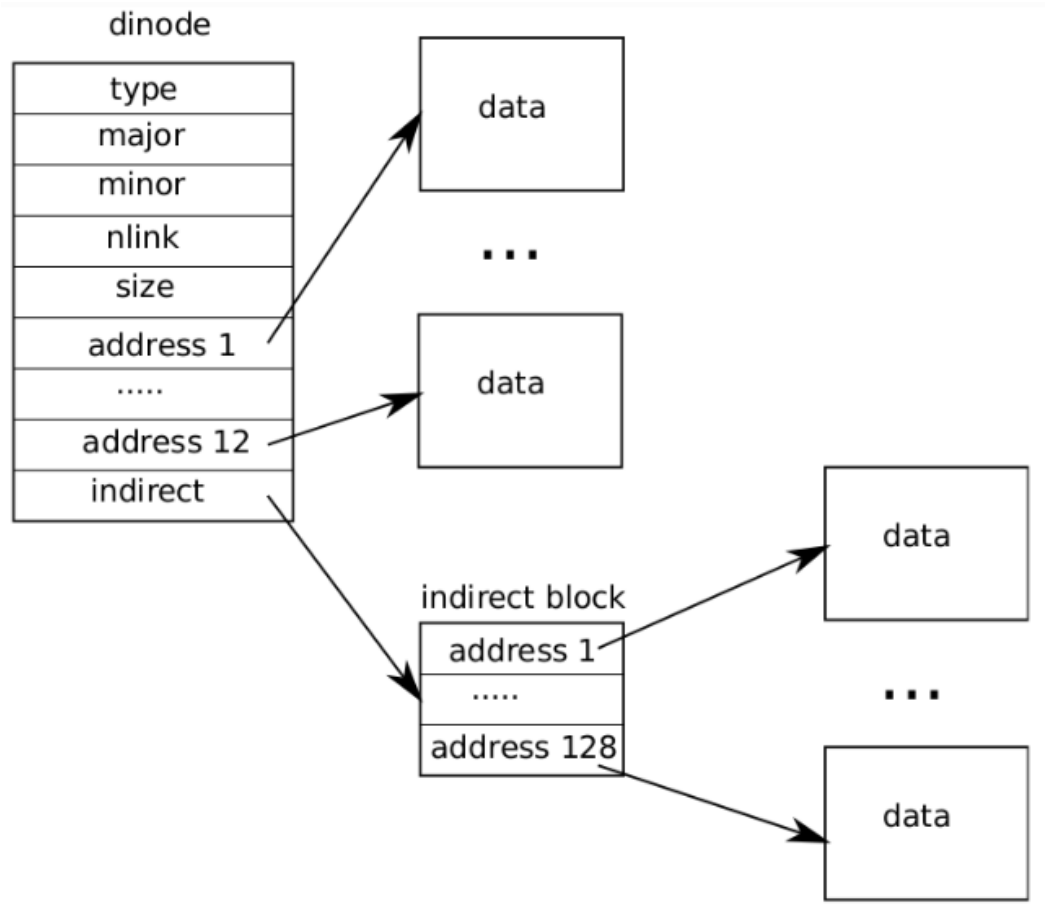
```
#define NDIRECT 12
#define NINDIRECT (BSIZE / sizeof(uint))
#define MAXFILE (NDIRECT + NINDIRECT)

// On-disk inode structure
struct dinode {
  short type;           // File type
  short major;          // Major device number (T_DEV only)
  short minor;          // Minor device number (T_DEV only)
  short nlink;          // Number of links to inode in file system
  uint size;            // Size of file (bytes)
  uint addrs[NDIRECT+1];   // Data block addresses
};

// Inodes per block.
#define IPB           (BSIZE / sizeof(struct dinode))
```

Xv6 – File system

- The representation of a file on disk



fs.h

```
#define NDIRECT 12
#define NINDIRECT (BSIZE / sizeof(uint))
#define MAXFILE (NDIRECT + NINDIRECT)
```

512 / 4 = 128

12 + 128 = 140

Max numbers of block per inode

```
// On-disk inode structure
struct dinode {
  short type;           // File type
  short major;          // Major device number (T_DEV only)
  short minor;          // Minor device number (T_DEV only)
  short nlink;          // Number of links to inode in file system
  uint size;            // Size of file (bytes)
  uint addrs[NDIRECT+1];   // Data block addresses
};

// Inodes per block.
#define IPB           (BSIZE / sizeof(struct dinode))
```

Xv6 – File system: Problem 1

- Max numbers of block per inode(file) = 140 blocks
- Max file size = 140 blocks × block size= 140 × 512 bytes = 70 KB
- Max number of files = 140 blocks × ?

Xv6 – File system: Problem 1

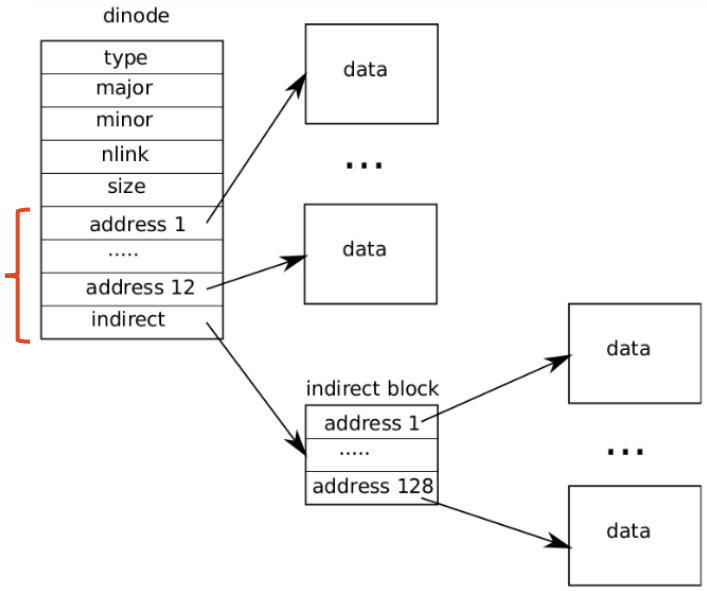
- Max numbers of block per inode(file) = 140 blocks
- Max file size = 140 blocks × block size= 140 × 512 bytes = 70 KB
- Max number of files = 140 blocks × ?

- **< Problem 1>**
  - **Calculate the (max number of files) from an existing xv6 file system**
  - Please refer to the fs.h file
  - Keep in mind that the number of files is related to the directory
  - Submit the calculation process and answer in a **report** (briefly)

Xv6 – File system: Problem 2

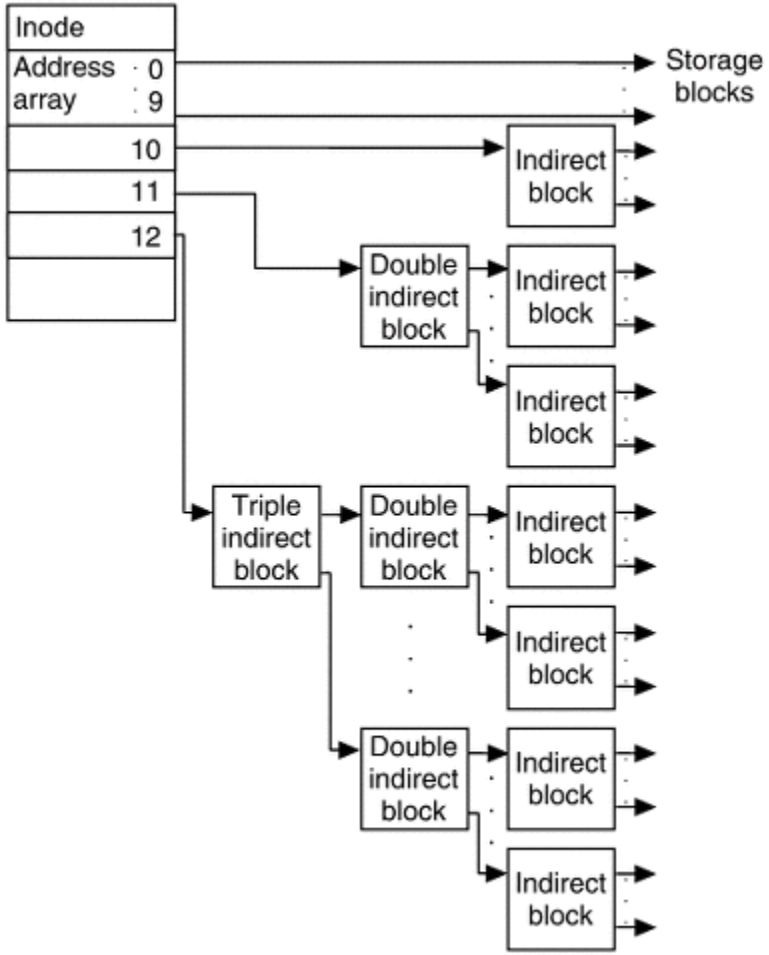
- Max numbers of block per inode(file) = 140 blocks
- Max file size = 140 blocks × block size= 140 × 512 bytes = 70 KB < It's too small!
  - One indirect block is not enough to support large files

- **<Problem 2>**
  - **Make sure that the maximum size of the file is at least 10 MB**
  - The size of <struct> dinode should not be changed
  - You can implement double indirect block or triple indirect block



Xv6 – File system: Problem 2

- Double Indirect Block & Triple Indirect Block



Xv6 – File system: Problem 2

- **<Problem 2>**
  - **Make sure that the maximum size of the file is at least 10 MB**
  - The sizeof(struct dinode) should not be changed
  - You can implement double indirect block or triple indirect block
    - Triple indirect block could take a long time to test ☹
  - You will need to modify these files
    - param.h
    - file.h
    - fs.h
    - fs.c
    - mkfs.c

Xv6 – File system: Problem 2

- **<Problem 2>**
  - **Make sure that the maximum size of the file is at least 10 MB**
  - The sizeof(struct dinode) should not be changed
  - You can implement double indirect block or triple indirect block
    - Triple indirect block could take a long time to test ☹
  - You will need to modify these files
    - param.h → `#define FSSIZE 1000`
    - file.h
    - fs.h
    - fs.c
    - mkfs.c

Xv6 – File system: Problem 2

- **<Problem 2>**
  - **Make sure that the maximum size of the file is at least 10 MB**
  - The sizeof(struct dinode) should not be changed
  - You can implement double indirect block or triple indirect block
    - Triple indirect block could take a long time to test ☹
  - You will need to modify these files
    - param.h → `#define FSSIZE` ~~1000~~ **At least 200000**
    - file.h
    - fs.h
    - fs.c
    - mkfs.c

Xv6 – File system: Problem 2

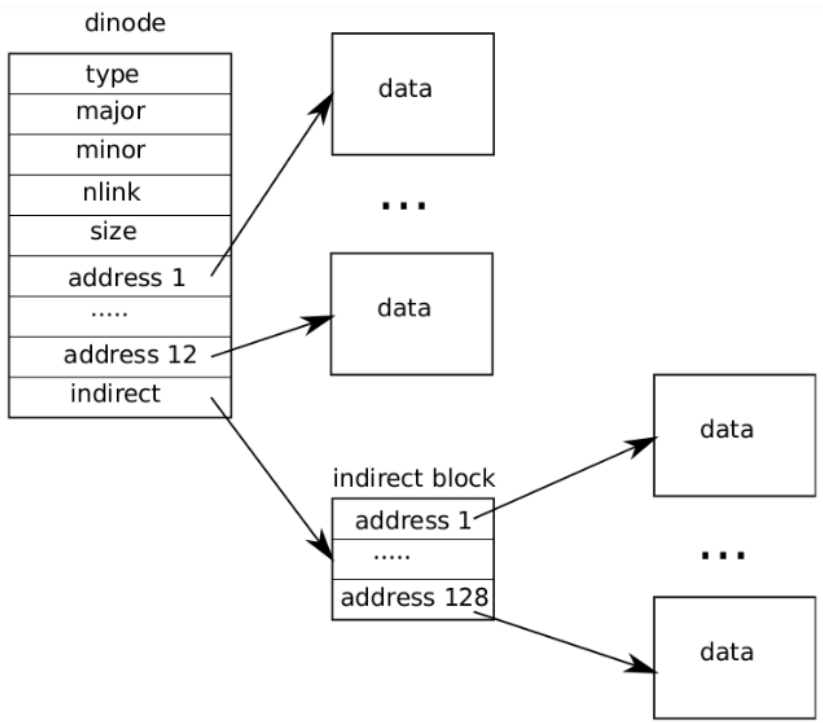
- **<Problem 2>**
  - **Make sure that the maximum size of the file is at least 10 MB**
  - The sizeof(struct dinode) should not be changed
  - You can implement double indirect block or triple indirect block
    - Triple indirect block could take a long time to test ☹
  - You will need to modify these files
    - param.h → `#define FSSIZE` ~~1000~~ **At least 200000**
    - file.h
    - fs.h
    - fs.c
    - mkfs.c

Modify the code at "To Do" / "Optional To Do"
Optional To Do: It may need to be modified according to your implementation method

Xv6 – File system: Problem 2

- **mkfs.c**
  - iappend(uint inum, void *xp, int n): Append inode at offset(*fbn*)
    - e.g., size of dinode.size = 7680 = 512 × 15 = 15 blocks

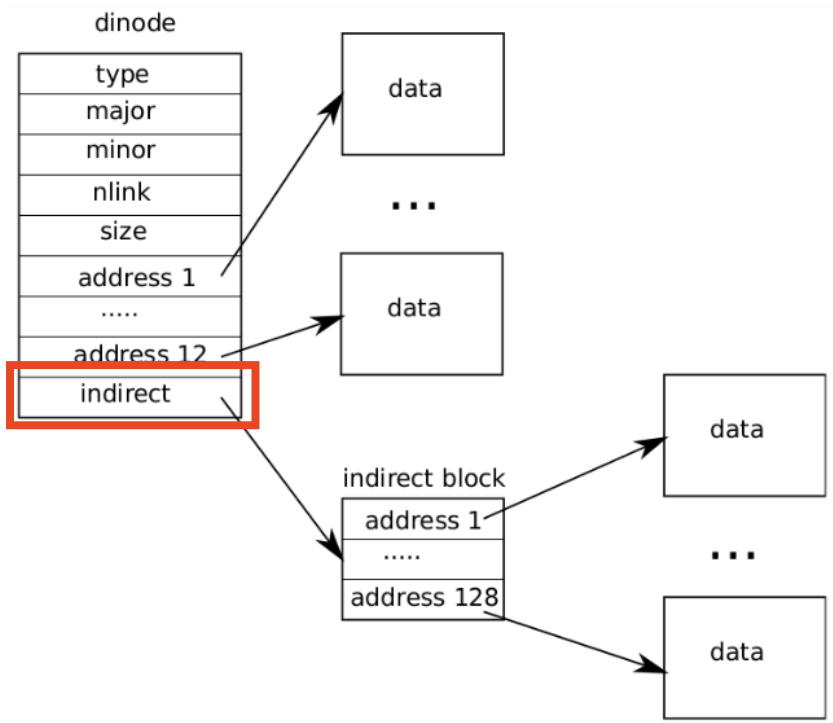
```
rinode(inum, &din);
off = xint(din.size);    off = 7680
// printf("append inum %d at off %d sz %d\n", inum, off, n);
while(n > 0){
  fbn = off / BSIZE;     fbn = 15
  assert(fbn < MAXFILE);
  if(fbn < NDIRECT){     15 > 12
    if(xint(din.addrs[fbn]) == 0){
      din.addrs[fbn] = xint(freeblock++);
    }
    x = xint(din.addrs[fbn]);
  } else {
    if(xint(din.addrs[NDIRECT]) == 0){
      din.addrs[NDIRECT] = xint(freeblock++);
    }
    rsect(xint(din.addrs[NDIRECT]), (char*)indirect);
    if(indirect[fbn - NDIRECT] == 0){
      indirect[fbn - NDIRECT] = xint(freeblock++);
      wsect(xint(din.addrs[NDIRECT]), (char*)indirect);
    }
    x = xint(indirect[fbn-NDIRECT]);
  }
```



Xv6 – File system: Problem 2

- **mkfs.c**
  - iappend(uint inum, void *xp, int n): Append inode at offset(*fbn*)
    - e.g., size of dinode.size = 7680 = 512 × 15 = 15 blocks

```
rinode(inum, &din);
off = xint(din.size);    off = 7680
// printf("append inum %d at off %d sz %d\n", inum, off, n);
while(n > 0){
  fbn = off / BSIZE;     fbn = 15
  assert(fbn < MAXFILE);
  if(fbn < NDIRECT){     15 > 12
    if(xint(din.addrs[fbn]) == 0){
      din.addrs[fbn] = xint(freeblock++);
    }
    x = xint(din.addrs[fbn]);
  } else {
    if(xint(din.addrs[NDIRECT]) == 0){
      din.addrs[NDIRECT] = xint(freeblock++);
    }
    rsect(xint(din.addrs[NDIRECT]), (char*)indirect);
    if(indirect[fbn - NDIRECT] == 0){
      indirect[fbn - NDIRECT] = xint(freeblock++);
      wsect(xint(din.addrs[NDIRECT]), (char*)indirect);
    }
    x = xint(indirect[fbn-NDIRECT]);
  }
```

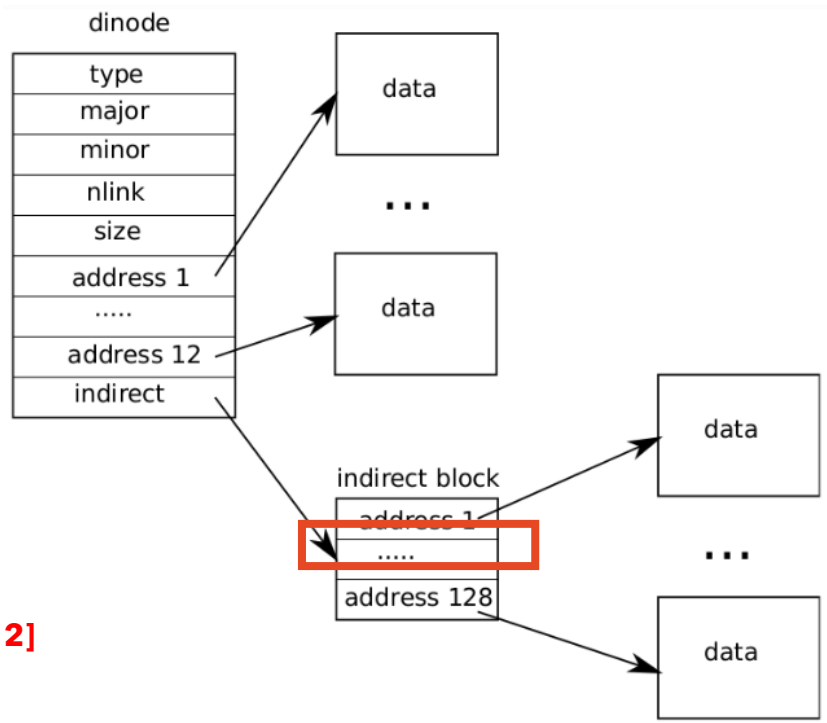


Xv6 – File system: Problem 2

- **mkfs.c**
  - iappend(uint inum, void *xp, int n): Append inode at offset(*fbn*)
    - e.g., size of dinode.size = 7680 = 512 × 15 = 15 blocks

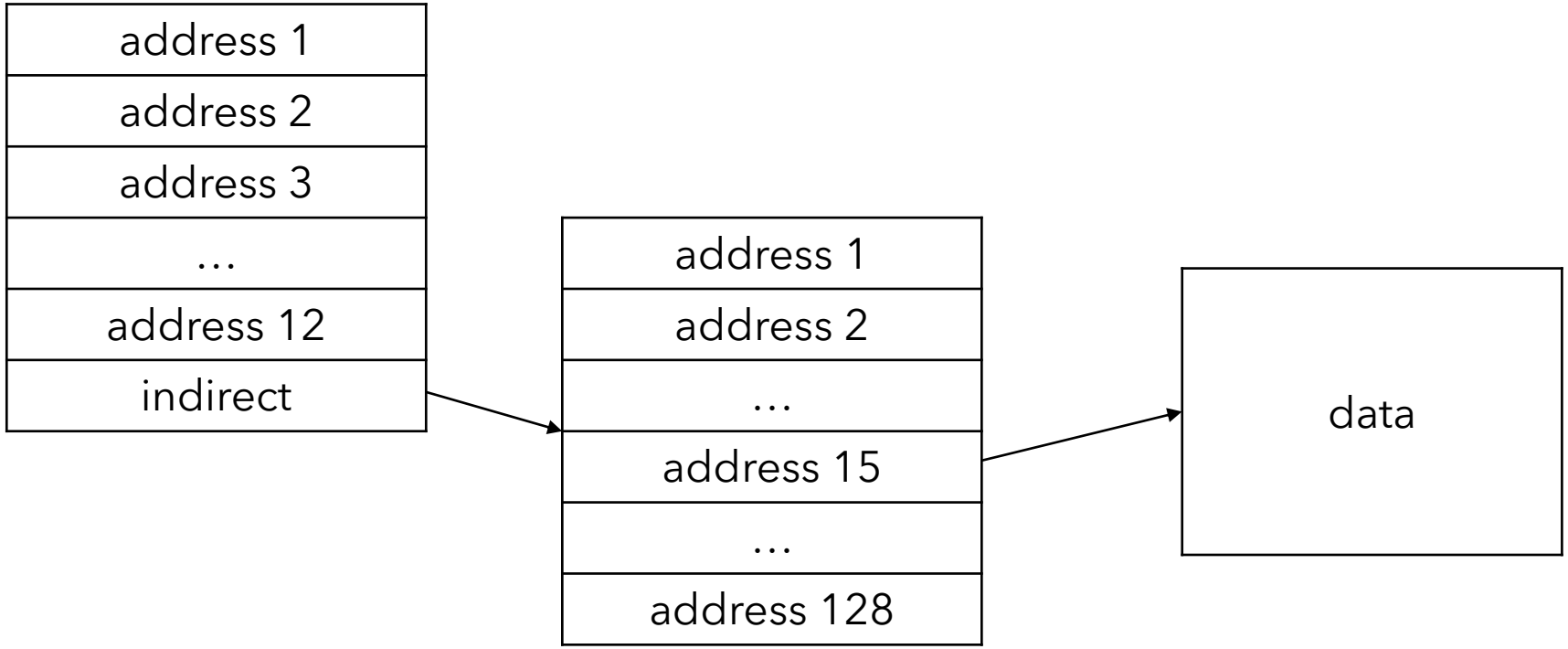
```
rinode(inum, &din);
off = xint(din.size);    off = 7680
// printf("append inum %d at off %d sz %d\n", inum, off, n);
while(n > 0){
  fbn = off / BSIZE;     fbn = 15
  assert(fbn < MAXFILE);
  if(fbn < NDIRECT){     15 > 12
    if(xint(din.addrs[fbn]) == 0){
      din.addrs[fbn] = xint(freeblock++);
    }
    x = xint(din.addrs[fbn]);
  } else {
    if(xint(din.addrs[NDIRECT]) == 0){
      din.addrs[NDIRECT] = xint(freeblock++);
    }
    rsect(xint(din.addrs[NDIRECT]), (char*)indirect);
    if(indirect[fbn - NDIRECT] == 0){
      indirect[fbn - NDIRECT] = xint(freeblock++);
      wsect(xint(din.addrs[NDIRECT]), (char*)indirect);
    }
    x = xint(indirect[fbn-NDIRECT]);
  }
```

Indirect[15-12]



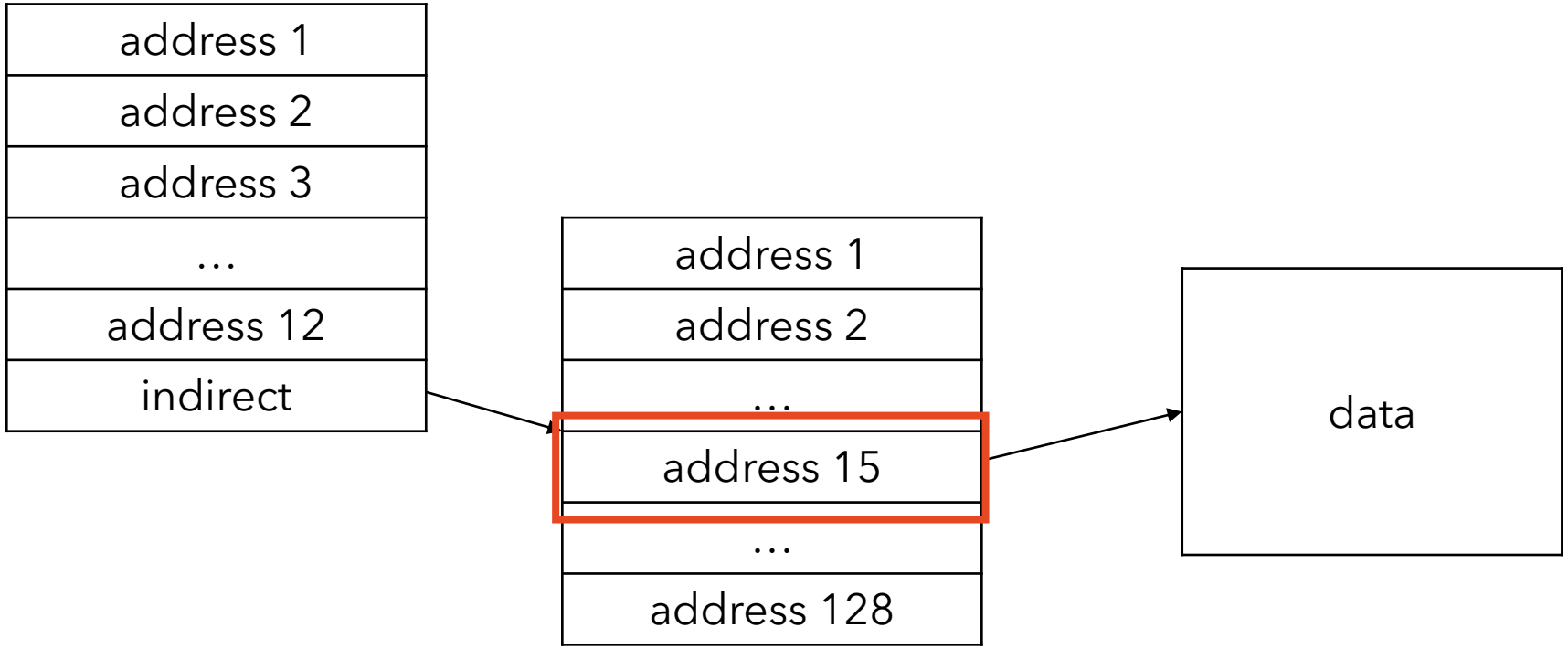
Xv6 – File system: Problem 2

- **fs.c**
  - bmap(struct inode *ip, uint n): Return the disk block address of the nth block in inode ip
    - e.g., if n = 27



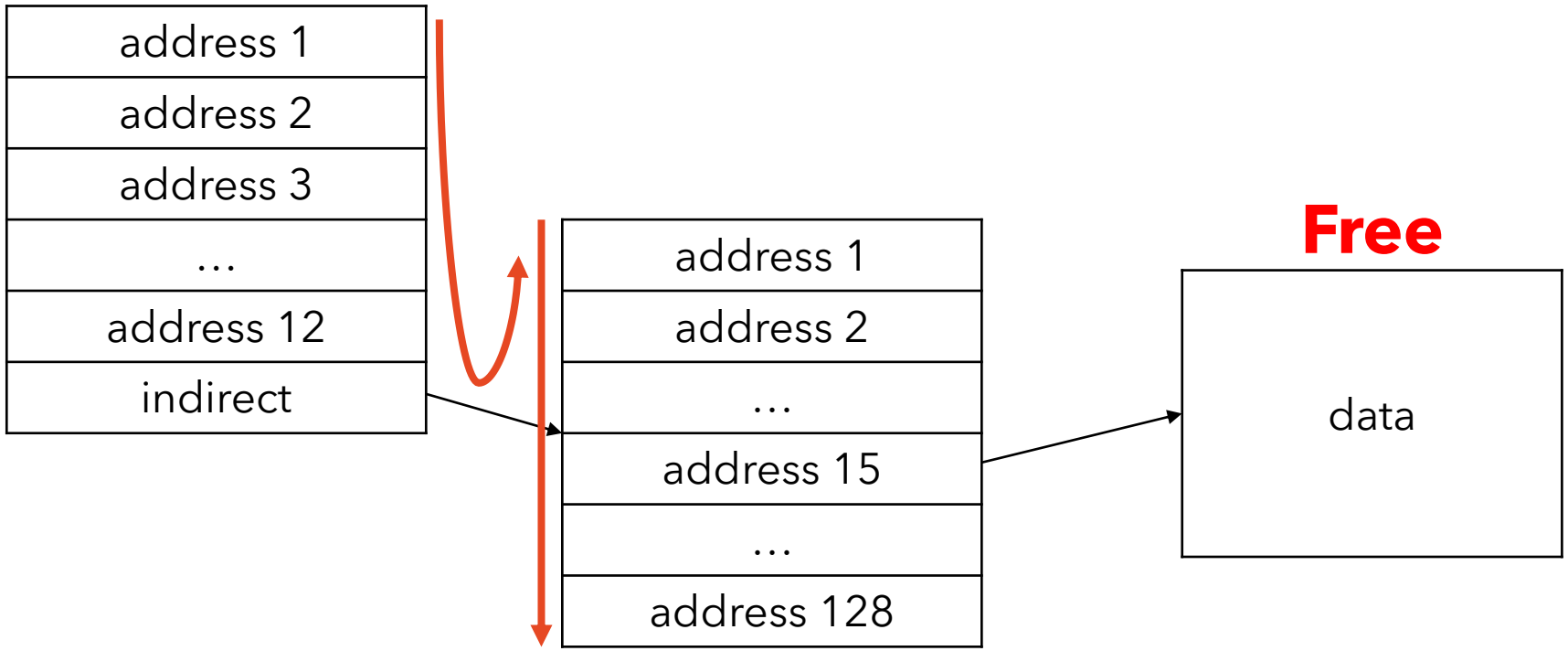
Xv6 – File system: Problem 2

- **fs.c**
  - bmap(struct inode *ip, uint n): Return the disk block address of the nth block in inode ip
    - e.g., if n = 27



Xv6 – File system: Problem 2

- **fs.c**
  - itrunc(struct inode *ip): Truncate inode. Only called when the inode has no links



Xv6 – File system: Problem 2

- Test your code with big / integrity / usertests programs
  - big: print the max number of blocks per file
    - If you implement triple indirect block, add limitation for test time

It doesn't have to be this number
At least 10MB is fine
Write the **reason for your value** in the report.

```
$ big
100 blocks
wrote 140 blocks, 71680 bytes files.
$
```

```
$ big
16500 blocks
wrote 16523 blocks, 8459776 bytes files.
$
```

  - integrity: check the file data's integrity (Read data should be identical to the data originally written)

```
Booting from Hard Disk..xv6...
cpu0: starting 0
sb: size 200000 nblocks 199893 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ integrity
integrity tests
create test done
100%
write test done
100%
read test done
before unlink
after unlink
integrity tests ok
--------------
integrity test 2
create test done
write test done
read test done
before unlink
after unlink
integrity test 2 ok
```

Xv6 – File system: Problem 2

- Test your code with big / integrity / usertests programs
  - If user code may be correct, “ALL TEST PASSED” will be printed

```
bigdir test
bigdir ok
uio test
pid 551 usertests: trap 13 err 0 on cpu 0 eip 0x3563 addr 0xcf9c--kill proc
uio test done
exec test
ALL TESTS PASSED
$
```

Project #3 – Advanced xv6 File system

- Deadline
  - ~ 2022.12.21 (Wed) 23:59
- Hand-in procedure
  - p3_201812345.patch (1/2)
  - Run the following command and upload p3_201812345.patch
    - git diff > p3_201812345.patch
  - Check the patch file with Notepad and confirm your modifications are in the patch file

Project #3 – Advanced xv6 File system

- Deadline
  - ~ 2022.12.21 (Wed) 23:59
- Hand-in procedure
  - Report (2/2)
    - Submit an 1~2 pages report
      - Free format (Korean/English)
      - **<Problem 1>** Calculate the (max number of files) from an existing xv6 file system
        - The process must be included
      - **<Problem 2>** Description of your implementation in a simple manner
        - (e.g., 12 direct blocks + 1 double indirect block)
        - Write the **number of blocks** and **file size** that your file system can theoretically make
        - The process must be included
      - Insert test code result image

Finally …

Do NOT hesitate to ask questions!

Mini Project #1, #2	Juhyung Park	*arter97@dgist.ac.kr*
Project #1	Minjae Kim	*jwya2149@dgist.ac.kr*
Project #2	Seonggyun Oh	*sungkyun123@dgist.ac.kr*
Project #3	**Soyoung Han**	***zsally@dgist.ac.kr***