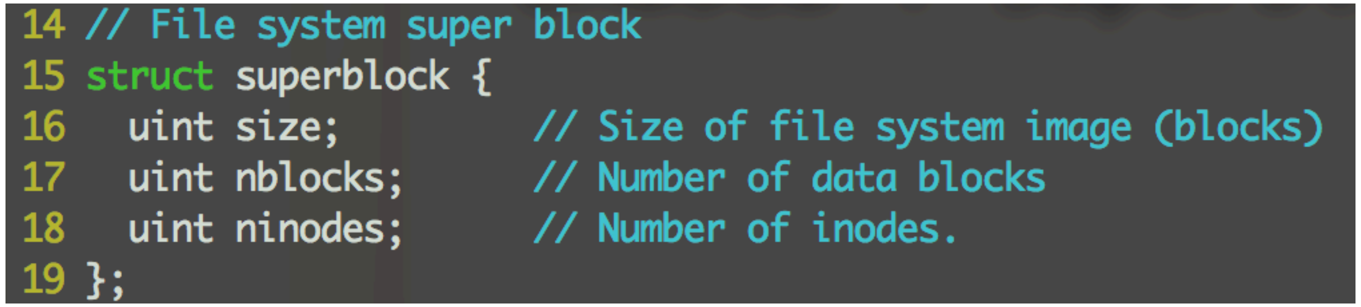
File System Checkers

Zhewen Song

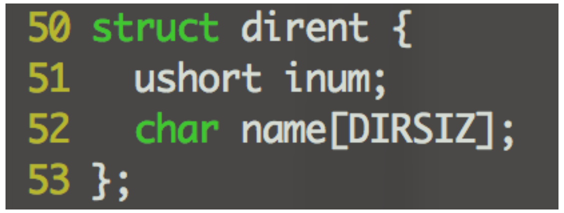
08/05/2017

File System Layout in xv6

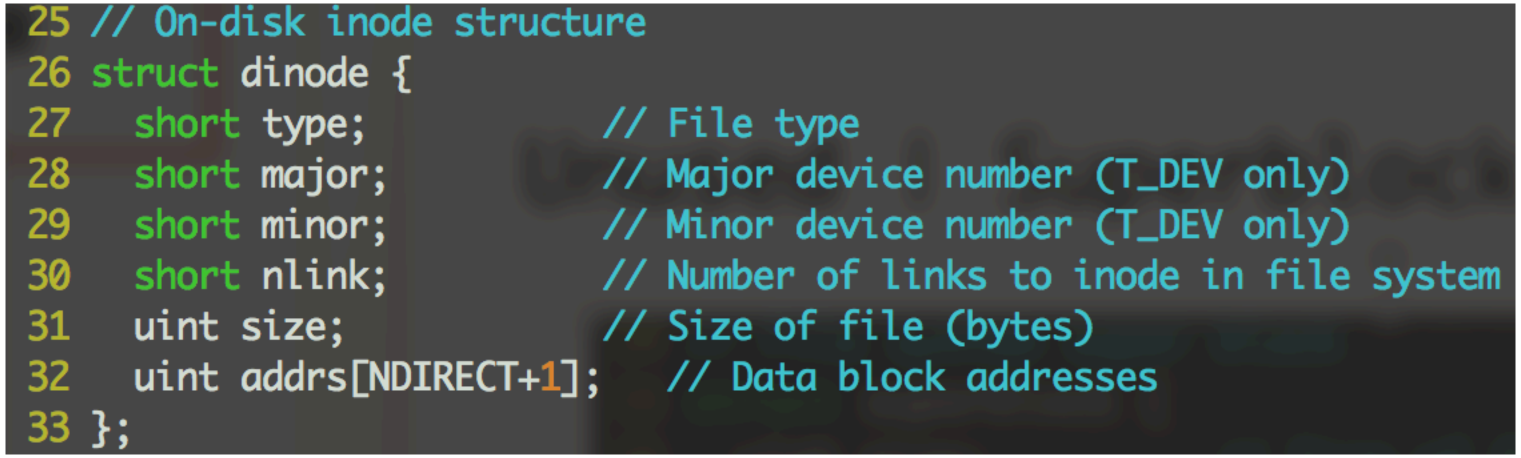
```
Unused | Superblock | Inodes ... | Bitmap | Data ...
```



```c
// File system super block
struct superblock {
  uint size;          // Size of file system image (blocks)
  uint nblocks;       // Number of data blocks
  uint ninodes;       // Number of inodes.
};
```

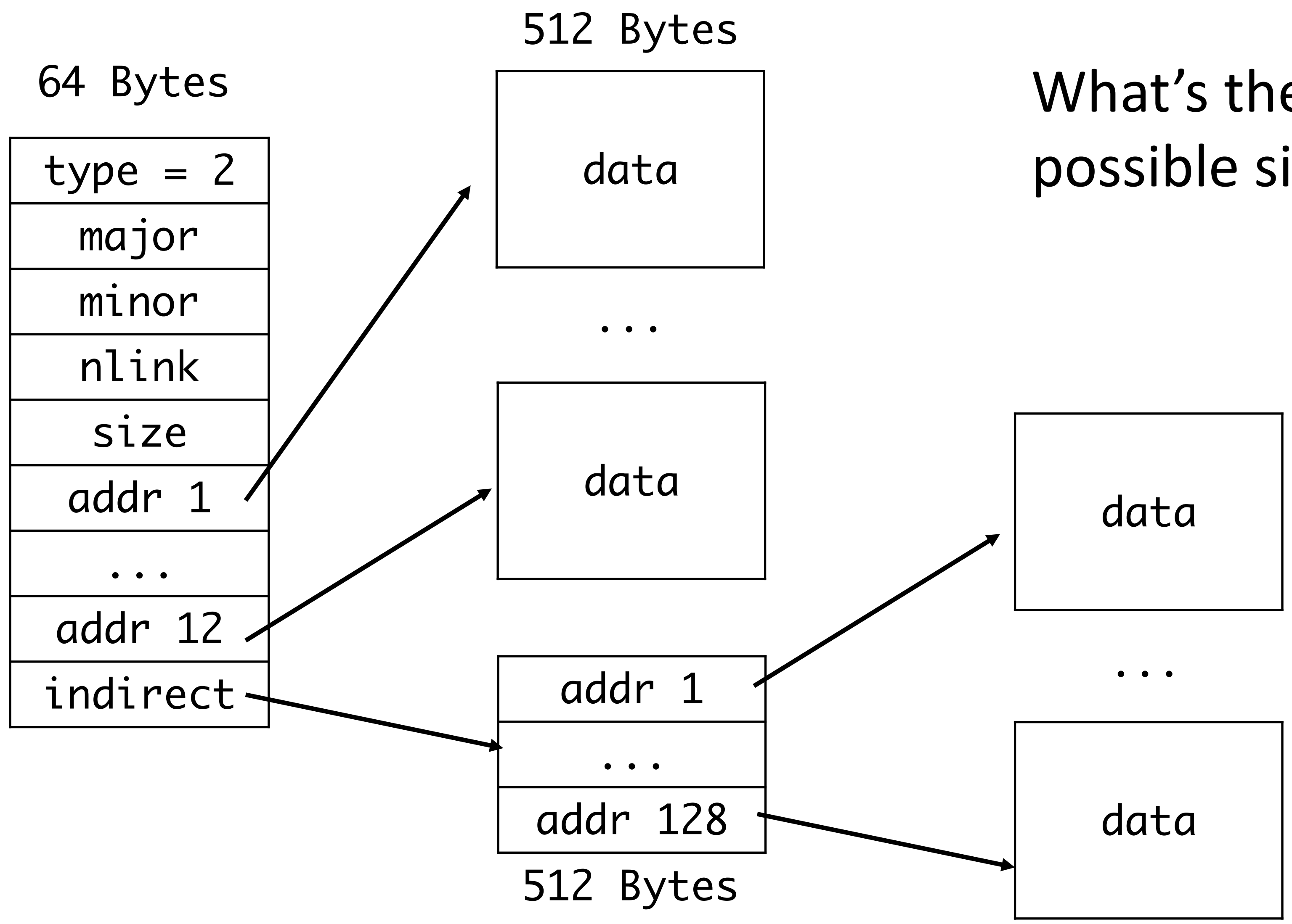


```c
struct dirent {
  ushort inum;
  char name[DIRSIZ];
};
```



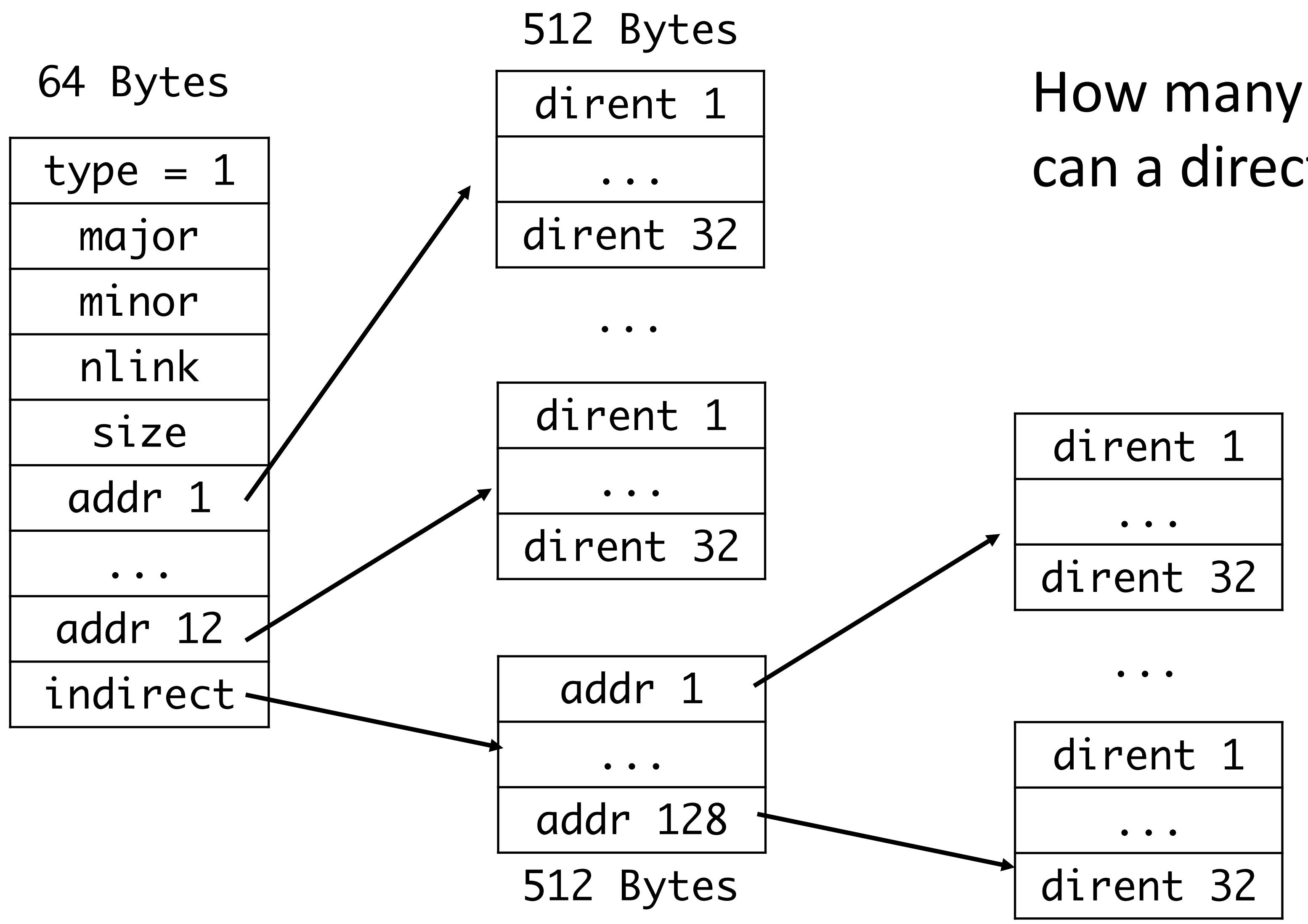
```c
// On-disk inode structure
struct dinode {
  short type;           // File type
  short major;          // Major device number (T_DEV only)
  short minor;          // Minor device number (T_DEV only)
  short nlink;          // Number of links to inode in file system
  uint size;            // Size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

Inode of a regular file in xv6



What's the maximum possible size of a regular file?

Inode of a directory in xv6



How many files/subdirectories can a directory have at most?

Bitmap in xv6

- Each bit in the bitmap is associated with a block, NOT an inode.
- Although the very first block is unused, it is always marked as 1 in the first bit of the bitmap, and so are all the blocks where the inodes and bitmap itself reside.
- Bitmap is grouped in byte.
- Intel x86 processors use little-endian.
- Example:
  - `ff c2 => 1111 1111 1100 0010`

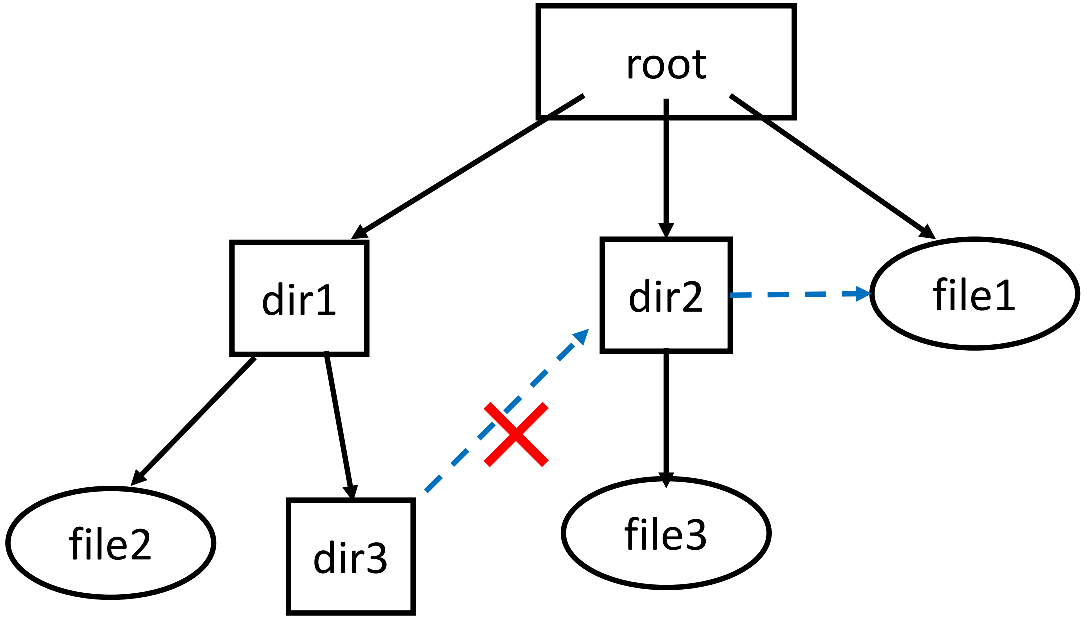
```
           7 6 5 4 3 2 1 0 | 15 14 13 12 11 10 9 8
```

Demos

- How to build your own file system image and reflect in xv6
  - Closer look at `mkfs.c`
- How xv6 files change update the image
  - Closer look at `fs.img` with `xxd`
- How to read image? – `mmap()!`

How `ln` works

- Reference counts (number of links) for regular files match the number of times file is referred to in directories (i.e., ***hard links*** work correctly).
- No extra links allowed for directories (each directory only appears in one other directory).


root
dir1
dir2
file1
file2
dir3
file3