ToyFs -- a simple inode based file system

ToyFS Disk: □ All numbers in Little Endian format □ Blocks are 4k file system blocks □ToyFs -- Block 0 □4 bytes: TyF4 □4 bytes: Size of disk in sectors □4 bytes: Number of inodes (multiple of 64) □4 bytes: Number of blocks of data block bit map (DBBM) □4 bytes: Number of words of inode bitmap (iCnt) $\Box 4*iCnt$ bytes: Inode bitmap, $0 \Rightarrow$ free inode, $1 \Rightarrow$ used □Block 1 - DBBM □4 bytes: Number of words of data bitmap (dCnt) $\Box 4*dCnt$ bytes: Data bitmap, $0 \Rightarrow$ free inode, $1 \Rightarrow$ used \square Sector DBBM+1 - DBBM+1+M: inodes (64 inodes per sector, M = NumberOfInodes/64) □ Sector DBBM+1M+1 - end: data blocks □ Inodes and data blocks are indexed starting at 1 \square Inode 1 is the root directory

On Disk Inode structure

- \Box 2 bytes: number of links
- $\Box 2$ bytes: mode (file type, r, w, x bits)
- □4 bytes: size of file in bytes
- □4 bytes: number of data blocks allocated
- □40 bytes: 10 direct links
- □4 bytes: single indirect
- □4 bytes: double indirect (for future use)
- □4 bytes: spare (unused)
- Total size: 64 bytes => 64 inodes per block

☐ Maximum of 10 sectors for limited directory size □ Each sector contains an integral number of entries □ No entry goes across a sector boundry □ Multiple entries per sector, entry format: □4 bytes, inode number □4 bytes, name length (max 255, no '\0' terminating) \Box n+1 bytes, name □ entries are padded with 0 to 3 charaters to make word aligned entries □ entry inode number meanings: □-1: end of all directory entries □ 0: end of directory entries in this sector, more next sector □>0: actual inode number, more entries follow □ Required names: . and ..

Directory entry

Toy Filesystem and related classes

toy_filesystem class -- Methods provided

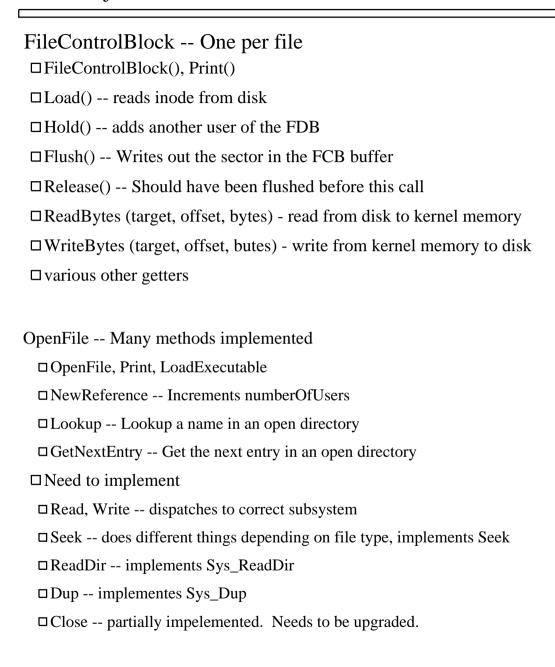
□toy_filesystem() the constructor
\square mountFS() called when we want to open the disk for use
□ saveRootandbMaps() save the core filesystem information
□openLastDir (string, workDir, lastElementLoc)
□nameToInodeNum (string, workDir)
□ AllocInode() returns inodeNum, -1 => no more left
□FreeInode (inodeNum)
□ AllocDataBlock () returns blockNum, -1 => no more left
\Box FreeDataBlock(blockNum)
□GetDiskInfo(diskInfo) implemented for you
□ Open(name, dir, flags, mode) Mostly done
□ Open: Open the file for operations
□ nameToInodeNum, GetFCB: lookup file and just get the FCB, not a full open.
□FCB still needs to be released.
☐ Should be used in Stat and ChangeMode methods. (see next page)

toy_filesystem class -- Methods you need to implement □ Read(openF, userBuf, size) □ Write(openF, userBuf, size) □ Stat (name, statBuf) □ ChangeMode (name, newmode) (CSCI 509) InodeData class (Class used by ToyFS) □ Methods of interest (not listing all getters) □ InodeData(), Print() □ WriteInode() -- writes the inode back to disk □ LoadIndSec() -- gets a frame / reads in indirect sector □ WriteIndSec() -- changes have been made to indiret block, save it ☐ Methods to Implement \square AllocateNewBlock(logicalBlock)

☐ May have to allocate a new indirect block

□ updates direct block or indirect block, save modified data

Other Objects in the Kernel



Locking in the File System □ fsLock -- filesystem operations □inodeLock -- serializes ALL inode operations as there is one lock for all inodes □each fcb has a lock □each OpenFile has a lock

Improvements still could be made to Toy FS.

