CI583: Data Structures and Operating Systems
File systems: S5FS

S5FS

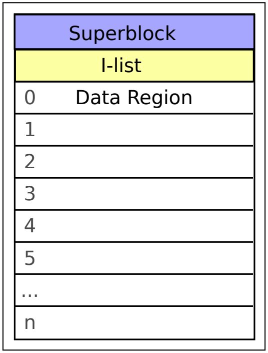
Typically of Thompson and Ritchie, the early UNIX file system was elegantly simple.

S5FS was the version shipped with version 5 of UNIX. Files can be accessed efficiently whether in sequential or random order.

Performance and robustness, however, are not sufficient for modern use. (Improving it at the time would have been premature optimisation – at a time when processors handled tens or hundreds of instructions per second, rather than millions, S5FS was fast enough.)

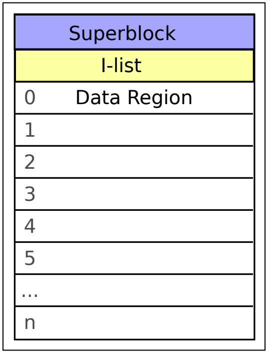
S5FS

In an S5FS disk, the first block after the boot block is occupied by the superblock, which describes the layout of the rest of the disk and contains pointers to the heads of various lists used to manage free space.



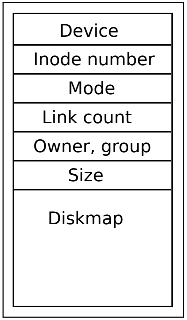
S5FS

Next is a block(s) containing an array of inodes, each of which describes a file or is free. Next is the data region.



S5FS

Each occupied inode describes a file. Directories are files containing pairs of pathnames (e.g. `/home/jim/.emacs`) and inode numbers, which are indexes into the i-list.



S5FS

The most interesting part of the indode is the disk map, which maps logical block numbers to physical blocks. Each block is 1024 bytes long. The first ten entries map directly to the first ten blocks (10kB) in the file.

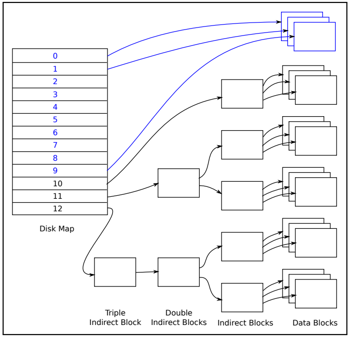
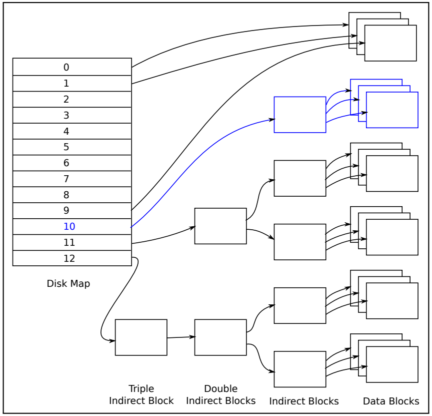


Figure adapted from Doeppner, *Operating Systems in Depth*.

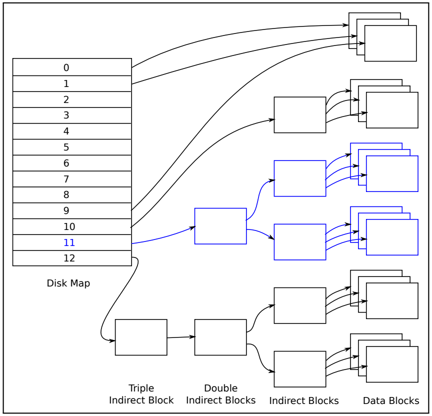
S5FS

The next entry in the disk map maps to an indirect block containing up to 256 4-byte pointers to real blocks, or 256kB of data.



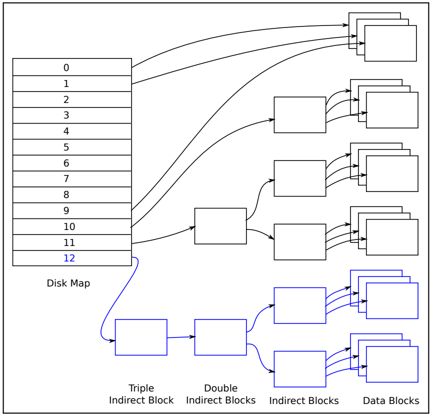
S5FS

The next entry maps to a double indirect block, containing up to 256 pointers to indirect blocks, or 64MB of data.



S5FS

If the file is bigger than this, the next entry maps to a triple indirect block, containing up to 256 pointers to double indirect blocks, or 16GB of data. (The real limit is less than this because the file size needs to be stored in 32-bits.)



S5FS

Sequential and random access are fairly efficient, but might require several lookups per block.

Imagine allocating 2GB of storage for an empty file: all pointers in the disk map are null except the last one, which points to the triple block.

Only four blocks are required.

S5FS

When we want to allocate a new free block, we don't want to start looking through the disk.

The same goes for freeing space – it should require very minimal IO.

S5FS

A list of 100 free blocks is maintained in the superblock.

When the list is empty, the disk is searched for another set of free blocks.

When a block is liberated, it is added to the list of free blocks.

The superblock maintains a list of free inodes in a similar way.