9 Days left...

• of your favorite class ever
• of the best time of you life
• of the most you’ve ever worked in your life
• to complete project 6
• with the coolest TA ever (jk – or am I?)
Project 6 - Immediate Files
### Why?

<table>
<thead>
<tr>
<th>File length</th>
<th>Percentage</th>
<th>File length</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.79</td>
<td>1024</td>
<td>48.05</td>
</tr>
<tr>
<td>2</td>
<td>1.88</td>
<td>2048</td>
<td>60.87</td>
</tr>
<tr>
<td>4</td>
<td>2.01</td>
<td>4096</td>
<td>73.51</td>
</tr>
<tr>
<td>8</td>
<td>2.31</td>
<td>8192</td>
<td>84.97</td>
</tr>
<tr>
<td>16</td>
<td>3.32</td>
<td>16384</td>
<td>92.53</td>
</tr>
<tr>
<td>32</td>
<td>5.13</td>
<td>32768</td>
<td>97.21</td>
</tr>
<tr>
<td>64</td>
<td>8.71</td>
<td>65536</td>
<td>99.18</td>
</tr>
<tr>
<td>128</td>
<td>14.73</td>
<td>131072</td>
<td>99.84</td>
</tr>
<tr>
<td>256</td>
<td>23.09</td>
<td>262144</td>
<td>99.96</td>
</tr>
<tr>
<td>512</td>
<td>34.44</td>
<td>524288</td>
<td>100.00</td>
</tr>
</tbody>
</table>

Inside Minix
v1, v2, v3 files

- v1 files are for older files -- ignore
- **v2 files are what this version of minix creates**
- v3 files don’t exist, however there are a few comments about them -- ignore
servers/mfs/inode.h

EXTERN struct inode {
    mode_t i_mode;       /* file type, protection, etc. */
    nlink_t i_nlinks;    /* how many links to this file */
    uid_t i_uid;         /* user id of the file's owner */
    gid_t i_gid;         /* group number */
    off_t i_size;        /* current file size in bytes */
    time_t i_atime;      /* time of last access (V2 only) */
    time_t i_mtime;      /* when file data last changed */
    time_t i_ctime;      /* when was inode itself changed */
    zone_t i_zone[V2_NR_TZONES]; /* zone numbers */

    ...  
    <remainder of struct not saved on disk>
}

include/minix/const.h

• Defines constants used by mfs
  ▫ I_REGULAR – regular file
  ▫ I_TYPE – mask for file type
• Note: These are used in ushorts (2 bytes)
• Suggestion: Add an I_IMMEDIATE that fits in ushort and doesn’t conflict with the masks
Constants

- I_TYPE 0170000 /* inode type */
- I_SYMBOLIC_LINK 0120000 /* symbolic link */
- I_REGULAR 0100000 /* regular file */
- #define I_BLOCK_SPECIAL 0060000 /* block special file */
- #define IDIRECTORY 0040000 /* file is a directory */
- #define I_CHAR_SPECIAL 0020000 /* character special file */
- #define I_NAMED_PIPE 0010000 /* named pipe (FIFO) */
- #define I_SET_UID_BIT 0004000 /* set effective uid_t */
- #define I_SET_GID_BIT 0002000 /* set effective gid_t */
- #define ALL_MODES 0006777 /* all bits for u,g,o */
- #define RWX_MODES 0000777 /* mode bits for RWX only */
- #define R_BIT 0000004 /* Rwx protection bit */
- #define W_BIT 0000002 /* rWx protection bit */
- #define X_BIT 0000001 /* rwX protection bit */
- #define I_NOT_ALLOC 0000000 /* this inode is free */
Adding Files

- Set immediate flag whenever a regular file is initially created.
- Suggestion: Trace all the places where files can be created back to common code.
- Hint: Somewhere in servers/mfs/open.c
Deleting Files

- When files are deleted typically indirect blocks need to be freed
- Skip this step if immediate
- Suggestion: As before trace the few places that perform this behavior to the common location
- Hint: servers/vfs/link.c
Writing Files

- When file size grows beyond 34 bytes switch to “normal” method
Reading Files

- If immediate read from inode
- If not read as normal
How to start

• Step 1: Successfully set immediate bit, and put checks on open/read/write/delete when an immediate file is encountered.
• Step 2: Implement the immediate file

• Warning: Make regular backups of your minix image, as you might destroy it