CS170 Discussion – 2009-05-01

Bryce Boe

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?

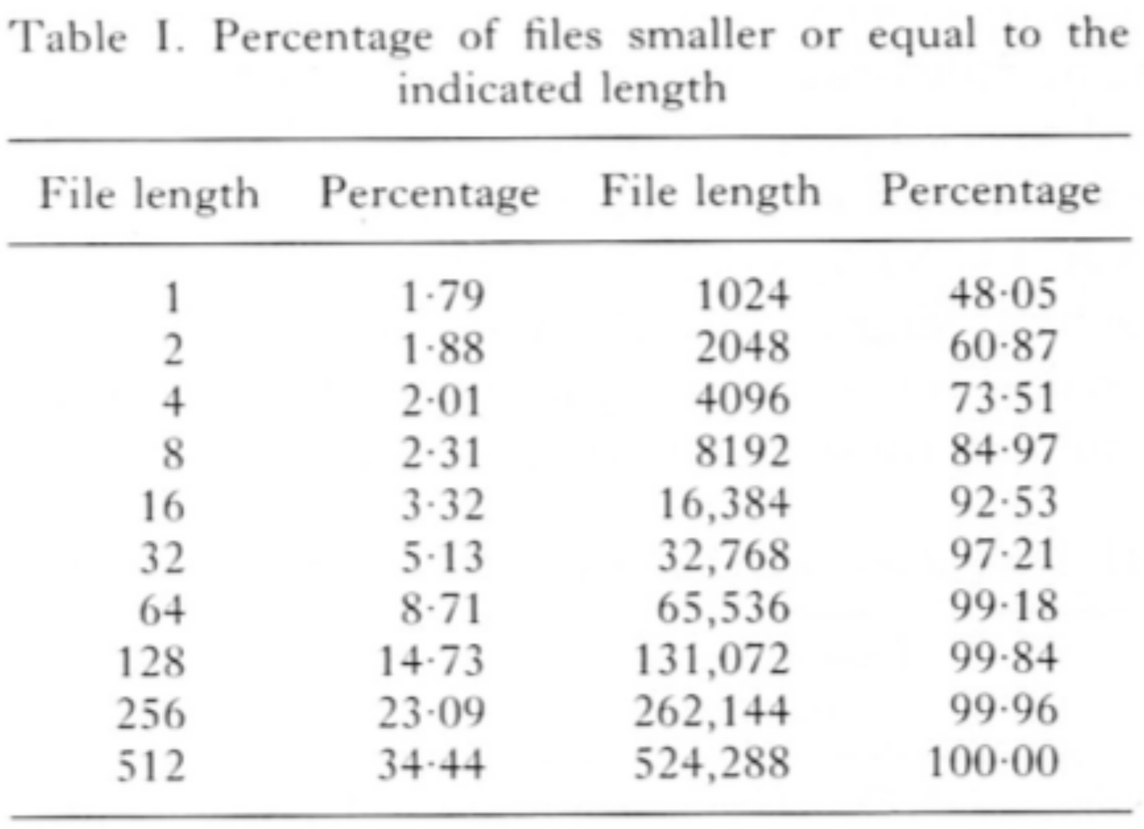

Table I. Percentage of files smaller or equal to the indicated length

File length	Percentage	File length	Percentage
1	1·79	1024	48·05
2	1·88	2048	60·87
4	2·01	4096	73·51
8	2·31	8192	84·97
16	3·32	16,384	92·53
32	5·13	32,768	97·21
64	8·71	65,536	99·18
128	14·73	131,072	99·84
256	23·09	262,144	99·96
512	34·44	524,288	100·00

Mullender, S. J. and Tanenbaum, A. S. 1984. Immediate files. Softw. Pract. Exper. 14, 4 (Jun. 1984), 365-368. DOI= http://dx.doi.org/10.1002/spe.4380140407

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 I_DIRECTORY       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