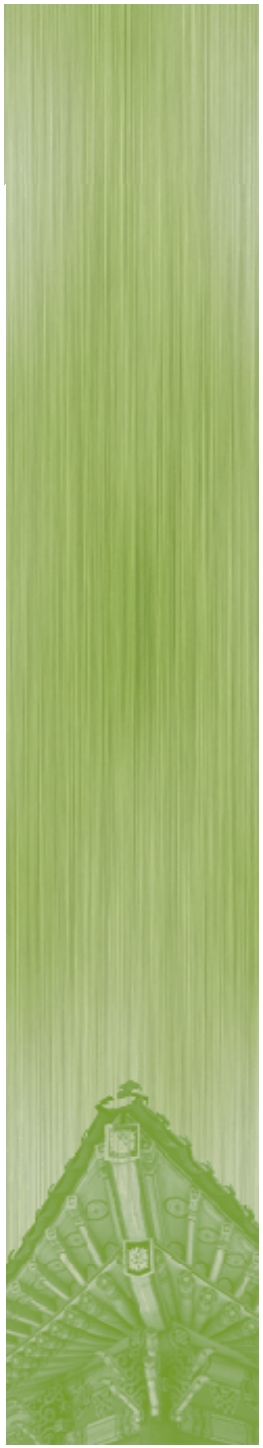
Final Exam

- **13:30 – 14:50, December 14 (Monday), 2009**
- **#330110**
- **Scope:**
  - Everything (including Pintos)
- **Closed-book exam**

- **Final exam scores will be posted in the lecture homepage**



File System Case Studies

Jin-Soo Kim (jinsookim@skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu

Today's Topics

- FFS
- Ext2
- FAT

FFS (1)

- **Fast file system (FFS)**
  - The original Unix file system (70's) was very simple and straightforwardly implemented:
    - Easy to implement and understand.
    - But very poor utilization of disk bandwidth (lots of seeking).
  - BSD Unix folks redesigned file system called FFS.
    - McKusick, Joy, Fabry, and Leffler (mid 80's)
    - Now it is the file system from which all other UNIX file systems have been compared.
  - The basic idea is aware of disk structure.
    - Place related things on nearby cylinders to reduce seeks.
    - Improved disk utilization, decreased response time.

FFS (2)

- **Data and i-node placement**
  - Original Unix FS had two major problems:

(1) Data blocks are allocated randomly in aging file systems.

  - Blocks for the same file allocated sequentially when FS is new.
  - As FS "ages" and fills, need to allocate blocks freed up when other files are deleted.
  - Problem: Deleted files essentially randomly placed.
  - So, blocks for new files become scattered across the disk.

(2) i-nodes are allocated far from blocks.

  - All i-nodes at the beginning of disk, far from data.
  - Traversing file name paths, manipulating files and directories require going back and forth from i-nodes to data blocks.

  - Both of these problems generate many long seeks!

FFS (3)

- **Cylinder groups**
  - BSD FFS addressed these problems using the notion of a cylinder group.
  - Disk partitioned into groups of cylinders.
  - Data blocks from a file all placed in the same cylinder group.
  - Files in same directory placed in the same cylinder group.
  - i-nodes for files allocated in the same cylinder group as file's data blocks.

Ext2 FS (1)

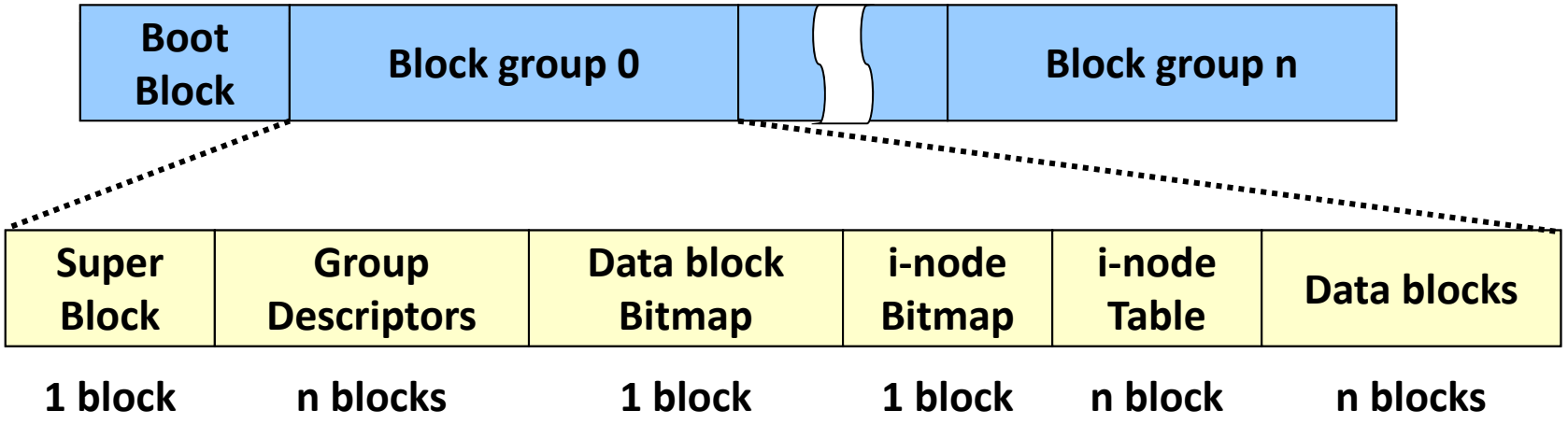
- **History**
  - Evolved from Minix filesystem.
    - Block addresses are stored in 16bit integers – maximal file system size is restricted to 64MB.
    - Directories contain fixed-size entries and the maximal file name was 14 characters.
  - Virtual File System (VFS) is added.
  - Extended Filesystem (Ext FS), 1992.
    - Added to Linux 0.96c
    - Maximum file system size was 2GB, and the maximal file name size was 255 characters.
  - Second Extended Filesystem (Ext2 FS), 1994.
  - Evolved to Ext3 File system (with journaling)

Ext2 FS (2)

- **Ext2 features**
  - Configurable block sizes (from 1KB to 4KB)
    - depending on the expected average file size.
  - Configurable number of i-nodes
    - depending on the expected number of files
  - Partitions disk blocks into groups.
    - lower average disk seek time
  - Preallocates disk data blocks to regular files.
    - reduces file fragmentation
  - Fast symbolic links
    - If the pathname of the symbolic link has 60 bytes or less, it is stored in the i-node.
  - Automatic consistency check at boot time.

Ext2 FS (3)

- **Disk layout**
  - Boot block
    - reserved for the partition boot sector
  - Block group
    - Similar to the cylinder group in FFS.
    - All the block groups have the same size and are stored sequentially.



Ext2 FS (4)

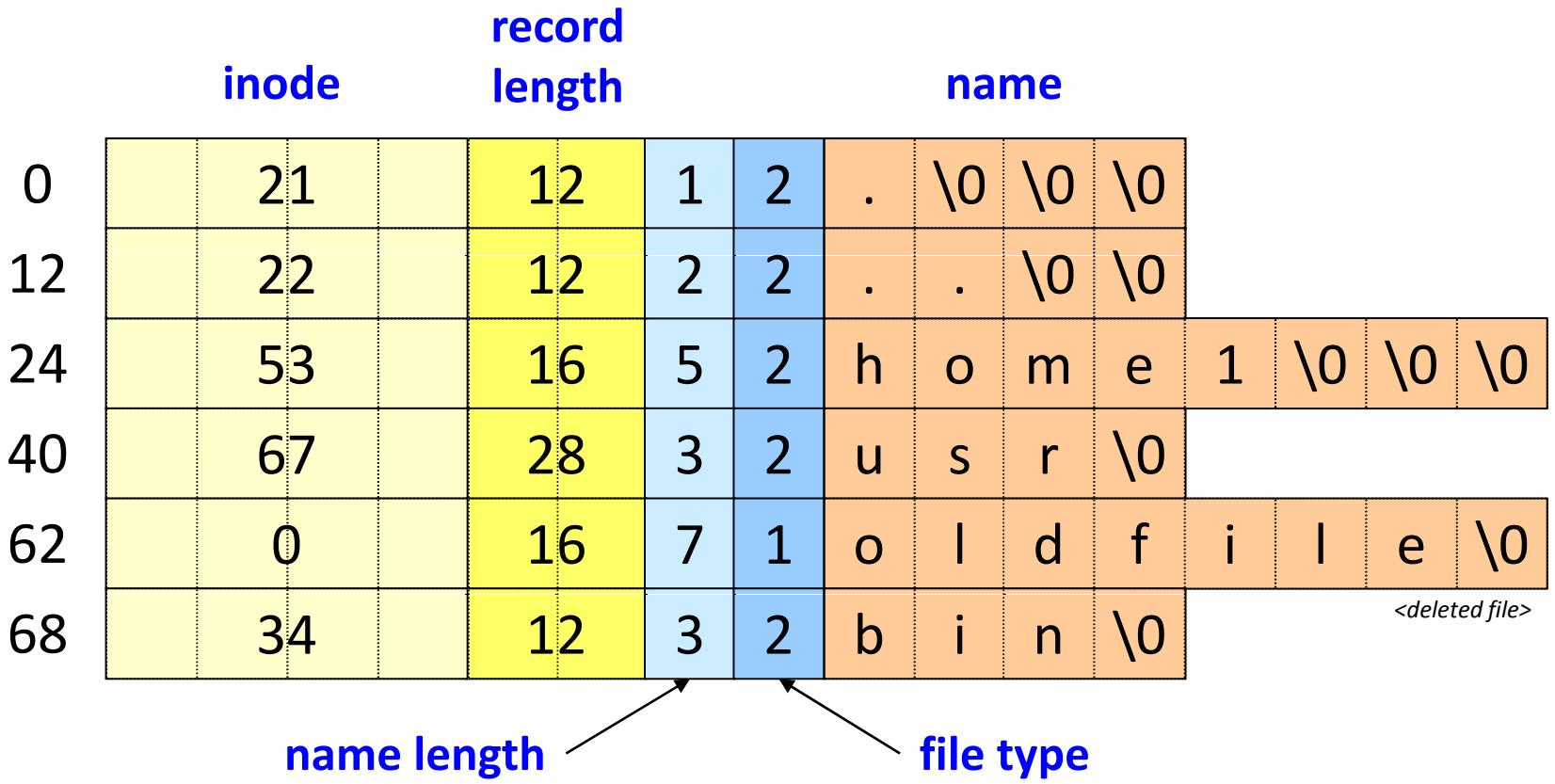
- **Block group**
  - Superblock: stores file system metadata
    - Total number of i-nodes,
    - File system size in blocks
    - Free blocks / i-nodes counter
    - Number of blocks / i-nodes per group
    - Block size, etc.
  - Group descriptor
    - Number of free blocks / i-nodes / directories in the group
    - Block number of block / i-node bitmap, etc.
  - Both the superblock and the group descriptors are duplicated in each block group.
    - Only those in block group 0 are used by the kernel.
    - fsck copies them into all other block groups.
    - When data corruption occurs, fsck uses old copies to bring the file system back to a consistent state.

Ext2 FS (5)

- **Block group size**
  - The block bitmap must be stored in a single block.
    - In each block group, there can be at most 8xb blocks, where b is the block size in bytes.
  - The smaller the block size, the larger the number of block groups.
  - Example: 8GB Ext2 partition with 4KB block size
    - Each 4KB block bitmap describes 32K data blocks
      = 32K * 4KB = 128MB
    - At most 64 block groups are needed.

Ext2 FS (6)

- Directory structure



FAT FS (1)

- **FAT filesystem**
  - Used in MS-DOS based OSes
    - MS-DOS, Windows 3.1, 95, 98, …
  - Originally developed as a simple file system suitable for floppy disk drives less than 500KB in size.
  - FAT stands for File Allocation Table.
    - Each FAT entry contains a pointer to a region on the disk
  - Currently there are three FAT file system types: FAT12, FAT16, FAT32

FAT FS (2)

- **FAT filesystem organization**

- FAT file system on disk data structure is all "little endian."
- The data area is divided into clusters.
  - Used for subdirectories and files.
- Root directory region doesn't exist on FAT32.

FAT FS (3)

- **Boot sector**
  - The first sector on the disk.
  - Contains BPB (BIOS Parameter Block).
    - Sectors per cluster
    - The number of sectors on the volume.
    - Volume label.
    - The number of root directory entries.
    - File system type (FAT12, FAT16, FAT32)
    - and many more.
  - If the volume is bootable, the first sector also contains the code required to boot the OS.

FAT FS (4)

- **FAT (File Allocation Table)**
  - Starts at sector 1 (after the boot sector)
  - The FAT defines a singly linked list of the clusters of a file.
  - The first two entries in the FAT can be ignored.
    - The first entry available is entry 2.
  - The individual entries in the FAT table define the "chains" of clusters that make up a file.
  - There are two copies so that corruption of the FAT can be detected and repaired.

FAT FS (5)

- **FAT12 example**
  - Each FAT12 entry is 12bits.
    - When designed, space was tight.
    - Pack 2 entries into 3 bytes.
    - 4096 possible clusters.
    - If a sector is 512bytes and cluster = 1 sector, can represent 2MB of data.
  - FAT12 entry values:
    - 0 Unused cluster
    - 0xFF0-0xFF6 Reserved cluster
    - 0xFF7 Bad cluster
    - 0xFF8-0xFFF End of Clusterchain mark
    - Other Next cluster in file

FAT FS (6)

- **Maximum partition size allowed**

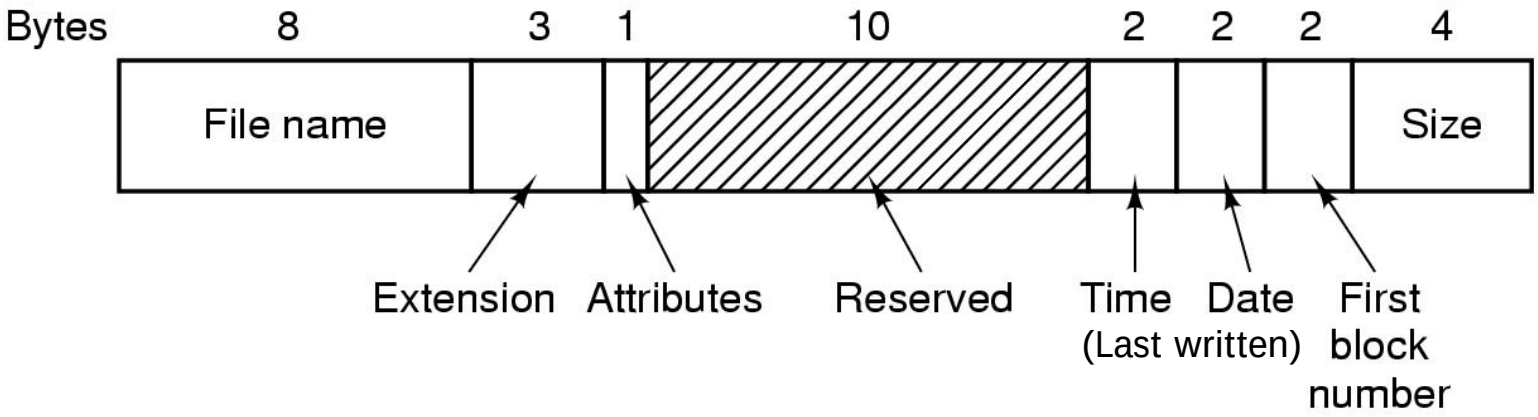
Block size	FAT-12	FAT-16	FAT-32
0.5 KB	2 MB		
1 KB	4 MB		
2 KB	8 MB	128 MB	
4 KB	16 MB	256 MB	1 TB
8 KB		512 MB	2 TB
16 KB		1024 MB	2 TB
32 KB		2048 MB	2 TB

FAT FS (7)

- **Directories**
  - The root directory is fixed in length and always located at the start of the volume (after the FAT).
    - FAT32 treats the root directory as just another cluster chain in the data area.
  - A subdirectory is nothing but a regular file that has a special attribute indicating it is a directory.
    - No size restriction
  - The data or contents of the "file" is a series of 32byte FAT directory entries.
    - Filename's first character is usage indicator:
      - » 0x00 Never been used.
      - » 0xe5 Used before but entry has been released.

FAT FS (8)

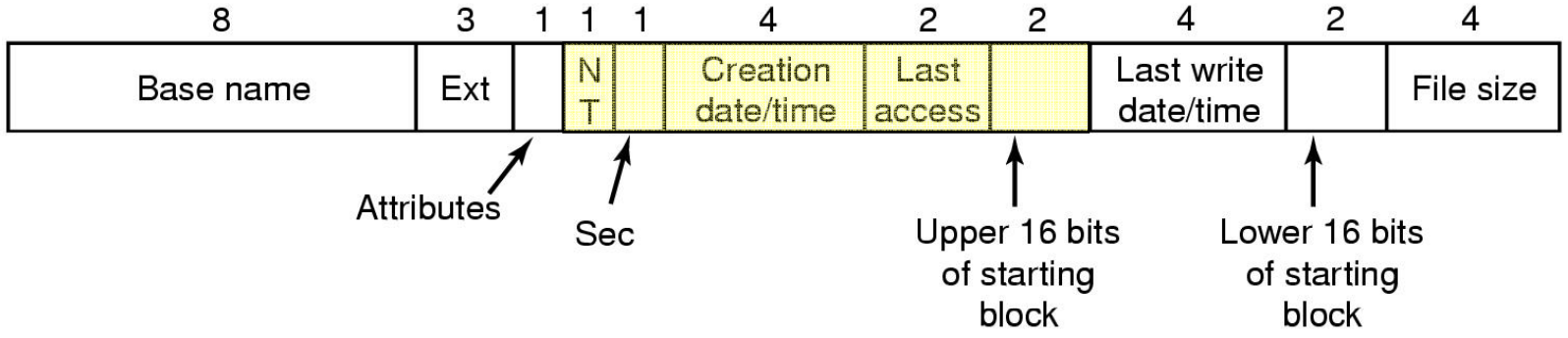
- **FAT directory entry**



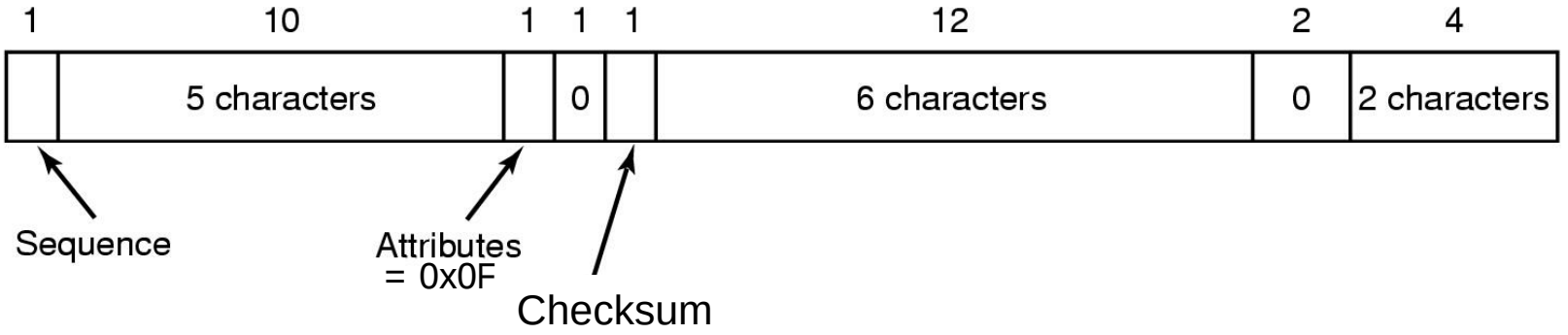
- Attributes:
  - Read Only, Hidden, System, Volume Label, Subdirectory, Archive

FAT FS (9)

- **FAT32 directory entry**



- An entry for a long file name



FAT FS (10)

- **Representing long file name in FAT32**
  - The quick brown fox jumps over the lazy dog