Agenda

Programming Review: Binary files, Structuring memory

Directories

Handling filenames

Handling symbolic links

Managing free disk space

Programming tips

Examples

Review: Working with binary files

```
struct point {
  int x;
  int y;
};

int main() {
  struct point p[3];
  for (int i = 0; i < 3; i++) {
    p[i].x = i;
    p[i].y = 2*i;
  }

  FILE* fp = fopen("points.bin", "wb");
  fwrite(p, sizeof(struct point), 3, fp);
  fclose(fp);
}
```

Write File

```
struct point {
  int x;
  int y;
};

int main() {
  struct point p[3];

  FILE* fp = fopen("points.bin", "rb");
  fread(p, sizeof(struct point), 3, fp);
  fclose(fp);

  for (int i = 0; i < 3; i++) {
    printf("p[%d] = (%d,%d)\n", i, p[i].x, p[i].y);
  }
}
```

Read File

What are the contents of points.bin?

alinen@sutekh:~/cs355/os-devel/lectures/files$ **xxd points.bin**
00000000: 0000 0000 0000 0000 0100 0000 0200 0000 ....
00000010: 0200 0000 0400 0000   ....

alinen@sutekh:~/cs355/os-devel/lectures/files$ **xxd -b points.bin**
00000000: 00000000 00000000 00000000 00000000 00000000 00000000 ......
00000006: 00000000 00000000 00000001 00000000 00000000 00000000 ......
0000000c: 00000010 00000000 00000000 00000000 00000010 00000000 ......
00000012: 00000000 00000000 00000100 00000000 00000000 00000000 ......

alinen@sutekh:~/cs355/os-devel/lectures/files$ **hexedit points.bin**

A simple filesystem

```
#define BLOCKSIZE 1024
#define MAX_FILENAME 20
#define INVALID_ID 255

struct superblock_t {
  long num_files;
};
struct direntry_t {
  char name[MAX_FILENAME];
  int first_block_id;
};
struct fileblock_t {
  char block[BLOCKSIZE];
};
struct fatentry_t {
  int nextid;
};
```

```
struct fs_t {
  superblock_t sb;
  direntry_t directory[8]; // up to 8 files
  unsigned char free_bitmap; // up to 8 blocks
  fatentry_t FAT[8];
  fileblock_t blocks[8];
};
```

A simple filesystem

```
fs_t filesystem;
// Create a file that uses 1 block
filesystem.sb.num_files = 1;

// Step 1: Create a directory entry
strncpy(filesystem.directory[0].name, "foo.txt", MAX_FILENAME);
filesystem.directory[0].first_block_id = 0;

// Step 2: Mark the block as used
filesystem.free_bitmap = 0x80;

// Step 3a: Initialize the FAT table (all entries invalid/not set)
memset(filesystem.FAT, INVALID_ID, sizeof(fatentry_t) * 8);

// Step 3b: Indicate the blocks for our file foo.txt
filesystem.FAT[0].nextid = INVALID_ID;

// Step 4: Store the file's data in blocks[0]
strncpy(filesystem.blocks[0].block, "The file contents!", BLOCKSIZE);

FILE* fp = fopen("fs.bin", "wb");
fwrite(&filesystem, sizeof(struct fs_t), 1, fp);
fclose(fp);
```

Write File

```
fs_t filesystem;

FILE* fp = fopen("fs.bin", "rb");
fread(&filesystem, sizeof(struct fs_t), 1, fp);
fclose(fp);

int num_files = filesystem.sb.num_files;

// Print all files
for (int i = 0; i < num_files; i++) {
 direntry_t dirent = filesystem.directory[i];
 printf("File Name: %s\n", dirent.name);
 printf("File Block: %d\n", dirent.first_block_id);

 printf("File Contents: %s\n",
     filesystem.blocks[dirent.first_block_id].block);

 int blockid = filesystem.FAT[dirent.first_block_id].nextid;
 while (blockid != INVALID_ID) {
  printf("File Contents: %s\n", filesystem.blocks[blockid].block);
  blockid = filesystem.FAT[dirent.first_block_id].nextid;
 }
}
```

Read File

Directories

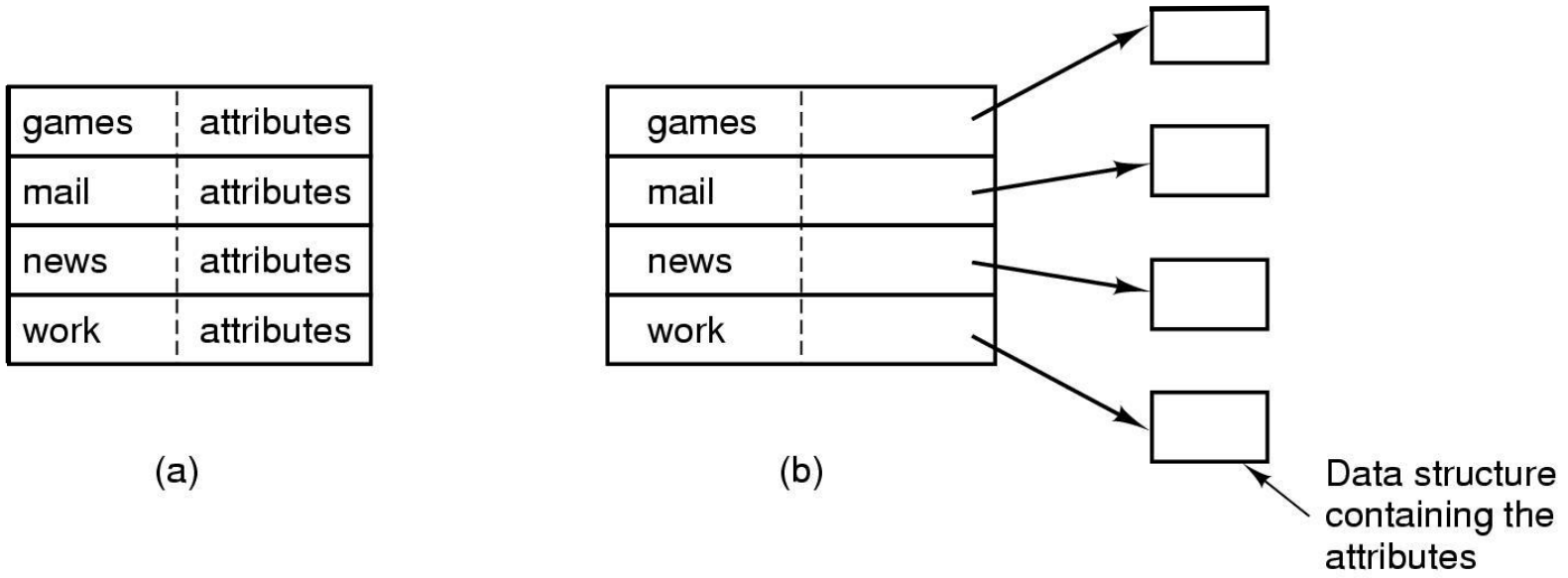
A **directory** is a special type of file

Main purpose: Map names to files

Some systems also store file attributes in directories (e.g. MS DOS)

Question: What are some examples of file attributes?

Example: Directories



(a) A simple MS-DOS directory

fixed size entries

disk addresses and attributes in directory entry

(b) Unix directory in which each entry just refers to an i-node

UNIX: Directory path names

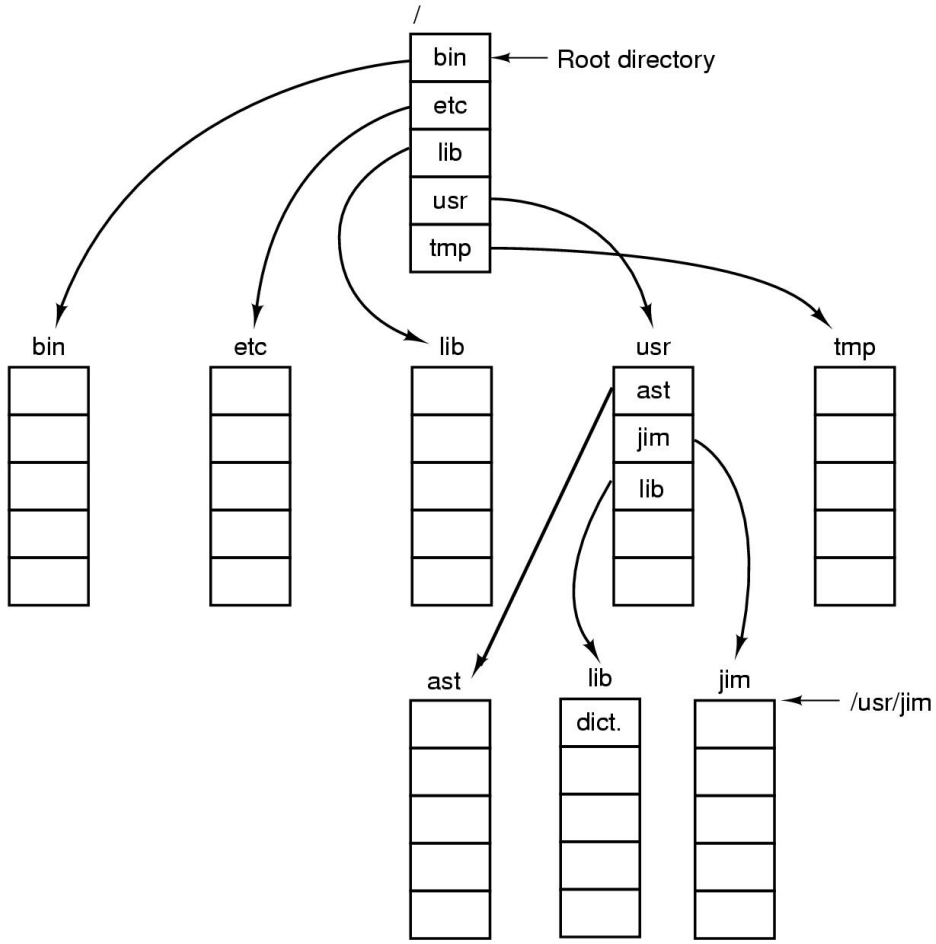
Idea: Resolving paths involves looking at the tables within each directory file

Absolute Paths:

- The root directory, e.g. /, typically stored at a known location in the inode list
- Resolving the name requires looking up the inode for / and then following the inodes in each directory table

Relative Paths:

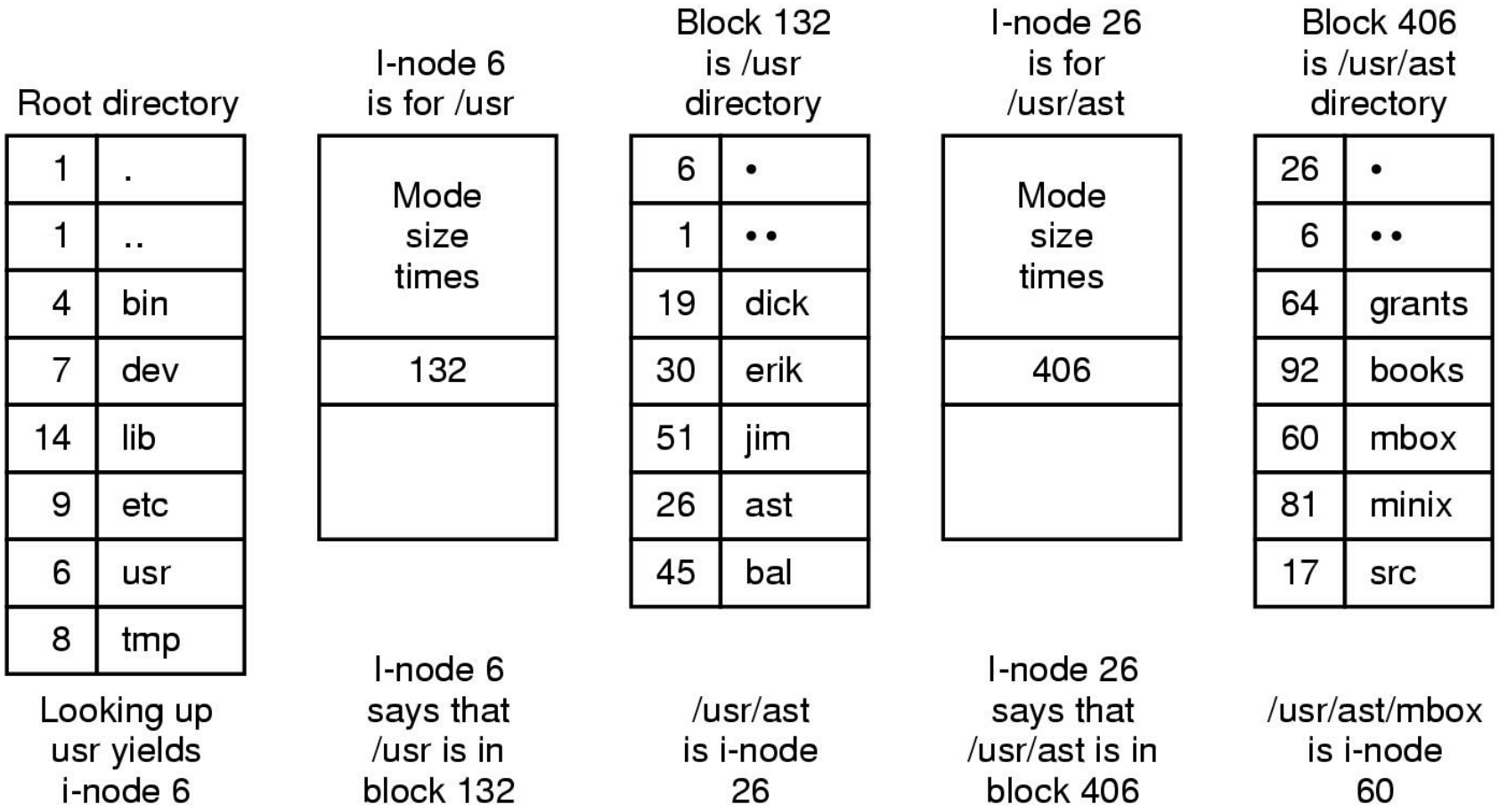
- We store the inodes corresponding to **.** (current directory) and **..** (parent directory)
- Resolving the name requires looking up the inodes for . or .. and then following the inodes in each directory table



A UNIX directory tree

NOTE: We can cache filenames to make lookups faster

The UNIX V7 Directory Lookup

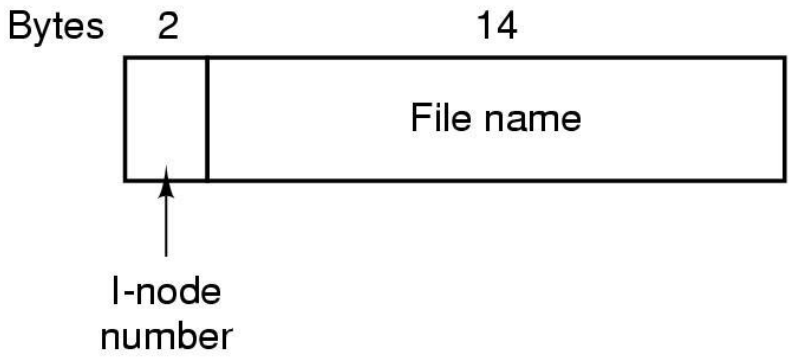


The steps in looking up */usr/ast/mbox*

Implementing filenames

Directories need to map filenames to physical data blocks

Example: UNIX V7 directories consisted of a list of 16 byte entries



Two Approaches

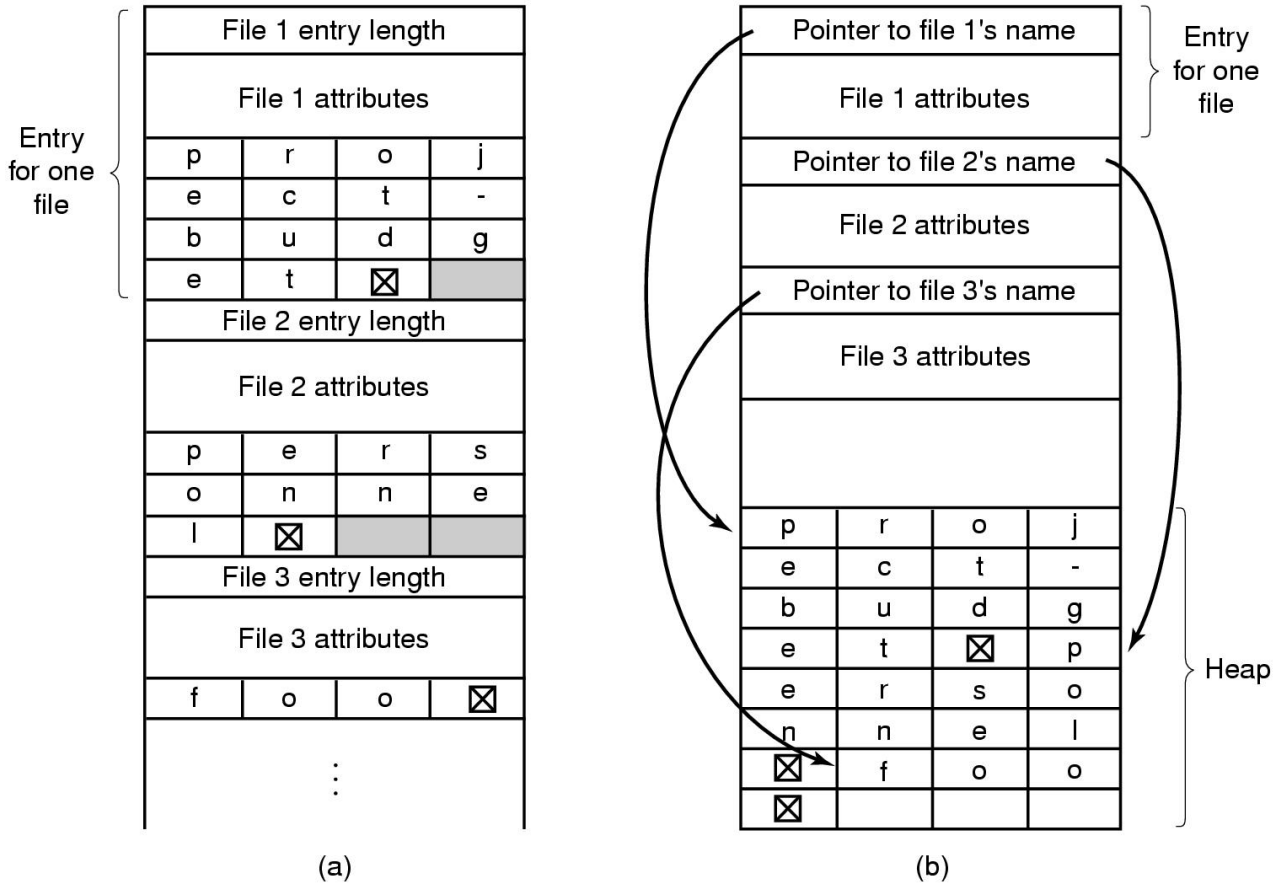
- Specify a max filename
  - Examples: MS-DOS Max was 8 chars + 3 chars for the extension; ext2 had 255 chars
- Support variable filenames

Variable length filenames

Two ways of handling long file names in directory

- (a) In-line
  - Store length of entry
  - Problem: Awkward to update when names are removed
- (b) In a heap
  - Store variable data at the end

Example shows storing the directory entries: project-budget, personnel, foo



Handling symbolic links

- Convenient solution to file sharing
- Hard link
  - pointer to i-node added to the directory entries
  - link counter in the i-node linked to incremented
  - i-node can only be deleted (data addresses cleared) if counter goes down to 0, why?
  - original owner can not free disk quota unless all hard links are deleted

- Soft link (symbolic)
  - new special file created of type LINK, which contains just the path name
  - new file also has its own i-node created
  - References to deleted files display as a broken link

Exercise: Shared Files – Link

Draw the directory structure corresponding to the following commands

```
$ pwd
/home/xenu

$ mkdir A
$ mkdir B
$ touch A/A.txt
$ cd B
$ ln –s ../A/A.txt B.txt
```

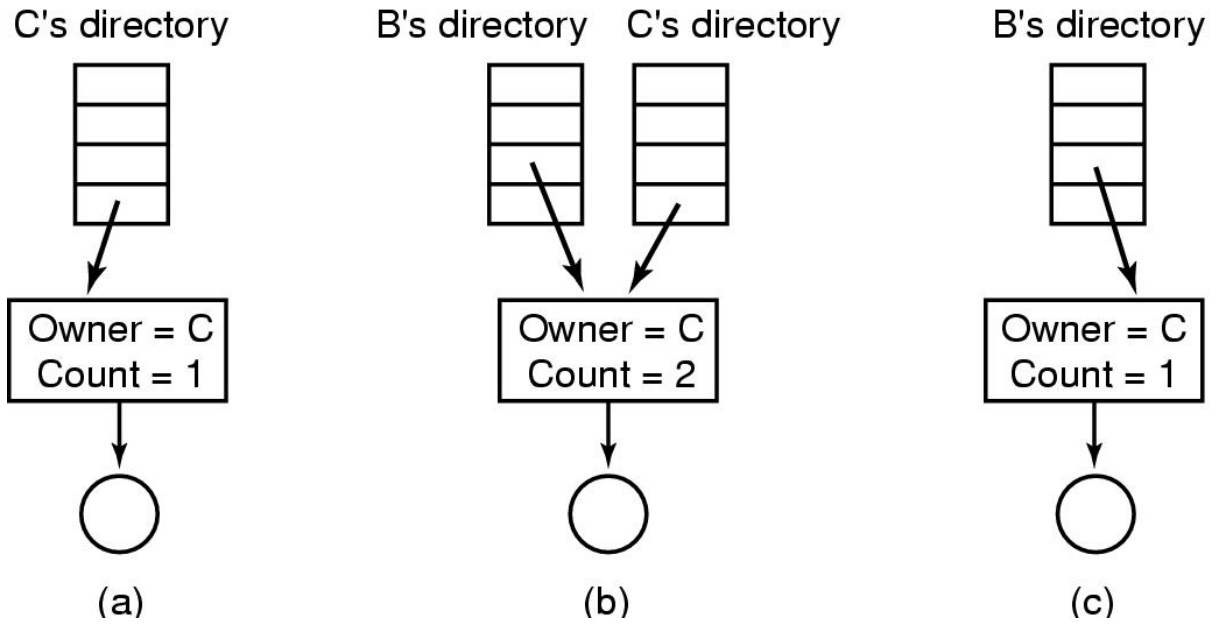
File system containing a shared file

Shared Files – i-node

(a) Situation prior to linking

(b) After the link is created

(c)After the original owner removes the file



Managing Free Disk Space

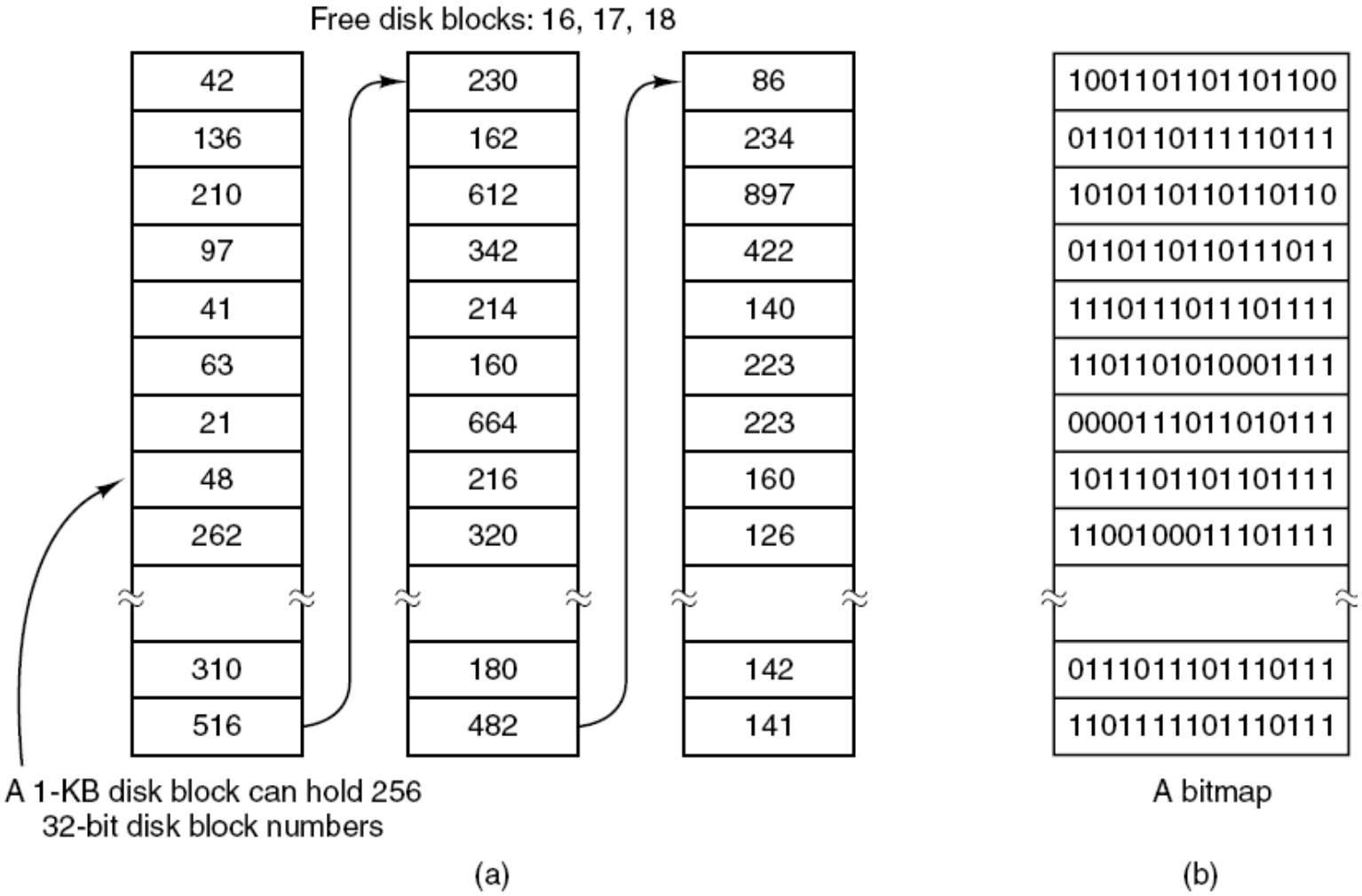
How should we keep track of free blocks?

Linked List Method

- Maintained as a list of blocks containing addresses of free blocks
- Each block address is 32 bits or 4 Bytes
- A 1KB block used can hold addresses of 255 free blocks and an address of the next block in the list for free space management
- Note: This list is stored in free blocks themselves

- Bitmap method:
  - Keep an array of bits, one per block indicating whether that block is free or not
  - A 16-GB disk has 16 million 1 KB blocks
  - Storing the bitmap requires 16 million bits, or 2048 blocks

Managing Free Blocks



(a) Storing the free list on a linked list

(b) A bit map

Example: Putting it all together (inodes)

Assume BLOCKSIZE 1024

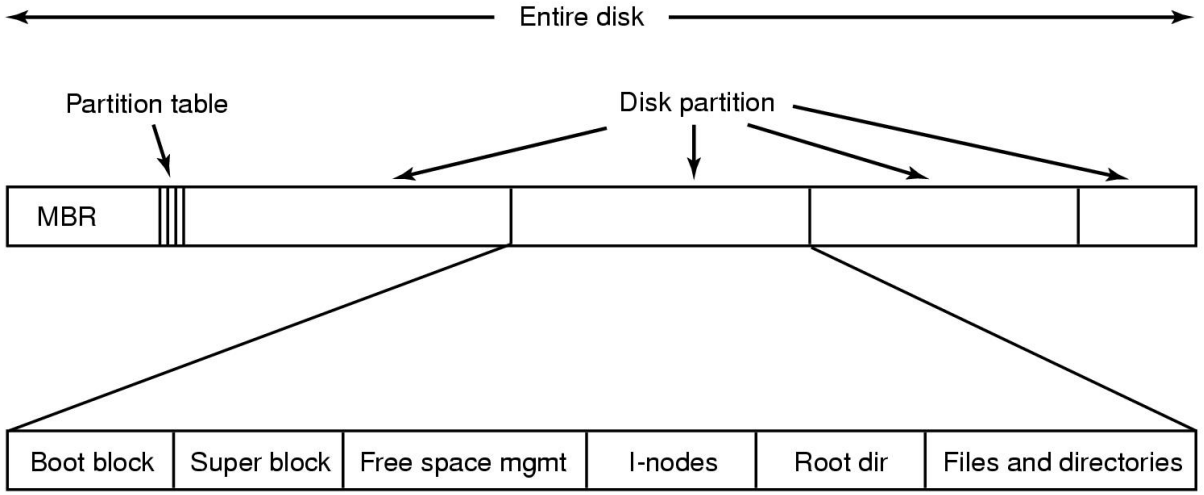
Assume Boot block, superblock, free space management are all 1 block

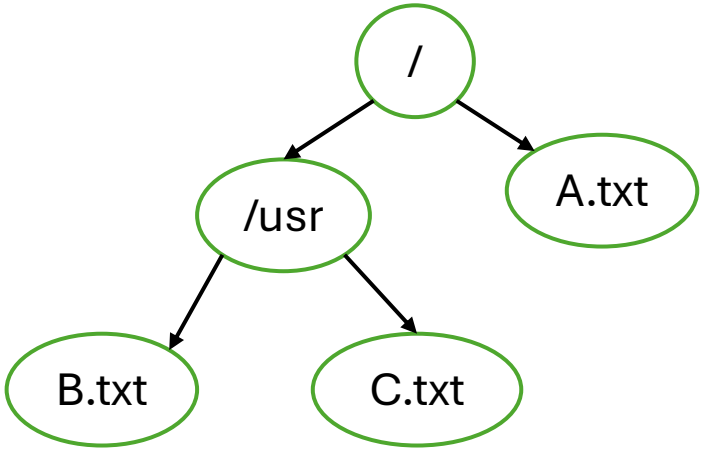
Superblock contains:

- blocksize
- inode offset (in blocks w.r.t superblock)
- data offset (in blocks w.r.t superblock)
- swap space offset

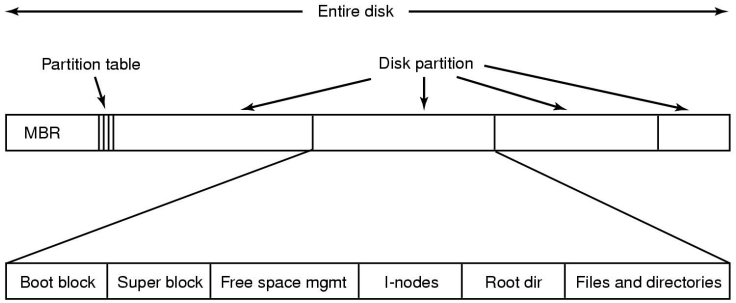
Assume inodes are 128 bytes each

Filenames are 14 characters

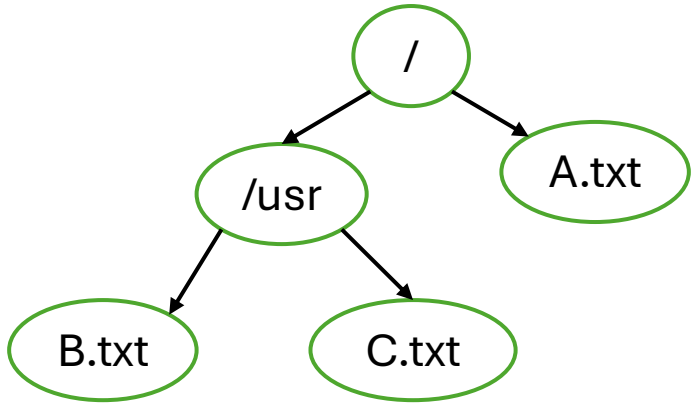




Example: Putting it all together



Draw inodes and blocks that could store the file system on the previous page. Assume all directories/files fit inside a 1 KB block.



How many inodes are needed?

How many blocks are needed?

Suppose a block has index 3, what is its memory address?

Example: Putting it together

Draw the inodes needed for this file system. Draw the data blocks.

Example: FAT

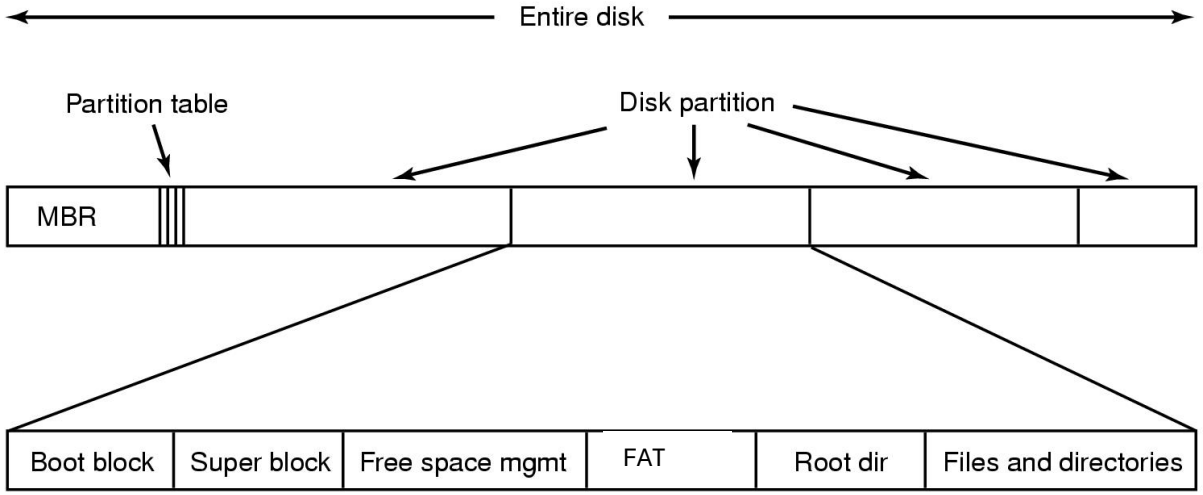
Assume BLOCKSIZE 1024 bytes (1 KB)

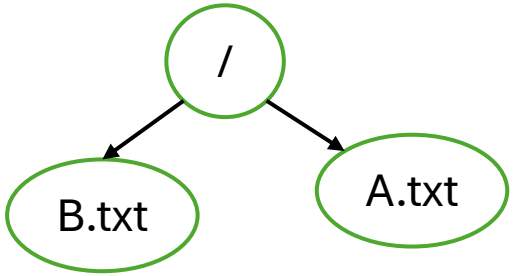
Assume Boot block, superblock, free space management and FAT are all 1 block

Superblock contains:

- blocksize
- data offset
- data offset (in blocks w.r.t superblock)
- swap space offset

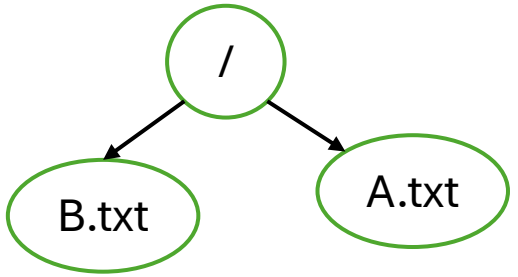
Free space is managed with a bitmap
Filenames are 8 characters



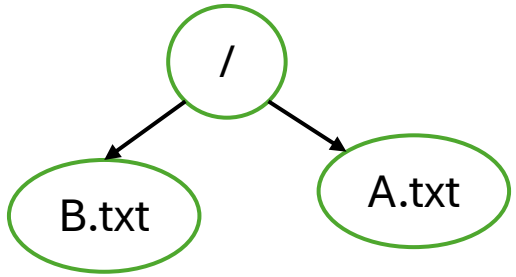


Example: FAT

All directories fit inside a 1 KB block.
Suppose the root directory is at block 0
Suppose A.txt uses blocks 2, 6, 7
Suppose B.txt uses blocks 3, 4, 8
Draw the contents of the first 10 FAT entries



Example: FAT



All directories fit inside a 1 KB block.

Suppose the root directory is at block 0

Suppose A.txt uses blocks 2, 6, 7

Suppose B.txt uses blocks 3, 4, 8

Draw the contents of the first 10 dir/file blocks

Assume directories contain names and FAT table ids

C programming: inode file system (Part 1)

```
char boot[512];
char super[512];
char swap[512];
inode inodes[4];
char blocks[BLOCKSIZE*NUM_BLOCKS];

 memset(boot, 0, 512);
 memset(super, 0, 512);
 memset(swap, 0, 512);
 memset(inodes, -1, sizeof(inode) * 4);

 superblock* sb = (superblock*) super;
 sb->size = BLOCKSIZE;
 sb->inode_offset = 0;
 sb->data_offset = (sizeof(inode) * 4)/BLOCKSIZE;
 sb->swap_offset = sb->data_offset + NUM_BLOCKS;
 sb->free_inode = 1;
 sb->free_block = 2;
```

C programming: inode file system (Part 2)

```
// setup root directory inode
 inodes[0].nlink = 1;
 inodes[0].size = sizeof(dentry) * 2;
 inodes[0].dblocks[0] = 0;
 inodes[0].dblocks[1] = 1;

 // setup root directory data
 dentry_t* directory = (dentry*) blocks;

 directory[0].inode = 0; // block 0
 strncpy(directory[0].name, ".", 8);
 directory[0].type = D;

 directory[1].inode = 0; // block 1
 strncpy(directory[1].name, "..", 8);
 directory[1].type = D;
```