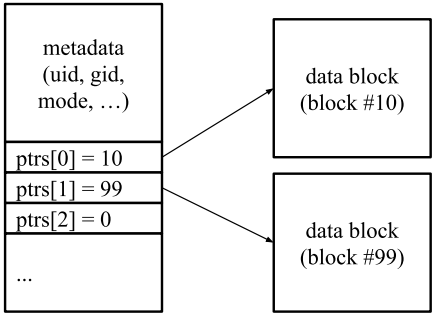
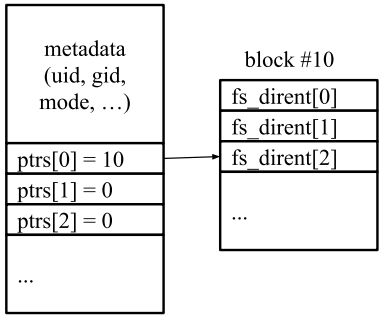
1. fs3650 visualization

fs3650 file inode:



fs3650 directory (also an inode):



2. fs3650 block layout

```
     +-------+----------+-------------------------+
     | super | root dir |     data blocks ...     |
     | block |   inode  |                         |
     +-------+----------+-------------------------+
block    0        1         2          3 ...
```

3. fs3650 inode mode

```
    |<-- S_IFMT --->|           |<-- user ->|<- group ->|<- world ->|
    +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
    | F | D |   |   |   |   |   | R | W | X | R | W | X | R | W | X |
    +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

4. interfaces

(a) fs_getattr – get attributes of a file/directory

```
    int fs_getattr(const char *path, struct stat *sb,
                   struct fuse_file_info *fi)
```

(b) fs_readdir – enumerate entries in a directory

```
    int fs_readdir(const char *path, void *ptr, fuse_fill_dir_t filler, off_t offset,
                   struct fuse_file_info *fi, enum fuse_readdir_flags flags)
```

(c) fs_read – read data from a file

```
    int fs_read(const char *path, char *buf, size_t len, off_t offset,
                struct fuse_file_info *fi)
```

```
CS3650 file system

// 1. Read/write disk in Lab4 (CS3650 file system)
// borrowed from Lab4, homework.c

   /* disk access. All access is in terms of 4KB blocks; read and
    * write functions return 0 (success) or -EIO.
    */
   extern int block_read(void *buf, int lba, int nblks);
   extern int block_write(void *buf, int lba, int nblks);

   /* below is a toy example of reading from a disk block
    */

   char buf[FS_BLOCK_SIZE];   // FS_BLOCK_SIZE=4096
   int bnum = 100;            // block number to read from
   int ret = block_read(&buf, bnum, 1);
   if (ret < 0) {  // error; ret should be -EIO
       return ret;
   }


// 2. CS3650 file system data structures
// borrowed from fs3650.h with minor changes


   #define FS_BLOCK_SIZE 4096
   #define FS_MAGIC 0x33363530

   #define MAX_PATH_NAMES 20    /* max depth of a path */
   #define MAX_PATH_BYTES 1024  /* max length of a path */

   /* how many buckets of size M do you need to hold N items?
    */
   #define DIV_ROUND_UP(N, M) ((N) + (M) - 1) / (M)

   /* Entry in a directory
    */
   struct fs_dirent {
       uint32_t valid : 1;
       uint32_t inode : 31;
       char name[28];              /* with trailing NUL */
   };

   /* Superblock - holds file system parameters.
    */
   struct fs_super {
       uint32_t magic;
       uint32_t disk_size;         /* in blocks */
       uint32_t root_inode;        /* block number */

       /* pad out to an entire block */
       char pad[FS_BLOCK_SIZE - 2 * sizeof(uint32_t)];
   };

```



```
   struct fs_inode {
       uint16_t uid;
       uint16_t gid;
       uint32_t mode;
       uint32_t ctime;
       uint32_t mtime;
       int32_t  size;
       uint32_t ptrs[FS_BLOCK_SIZE/4 - 5]; /* inode = 4096 bytes */
   };


// 3. "struct stat" from Linux
// borrowed from "man fstat"

   struct stat {
       dev_t     st_dev;         /* ID of device containing file */
       ino_t     st_ino;         /* Inode number */
       mode_t    st_mode;        /* File type and mode */
       nlink_t   st_nlink;       /* Number of hard links */
       uid_t     st_uid;         /* User ID of owner */
       gid_t     st_gid;         /* Group ID of owner */
       dev_t     st_rdev;        /* Device ID (if special file) */
       off_t     st_size;        /* Total size, in bytes */
       blksize_t st_blksize;     /* Block size for filesystem I/O */
       blkcnt_t  st_blocks;      /* Number of 512B blocks allocated */

       /* Since Linux 2.6, the kernel supports nanosecond
          precision for the following timestamp fields.
          For the details before Linux 2.6, see NOTES. */

       struct timespec st_atim;  /* Time of last access */
       struct timespec st_mtim;  /* Time of last modification */
       struct timespec st_ctim;  /* Time of last status change */

   #define st_atime st_atim.tv_sec      /* Backward compatibility */
   #define st_mtime st_mtim.tv_sec
   #define st_ctime st_ctim.tv_sec
   };
```