CS 111 Project 1:

Reading Unix V6 File Systems

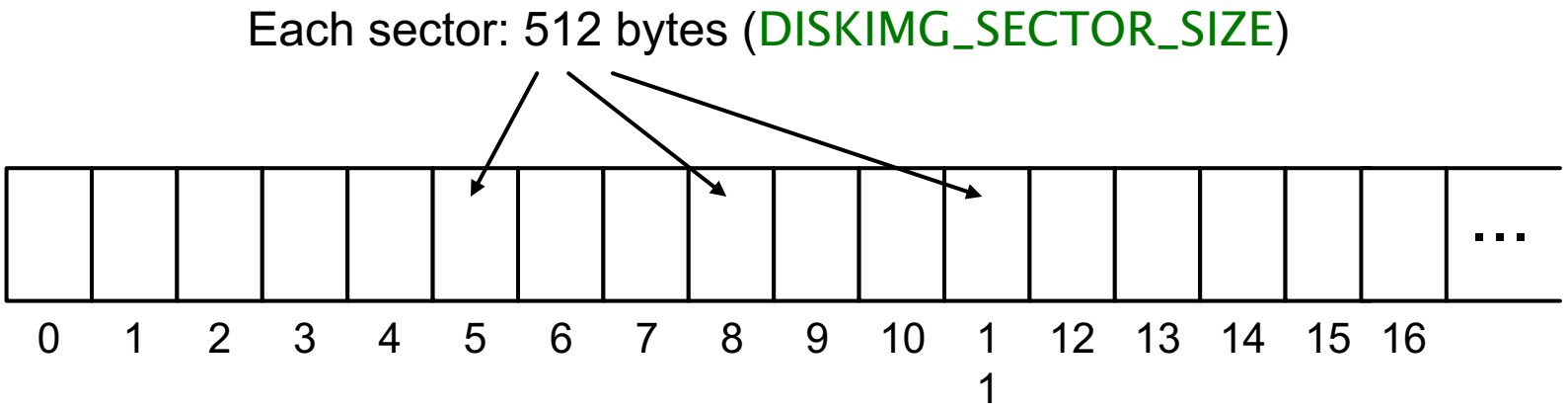
Layers

- **Block layer (filsys.h, unixfilesystem.h, diskimg.h, diskimg.c):**
  - Read physical blocks from the device
- **Inode layer (ino.h, inode.h, inode.c):**
  - Find inodes on the device, map block index within file to physical block number
- **File layer(file.h, file.c):**
  - Given <inumber, block index>, find and read physical block from device
- **Directory layer (direntv6.h, directory.h, directory.c):**
  - Read directory file to find a given name
- **Pathname layer (pathname.h, pathname.c):**
  - Parse a file path (“/a/b/c”), read many directories to find file’s inumber

Block Layer

int diskimg_readsector(int fd, int sectorNum, void *buf)

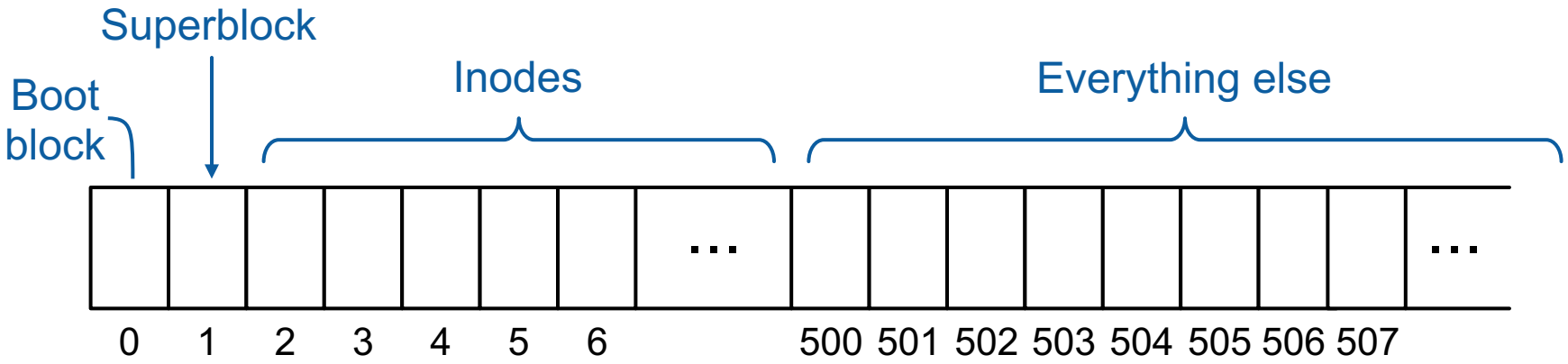
- Must always read full sectors



Disk: array of sectors (block == sector)

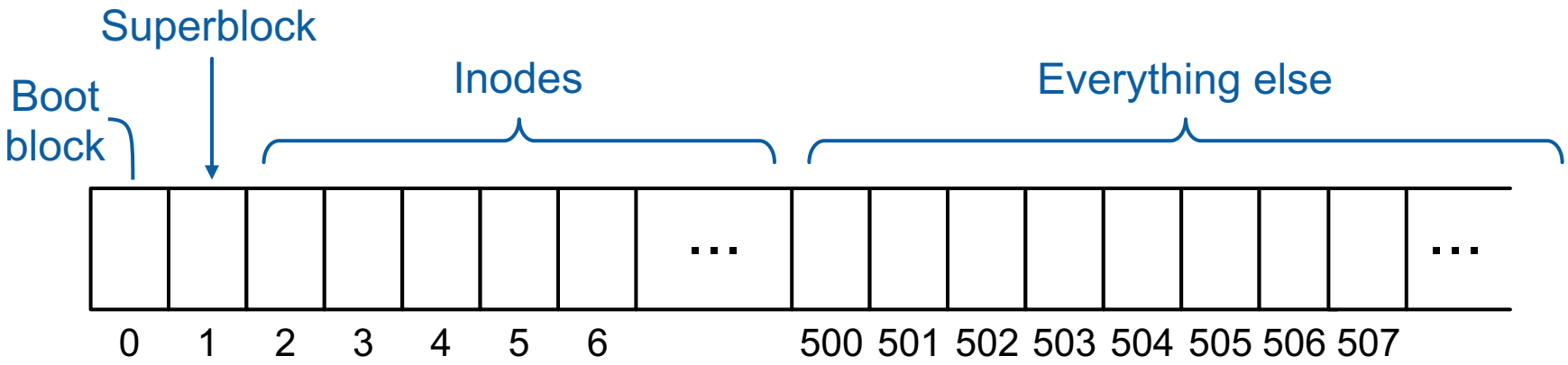
Unix V6 Disk Layout

- **Boot block: tiny program used to start up system**
- **Superblock: overall information about filesystem**
  - See filsys.h
  - Example: `s_isize` (# of blocks of inodes)
- **Everything else: file blocks, directory blocks, indirect blocks**



Metadata

- **Information stored on disk that allows us to identify what everything is**
- **Some metadata stored in fixed position on disk**
  - Superblock
  - Inodes
- **Some metadata is identified by other metadata**
  - Inodes tell us which blocks are indirect blocks, etc.



struct unixfilesystem

- **Overall information passed into the functions you'll write (declared in unixfilesystem.h)**

```
struct unixfilesystem {
    int dfd;
    struct filsys superblock;
};
```

"handle" for disk image;
pass to dskimage_readsector

Superblock read
from disk (see filsys.h)

Inode Layer

- **Read an inode into memory:**

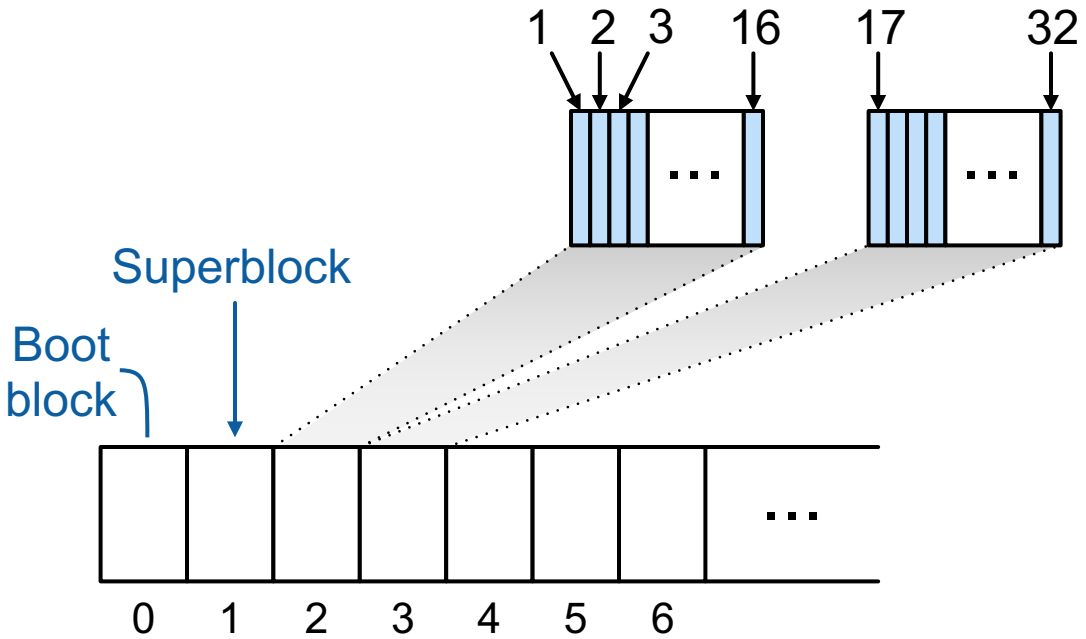
```
int inode_iget(struct unixfilesystem *fs, int inumber,
               struct inode *inp);
```

- **Given block index in file, find physical block number on disk:**

```
int inode_indexlookup(struct unixfilesystem *fs,
                      struct inode *inp, int fileBlockIndex);
```

inode_iget

```
int inode_iget(struct unixfilesystem *fs, int inumber,
               struct inode *inp);
```



- **inumbers start at 1, not 0.**
  - **You must zero-index it**
- **Inodes are 32 bytes (16 inodes per block)**
  - DISKIMG_SECTOR_SIZE = 512
  - DISKIMG_SECTOR_SIZE / 32 = 16

Writing Clean Code

- **Don't use numbers such as 1 and 2**
- **Instead, use symbols: ROOT_INUMBER (= 1) and INODE_START_SECTOR (= 2)**

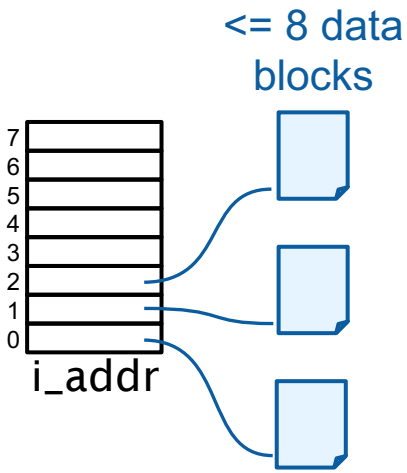
- **Worse: character buffers**

```
char buffer[512];
diskimg_readsector(fs->dfd, sector, buffer);
memcpy(inp, buffer+32*ix, 32);
```

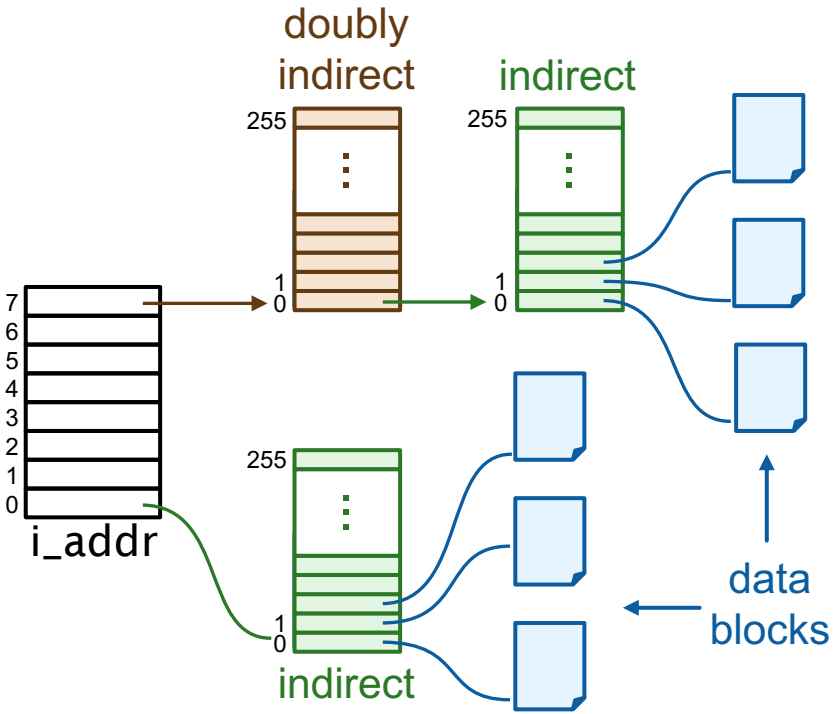
- **Better: typed buffers** (helpful for inode_iget !!!)

```
struct inode inodes[INODES_PER_BLOCK];
diskimg_readsector(fs->dfd, sector, inodes);
*inp = inodes[ix];
```

2 Inode Formats



Small

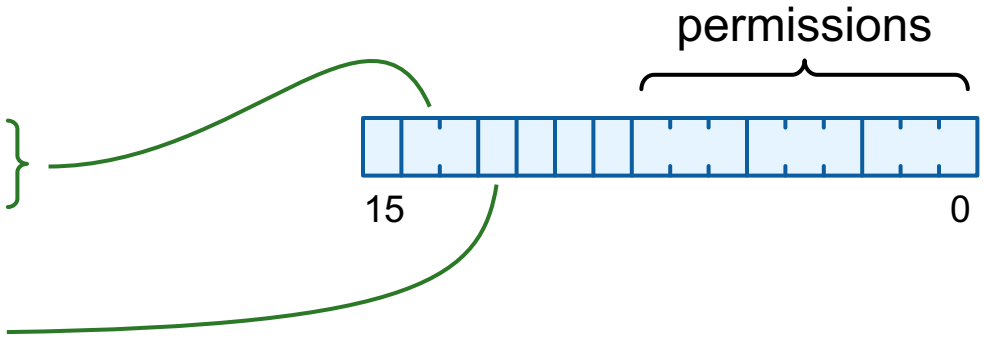


Large

i_mode

16-bit value with many packed fields (see ino.h):

```
#define IALLOC 0100000
#define IFMT    060000
#define IFDIR   040000
#define IFCHR   020000
#define IFBLK   060000
#define ILARG   010000
#define ISUID    04000
#define ISGID    02000
#define ISVTX    01000
#define IREAD     0400
#define IWRITE    0200
#define IEXEC     0100
```



Use binary ops to test fields:

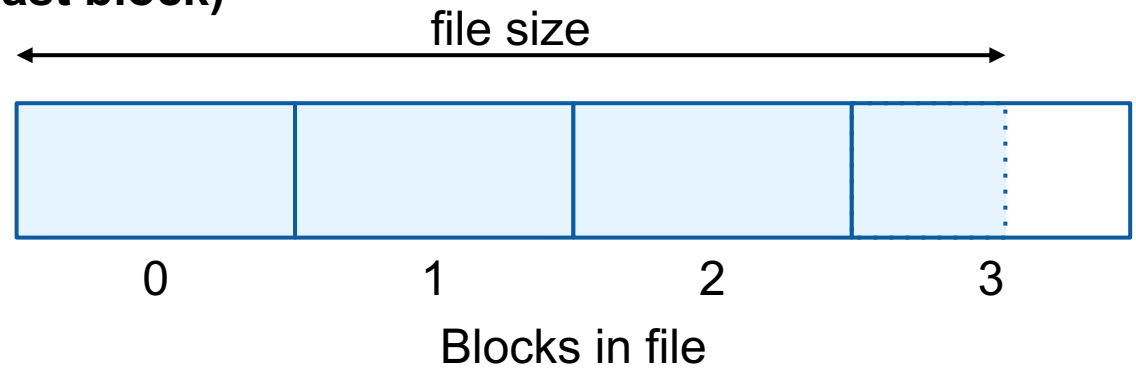
- **Is file large?** (Helpful for inode_indexlookup !!!)
  if (inode.i_mode & ILARG) ...

File Layer

- **Read a block of a file into memory:**

```
int file_getblock(struct unixfilesystem *fs, int inumber,
                  int fileBlockIndex, void *buf);
```

- **Steps: Get file's inode, find disk block number, read disk block**
- **Returns number of valid bytes in the buffer (< DISKIMG_SECTOR_SIZE for last block)**



Directory Layer

- **Find directory entry with given name:**

```
int directory_findname(
    struct unixfilesystem *fs,
    const char *name,
    int dirinumber,
    struct direntv6 *dirEnt);
```

- **Read directory file one block at a time: use file_getblock()**
  - Each block contains NUM_DIRENTS_PER_BLOCK entries

- **Struct direntv6 (from direntv6.h):**

```
struct direntv6 {
    uint16_t d_inumber;
    char d_name[MAX_COMPONENT_LENGTH];
};
```

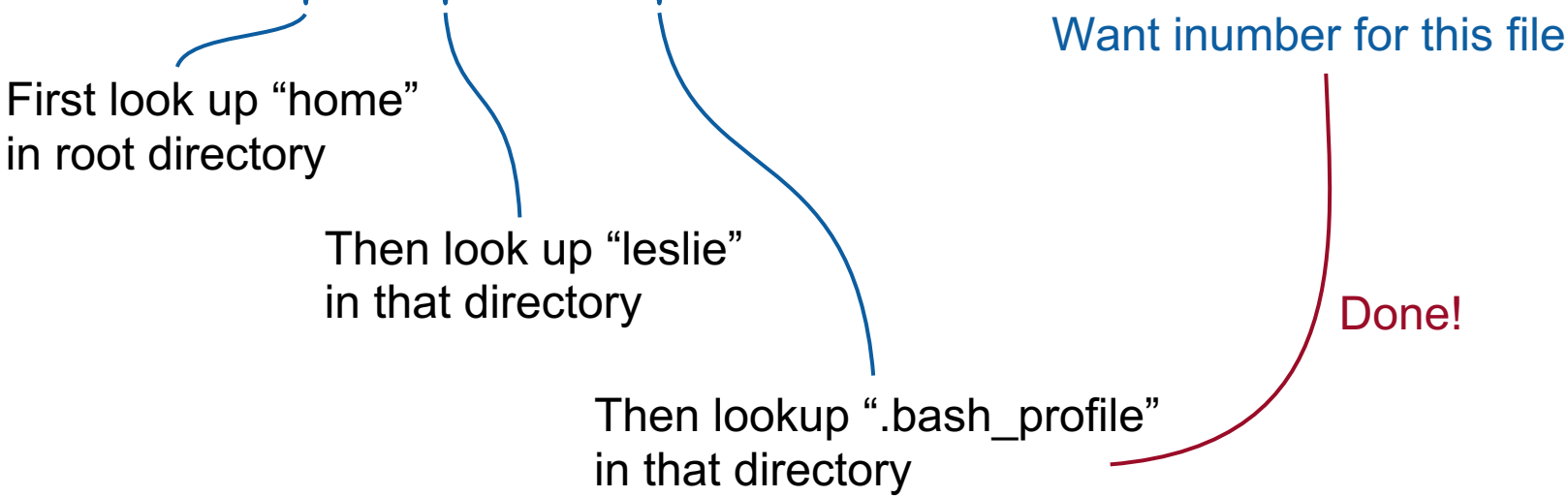
- **Note: d_name is only null-terminated if it has fewer than MAX_COMPONENT_LENGTH chars!**
  - Use strncmp() for to compare with name, not strcmp()

Pathname Layer

- **Scan a multi-level file path, find inumber for file:**

```
int pathname_lookup(struct unixfilesystem *fs,
                    const char *pathname);
```

- **Example: /home/leslie/.bash_profile**



Pathname Layer, cont'd

- Use `strsep()` to parse the path to get each string before "/"
- But, `strsep()` modifies its argument; can't modify `pathname` argument to `pathname_lookup()`
- Must copy `pathname`
- Use recursion or iteration?
  - Iteration is likely to be simpler

Odds and Ends

- **Functions that may be useful in this project:**
  - strsep
  - strlen
  - strcpy
  - strncmp
- **No need for heap allocation of memory**
- **Most functions can return errors**
  - Must check for errors:
    - calling diskimg_readsector
    - Functions you wrote and later call may return errors
  - Print human-readable messages after errors (and propagate errors upwards)
- **Don’t duplicate code from lower layers: use the functions**
- **Valgrind will work for this project**