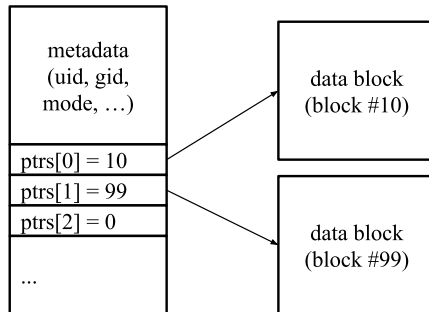
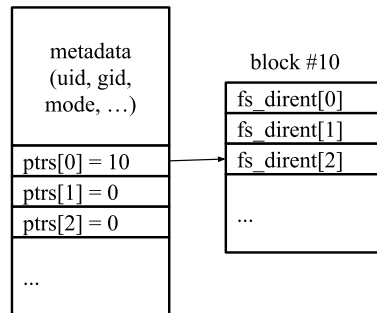


1. fs5600 visualization

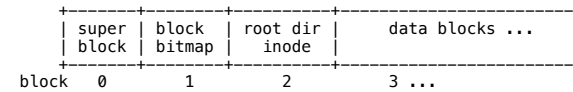
fs5600 file inode:



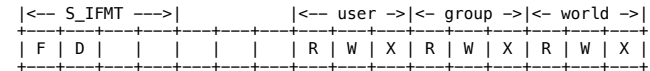
fs5600 directory (also an inode):



2. fs5600 block layout



3. fs5600 inode mode



4. interfaces

fs_init - constructor (i.e. put your init code here)
 fs_statfs - report file system statistics
 fs_getattr - get attributes of a file/directory
 fs_readdir - enumerate entries in a directory
 fs_read - read data from a file
 fs_rename - rename a file
 fs_chmod - change file permissions

fs_create - create a new (empty) file
 fs_mkdir - create new (empty) directory
 fs_unlink - remove a file
 fs_rmdir - remove a directory
 fs_write - write to a file
 fs_truncate - delete the contents of a file
 fs_utime - change access and modification times

```

1 CS5600 file system
2
3 // 1. Read/write disk in Lab4 (CS5600 file system)
4 // borrowed from Lab4, fs5600.c
5
6 /* disk access.
7  * All access is in terms of 4KB blocks; read and
8  * write functions return 0 (success) or -EIO.
9  *
10 * read/write "nblks" blocks of data
11 * starting from block id "lba"
12 * to/from memory "buf".
13 * (see implementations in misc.c)
14 */
15 extern int block_read(void *buf, int lba, int nblks);
16 extern int block_write(void *buf, int lba, int nblks);
17
18
19 /*
20 * below are two toy examples of
21 * reading from and writing to a disk block
22 */
23
24 // Reading one block
25
26 char buf[FS_BLOCK_SIZE]; // FS_BLOCK_SIZE=4096; see panel 2
27 int inum = 100; // block number to read from
28 int ret = block_read(&buf, inum, 1);
29 if (ret < 0) { // error; ret should be -EIO
30     return ret;
31 }
32
33
34 // Writing one block
35
36 char buf[FS_BLOCK_SIZE]; // again, 4KB buffer
37 ... // update buf
38
39 int ret = block_write(&buf, 99, 1); // update the 99th block
40 if (ret < 0) { // error; ret should be -EIO
41     return ret;
42 }
43
44

```

```

45
46
47 // 2. CS5600 file system data structures
48 // borrowed from fs5600.h with minor changes
49
50 /*
51  * file: fs5600.h
52  * description: Data structures for CS5600 file system.
53  *
54  * CS 5600, Computer Systems, Northeastern CCIS
55  * Peter Desnoyers, November 2016
56  *
57  * Modified by CS5600 staff in fall 2021.
58 */
59
60 #define FS_BLOCK_SIZE 4096
61 #define FS_MAGIC 0x30303635
62
63 #define INODE_NUM_PTRS (FS_BLOCK_SIZE/4 - 5)
64
65 /* Superblock - holds file system parameters.
66 */
67 struct fs_super {
68     uint32_t magic;
69     uint32_t disk_size; // in blocks */
70
71     /* pad out to an entire block */
72     char pad[FS_BLOCK_SIZE - 2 * sizeof(uint32_t)];
73 };
74
75
76 /* inode = 4096 bytes */
77 struct fs_inode {
78     uint16_t uid;
79     uint16_t gid;
80     uint32_t mode;
81     uint32_t ctime;
82     uint32_t mtime;
83     int32_t size;
84     uint32_t ptrs[INODE_NUM_PTRS];
85 };
86
87
88 /* Entry in a directory
89 */
90 struct fs_dirent {
91     uint32_t valid : 1;
92     uint32_t inode : 31;
93     char name[28]; // with trailing NUL */
94 };

```