Quinnipiac File System (QFS)

Introduction

This semester we have learned about file system concepts such as file access tables, superblocks, directories, and file storage methods. The Quinnipiac File System (QFS) is a simple implementation that demonstrates these concepts in practice. This file system is designed for small disks/images (up to 120MB in size).

This document outlines the structure and components of QFS. QFS is designed to be as straightforward as possible while still illustrating the key principles of file system design.

Disk Structure

A QFS disk is divided into several key sections:

- **Superblock:** Contains metadata about the file system, such as block size, total number of blocks, free block count, total number of directory entries, and number of free directory entries.
- **Directory Entries:** A fixed-size table that holds information about files and directories.
- **File Data Blocks:** The area where file data is stored.

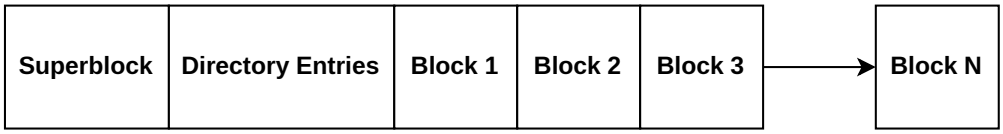

Figure 1: QFS Disk Structure

Figure 1 illustrates the overall structure of a QFS disk. The superblock is located at the beginning of the disk, followed by the directory entries, and finally the data blocks. The superblock and directory entries have a fixed size for simplicity.

The remaining parts of this document will go into more detail about each component.

Superblock Structure

The superblock starts at position 0 of the QFS disk and contains essential metadata about the file system. It includes the following fields:

```
uint8_t   fs_type;              // File system type/Magic number (0x51 for QFS)
uint16_t  total_blocks;         // Total number of blocks
uint16_t  available_blocks;     // Number of blocks available
uint16_t  bytes_per_block;      // Number of bytes per block
uint8_t   total_direntries;     // Total number of directory entries
uint8_t   available_direntries; // Number of available dir entries
uint8_t   reserved[8];          // Reserved, all set to 0
char      label[15];            // NULL-terminated volume name (optional)
```

Figure 2 shows the layout of the superblock. Note that the size of each section of the superblock is fixed, and the sizes (in bytes) are shown in the figure. The total size of the superblock is 32 bytes. Here are the field definitions:

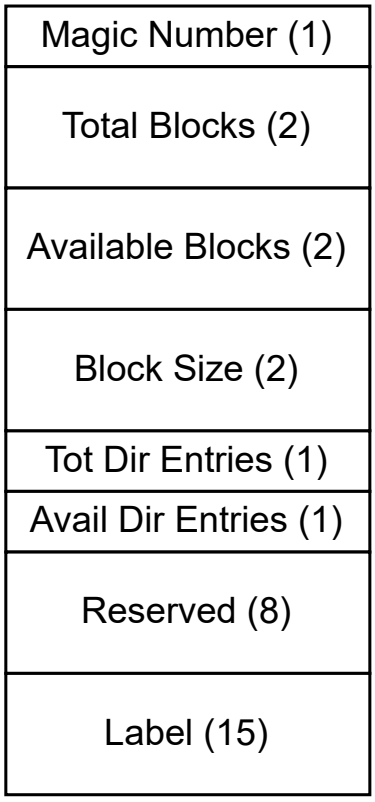

Figure 2: QFS Superblock Structure

The reserved field is included for potential future use and should be initialized to zero. The volume name is a NULL-terminated string with a maximum length of 15 characters (14 plus the NULL terminator). This field is optional. The magic number for QFS is 0x51 (ASCII 'Q').

Directory Entry Structure

Each directory entry in QFS represents a file or directory and contains the following fields:

```
    char     filename[23];      // NULL-terminated
    uint8_t  permissions;       // File permissions/type
    uint8_t  owner_id;          // Owner ID
    uint8_t  group_id;          // Group ID
    uint16_t starting_block;    // Starting block number
    uint32_t file_size;         // Size of the file in bytes
```

Figure 3 shows the layout of a directory entry. Each directory entry is 32 bytes in size.

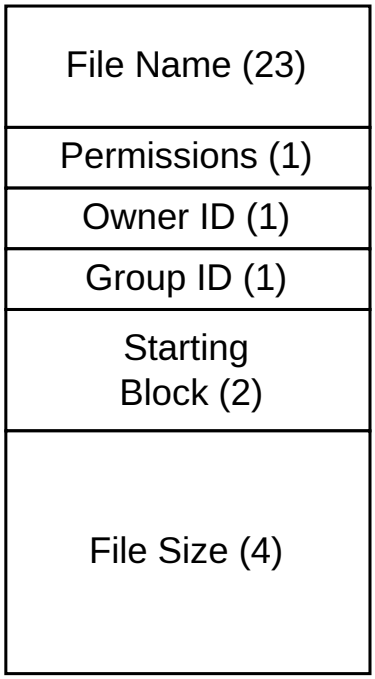

Figure 3: QFS Directory Entry Structure

A directory entry is free/available if its filename field is empty (i.e., the first character is the NULL terminator). Because there are a maximum of 255 directory entries, the directory entry table size is fixed at 8,160 bytes (255 entries * 32 bytes per entry). The directory entries are stored immediately after the superblock, starting at byte offset 32. Owner and group IDs follow the general Unix process, and are limited to 0 through 255. This file system is meant to store files only, so there is no execute permission. The 8 bits for the permission field are therefore:

User Permission Bits [0:1]
Group Permission Bits [2:3]
World Permission Bits [4:5]
File Type Bits [6:7]

Blocks

Blocks are the fundamental units of data storage in QFS. Each block has a fixed size, defined in the superblock by the `bytes_per_block` field. The default block size is determined by the size of the disk (not including the superblock and directory entry table). The following list shows the block sizes based on this:

- $size <= 30MB$ (31457280 bytes): 512 bytes per block
- $30MB < size <= 60MB$ (62914560 bytes): 1024 bytes per block
- $60MB < size <= 120MB$ (125829120 bytes): 2048 bytes per block

Blocks are numbered starting with 0 and start immediately after the directory entry table, at offset 8192 (32 + 8160). The Table 1 shows the starting offsets for blocks based on the default block sizes:

Block Size	Block Number	Starting Offset
512 bytes	5	10,752 bytes $(8192 + 512 * 5)$
1024 bytes	8	16,384 bytes $(8192 + 1024 * 8)$
2048 bytes	2	12,288 bytes $(8192 + 2048 * 2)$

Table 1: Disk Offsets for Blocks Based on Block Size

The first byte of each block indicates whether the block is in use (1) or free (0). The blocks are used to store file data, and files can span multiple blocks if necessary. If a file is larger than a single block, each block will be linked to the next block in the file's data chain. The starting block of a file is indicated in its directory entry. Each block not containing the end of a file will have a 16-bit pointer at the end indicating the block number of the next block in the file.

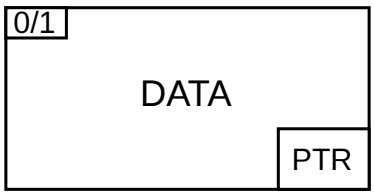

Figure 4: Block Structure

Figure 4 illustrates the structure of a block. The following fields could be defined in a struct:

```
1    uint8_t  is_busy;       // Free/busy byte (1 = free, 0 = busy)
2    uint8_t  *data;         // Data area (of size bytes_per_block - 3)
3    uint16_t next_block;    // Next block number (if applicable)
```

File Storage

As mentioned above, files in QFS are stored in blocks, and each file can occupy multiple blocks if its size exceeds the block size. The file data is stored sequentially across (possibly non-contiguous) blocks, with each block containing a pointer to the next block in the file's data chain. The size of a file is recorded in its directory entry. In this case the pointer is a 16-bit unsigned integer located at the end of each block containing the block number of the next block in the file.

When a file is created, the file system allocates the necessary number of blocks to store the file's data. Free blocks are identified by the first byte in the block (1 is free, and 0 is busy). When a file is deleted, the blocks it occupied have their first bytes set to 1 and the corresponding directory entry is cleared.

Figure 5 illustrates how a regular file is stored (test.png) across multiple blocks in QFS. Assuming the block size is 512 bytes, a file of size 1256 bytes would occupy three blocks. The first block would contain the first 509 bytes of data and the block number the second block, the second block would contain the next 509 bytes and the block number of the third block, and the third block would contain the remaining 238 bytes of data. This figure includes the directory entry for the file, which indicates its starting block and size.

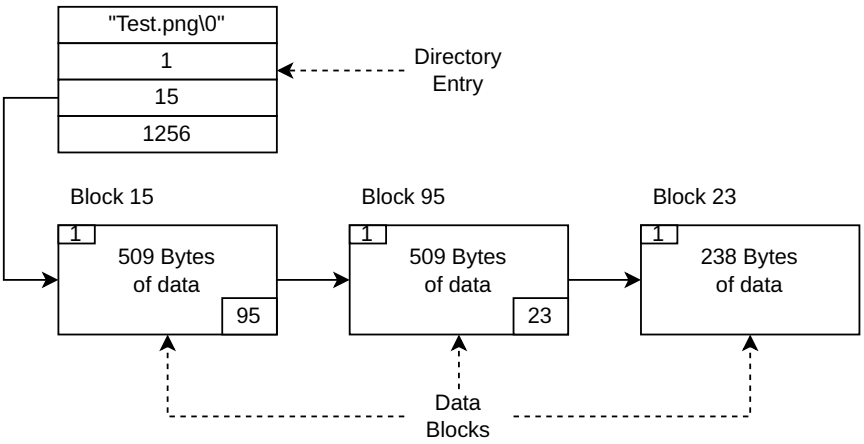

Figure 5: QFS File Storage Across Blocks