Project 3 Overview

- Immediate Files
- Immediate Files in Minix
- List Resources “lsr” System Call

Immediate Files

- What is an immediate file?
  - A file where the entire data contents are stored in the inode block containing file meta-data
- Why would we want an immediate file?
  - Minimizes disk waste for small files
  - Maximizes performance for small files

More on Why (#1)

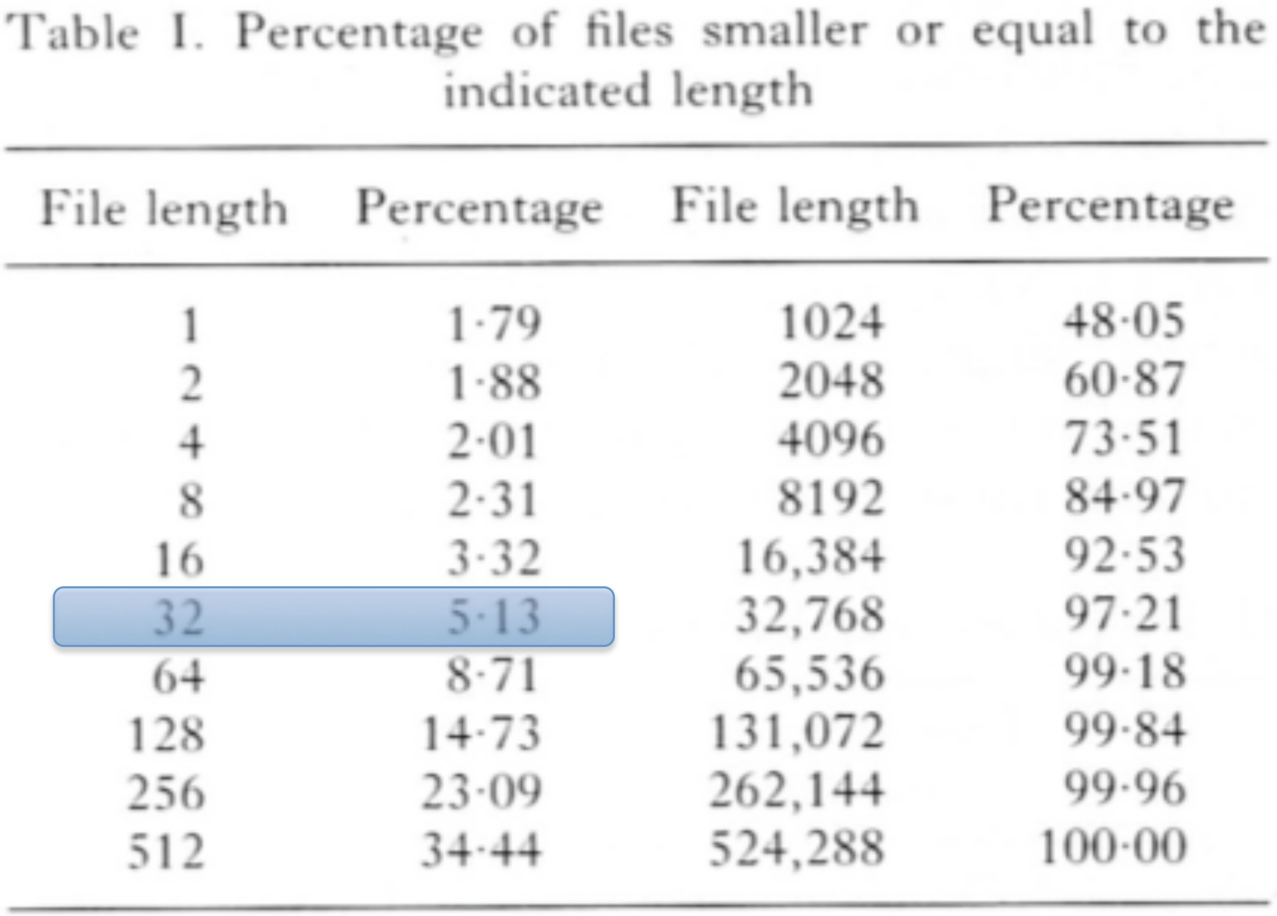


Table I. Percentage of files smaller or equal to the indicated length

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

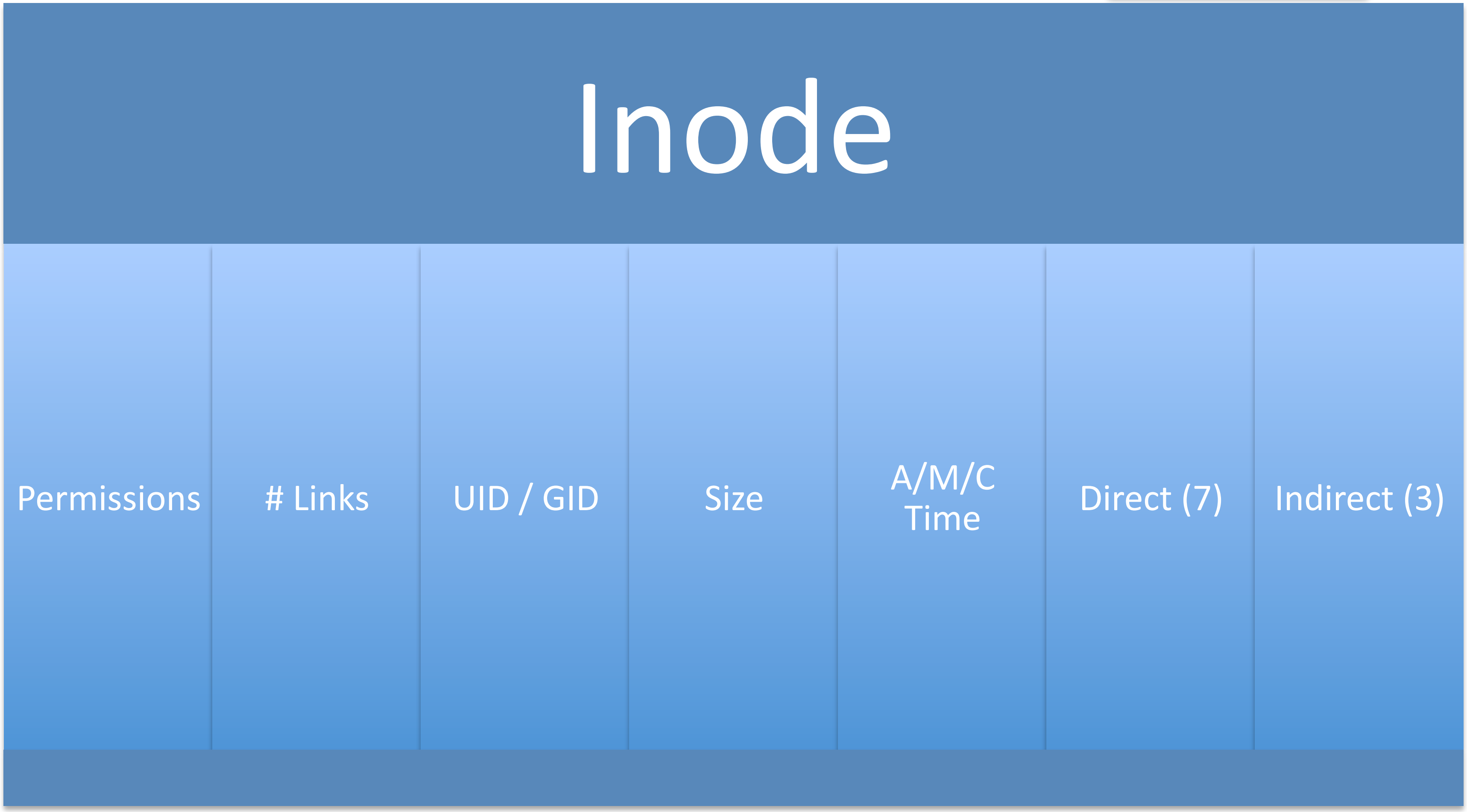
Immediate Files in Minix

- Inode structure: mfs/inode.h
- Constants: mfs/const.h
- Syscalls to modify
  - Create / Unlink
  - Open / Truncate
  - Read / Write
  - Others?

Minix FS Services

- User processes interact with the virtual file system (VFS)
- VFS talks with the appropriate file system service such as ext2, hgfs, iso9660fs, mfs, pfs
- If you took the default, this will be mfs for you
- **Make sure this is MFS for you**

servers/mfs/inode.h



Inode

Permissions

Links

UID / GID

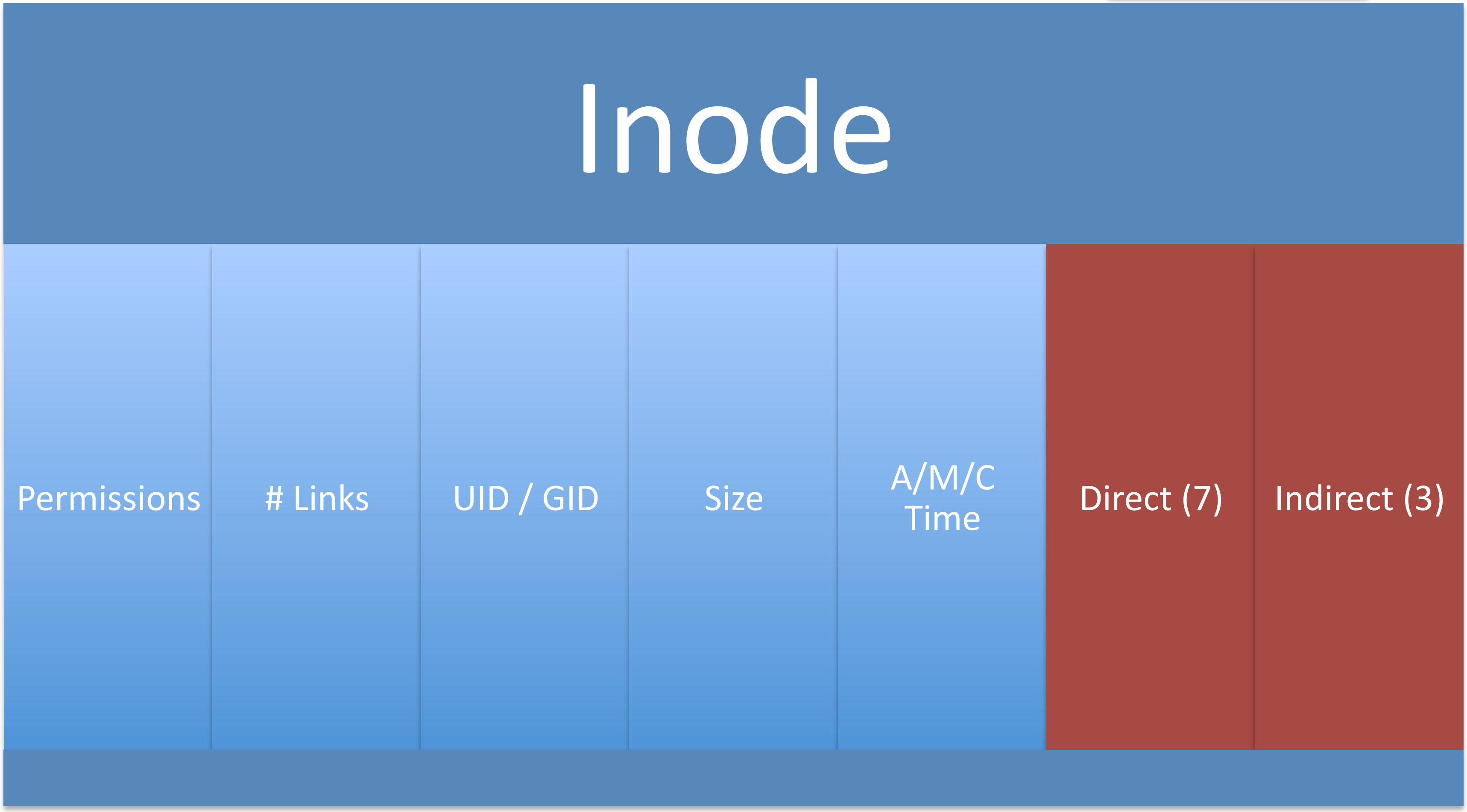
Size

A/M/C Time

Direct (7)

Indirect (3)

servers/mfs/inode.h



Inode

Permissions

Links

UID / GID

Size

A/M/C Time

Direct (7)

Indirect (3)

Another view

```
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>
}

#define V2_NR_DZONES  7 /* # direct zone numbers in ... */
#define V2_NR_TZONES 10 /* total # zone numbers in ... */
```

Block Pointers

- 7 direct block pointers take up 28 bytes
- 3 indirect block pointers take up 12 bytes
- 40 bytes of each inode used for pointers
  - Let's use it for data! Yay!!!

HOW TO DISTINGUISH BETWEEN REGULAR AND IMMEDIATE?

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 existing masks

Constants used for i_mode

???

```
•  #define I_TYPE           0170000 /* inode type */
•  #define I_SYMBOLIC_LINK  0120000 /* symbolic link */
•  #define 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 */
```

IMPLEMENTATION TIPS

Tip: How to start

- Step 1: Successfully set immediate bit on creation, 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

Tip: Adding Files

- Set immediate flag whenever a regular file is initially created
- Suggestion: Find all places where files can be created in the MFS.

Tip: 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 in the MFS.

Tip: Reading / Writing Files

- If immediate read from inode otherwise read as regular
- When file size grows beyond 34 bytes convert to regular file

What are 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

List Resources System Call

- int lsr(char *path);
- Path can be absolute or relative
- Must Output
  - All process ids that have the file open
  - All blocks on disk that contain the file contents
    - If immediate list “immediate”
    - If empty list “empty”
  - If the file doesn’t exist, return ENOENT

References

1. 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