CMSC 412

GeekOS Project 5
File System

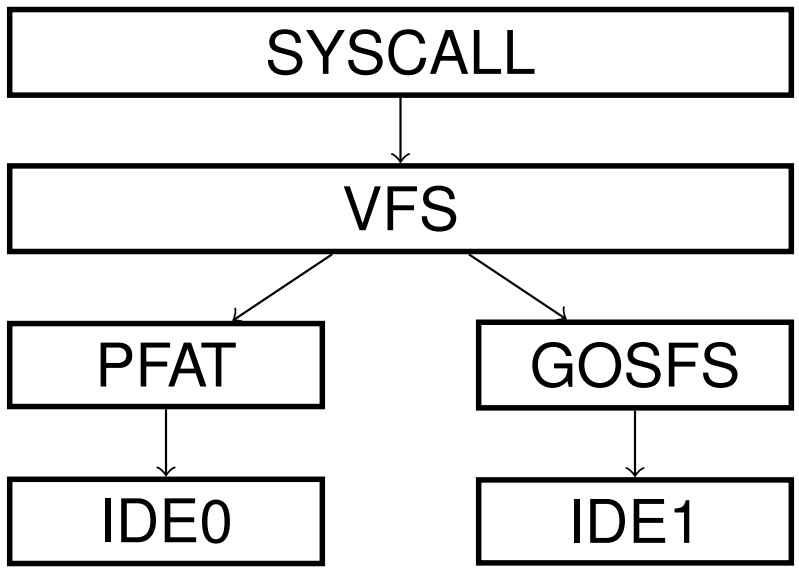
R. Gove[1]

[1]Department of Computer Science
University of Maryland

April, 2010

New File System: GOSFS

- You will implement a new file system: GOSFS
- Mount `diskd.img` as second IDE (drive `/d`)
- SYS calls go to VFS layer, VFS sends to PFAT or GOSFS



GOSFS disk layout

- GOSFS uses 4KB blocks. Block 0 is the super block. It contains:
  - Magic number, index of root directory, size of disk (num FS blocks, disk blocks are 512 bytes), bitmap of free blocks

<table>
<tr><td colspan="5">SUPERBLOCK</td><td>REST OF THE DISK</td></tr>
<tr><td colspan="5">4KB</td><td>32 MB-4KB</td></tr>
<tr><td>4 bytes</td><td>4 bytes</td><td>4 bytes</td><td>1024 bytes</td><td>4KB-1036 bytes= 3060 bytes</td><td>32 MB-4KB</td></tr>
<tr><td>Magic</td><td>Root Dir Pointer</td><td>Size</td><td>Free Blocks Bitmap</td><td>Free</td><td>Data
(Not allocated blocks or blocks allocated to directories/files)</td></tr>
</table>

Format a drive with GOSFS

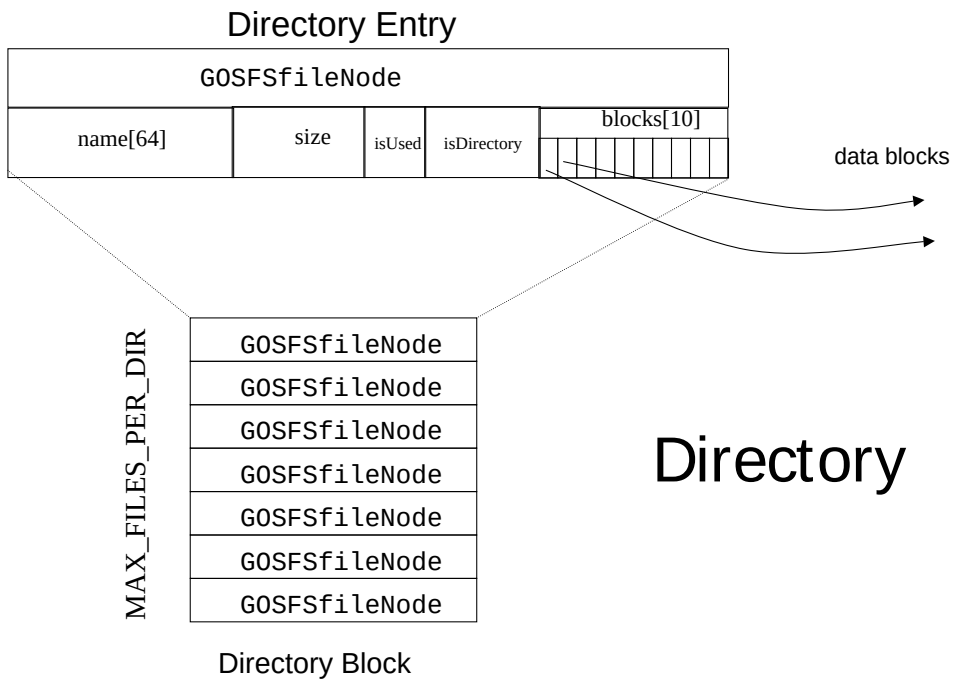
- `GOSFS_Format()`
  - Use `Get_Num_Blocks()` (convert to GOSFS blocks)
  - Use `Block_Read()` and `Block_Write()` to do disk I/O.
  - Do all changes in memory buffer(s), then write change(s) to the disk block(s)
- `GOSFS_Mount()`
  - Read the disk, check the magic num
  - Make struct to hold FS data in `mountPoint->fsData`
  - `Mount_Point_Ops` struct to hold pointers to FS functions (`mountPoint->ops`)
- Test Format and Mount with:

```
$ format ide1 gosfs
$ mount ide1 /d gosfs
```

- This will format and then mount `/d` (aka `diskd.img`)

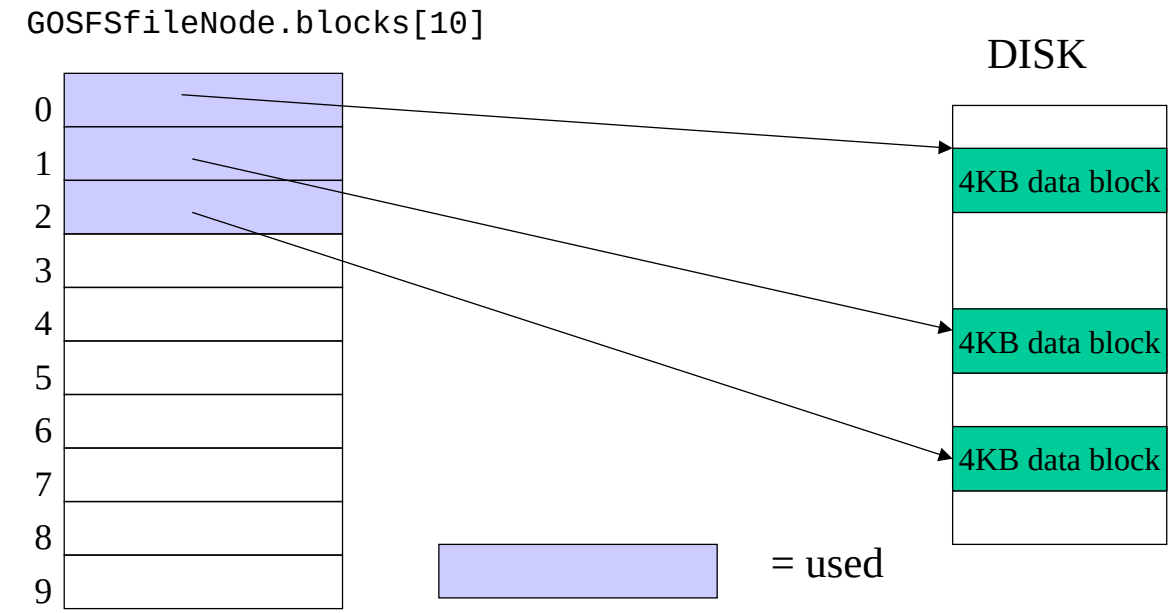
Files and directories

- Files and directories represented by `GOSFSfileNode`, stored in a single FS block.
- Files: `blocks[]` points to the FS blocks that hold the data
- Directories: `blocks[0]` points to the FS block that holds `GOSFSdirectory` (array of files).
- The root directory `GOSFSfileNode` is probably in block 1



Direct mapping

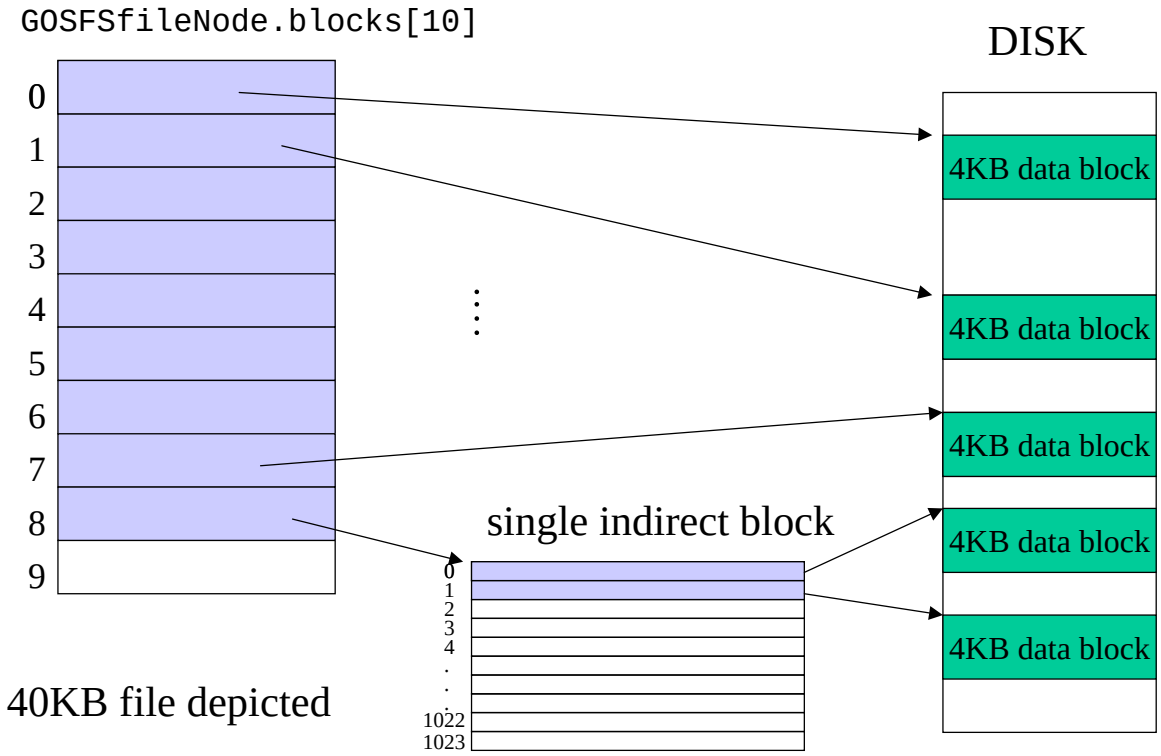
- `blocks[0]` through `blocks[7]` use direct mapping
- $0 \leq pos < 32768$



12KB file depicted

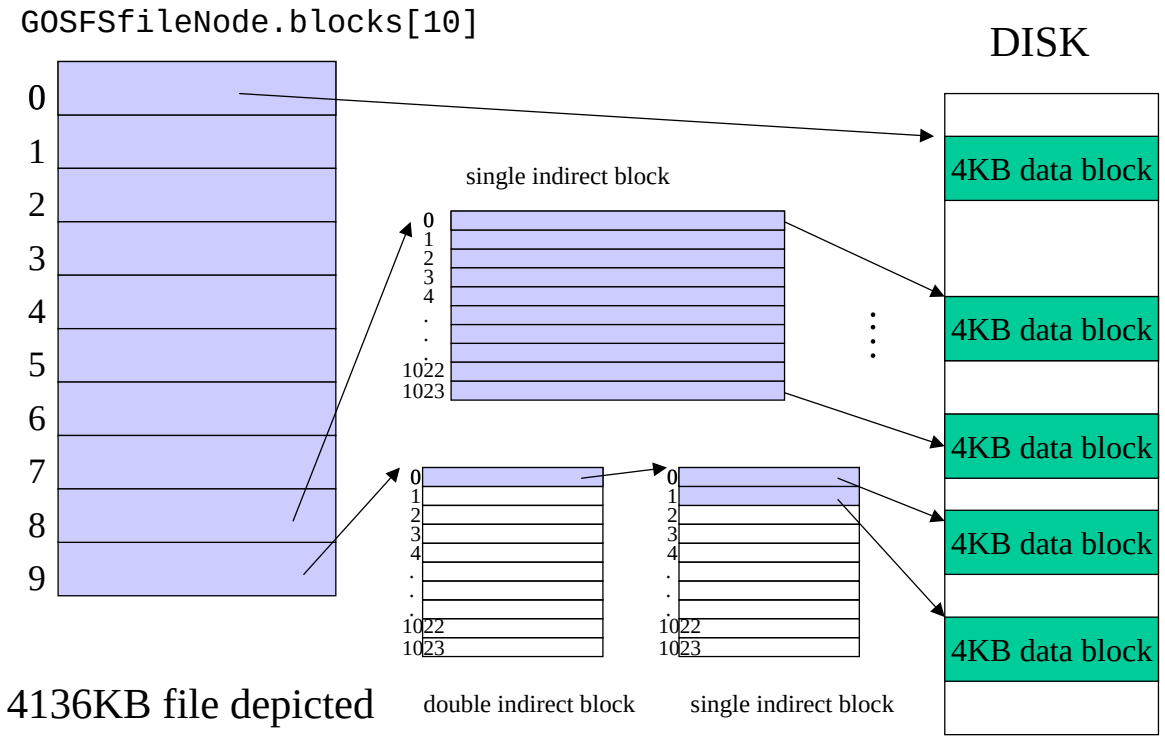
Single indirect mapping

- `blocks[8]` uses single indirect mapping
- $32768 \leq pos < 4227072$



Double indirect mapping

- `blocks[9]` uses double indirect mapping
- $pos \geq 4227072$



GOSFS implementation

- System calls are already implemented
- User threads have a new array: `struct File *fileList[USER_MAX_FILES]` of open files, indexed by file descriptor. Syscalls that take file descriptors use it.

GOSFS functions

- `GOSFS_Mount()`: Make sure it works!
- `GOSFS_FStat()`: Passed a FD to `Sys_FStat()`, and VFS gave us `File` struct. Get the file node info from disk, put it in `VFS_File_Stat`
- `GOSFS_Open()`: Search for file on disk by path. Set file permissions (mode: `O_CREATE|O_READ|O_WRITE`), create file if appropriate (and populate `File` struct). Use `GOSFS_Open_Directory()` for directories.
- `GOSFS_Delete()`: Don't delete non-empty directories.

GOSFS functions

- `GOSFS_Write()`: “Grow on write,” so allocate blocks on the fly as the file grows. Cannot use on directories.
- `GOSFS_Read()`: Return num bytes read. Not for directories.
- Both read and write need to increase the file position.

GOSFS functions

- `GOSFS_Create_Directory()`: Create directory. Can be recursive: `/d/d1/d2/d3/d4`
- `GOSFS_Open_Directory()`: Open a directory, create if need be. Do not create recursively: `/d/d1/d2/d3/d4`
- `GOSFS_Read_Entry()`: For directories. Read the `File` directory struct, make it point to next in directory.

GOSFS functions

- `GOSFS_Seek()`: `pos` is an absolute position. Not allowed past the end of the file.
- `GOSFS_Sync()`: Might not need it.

GOSFS misc

- Need to support disks at least 32MB (i.e. 1024 bytes for block allocation in superblock)
- Need to support files at least 5MB (i.e. you need to support double indirection)
- `user/p5test.c` will test your implementation.
- Check project spec for return values and reasons for failure (some failures are handled in the SYS calls)

GOSFS misc

- `bitset` will be helpful for managing FS disk blocks bitmap
- It might be helpful to create a `GOSFSsuperblock` struct to initialize the superblock (or maybe not)
- Can change the size of `diskd.img` in `Makefile.common` (10MB by default)
- Need to have `File_Ops` struct instances to use for files and directories