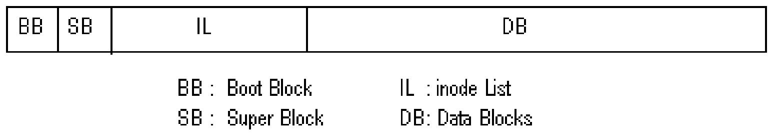
System & Service Management
UNIX File System

Overview



Boot Block
The boot block contains the code to bootstrap the OS. The super block contains information about the entire disk. The I-node list a list of inodes, and the data blocks contains the actual data in the form of directories and files. Now let us dissect each block one by one.
The **super block** contains the following information, to keep track of the entire file system.
Size of the file system
Number of free blocks on the system
A list of free blocks
Index to next free block on the list
Size of the inode list
Number of free the inodes
A list of free inodes
Index to next free inode on the list
Lock fields for free block and free inode lists
Flag to indicate modification of super block

Size of the file system represents the actual no of blocks (used + unused) present in the file system.
The super block contains an array of free disk block numbers, one of which points to the next entry in the list. That entry in turn will be a data block, which contains an array of some other free blocks and a next pointer. When process requests for a block, it searches the free block list returns the available disk block from the array of free blocks in the super block. If the super block contains only one entry which is a pointer to a data block, which contains a list of other free blocks, all the entries from that block will be copied to the super block free list and returns that block to the process. Freeing of a block is reverse process of allocation. If the list of free blocks in super block has enough space for the entry then, this block address will be marked in the list. If the list is full, all the entries from the super block will be copied to the freed block and mark an entry for this block in the super block. Now the list in super block contains only this entry.
Index indexes to the next free disk block in the free disk block list.

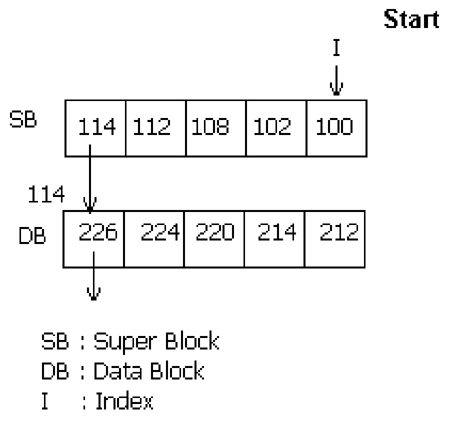


Figure 2: Allocation of Data Blocks

The super block also contains an array to represent free inodes. The size of this array need not be the actual number of free inodes. During assignment of an inode to a new file, the kernel searches the free inode list. If one free inode is found, that one is returned. If the free inode list is empty, then it searches the inode list for free inodes. Each inode will contain a type field, which if 0, means the inode is free. It then fills the free inode list of super block as much as it can, with number of free inodes from inode list. It then returns one of these ones. It then remembers the highest inode number. Next time it has to scan the inode list for free ones, it starts from this remembered one. Hence it doesn't have to scan already scanned ones. This improves the efficiency. While freeing an inode, if the free list in super block has space enough, the freed one is put there. If there is not enough space, and the freed inode number is less than the remembered inode, then the remembered inode is updated with the freed inode. If the freed inode number is greater than the remembered one, then it doesn't have to update, because it will be scanned from the remembered node and the freed one will be covered later.
Index indexes to the next free inode in the free disk block list.

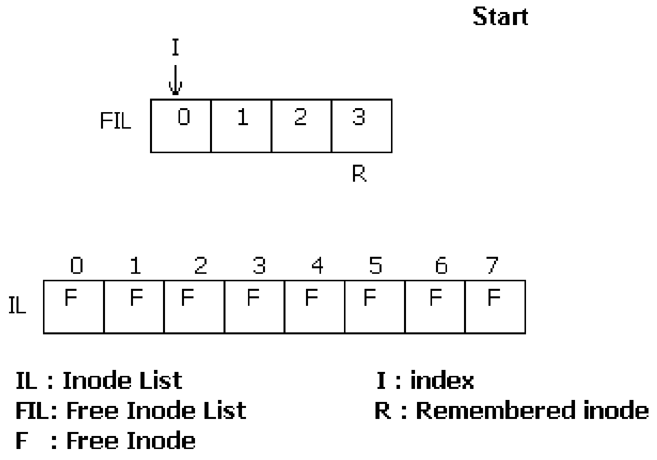


Figure 3: Assignment of new inodes

During all these allocations, in a multi-tasking environment, there is a chance of the inodes getting corrupted. Like if one process was trying to allocate an inode and was preempted by the scheduler, and a second process does the same for same inode, it will be a critical problem. Hence lock flags are introduced. While accessing inodes, that inode will be locked. One more flag to indicate that the super block has been modified, is present in the super block.

The **I-node list** (which server the purpose as FAT+ directory entries in DOS) is a list of inodes, which contains the following entries.
Owner
Type
Last modified time
Last accessed time
Last inode modified time
Access Permissions
No of links to the file
Size of the file
Data blocks owned

Owner indicates who owns the file(s) corresponding to this inode.
Type indicates whether inode represents a file, a directory, a FIFO, a character device or a block device. If the type value is 0 then the inode is free.
The times represent when, the file has been modified, when it was last accessed, or when the inode has been modified last. Whenever the contents of the file is changed, the "inode modified time" also changes. Moreover it changes when there are changes for the inode like permission change, creating a link etc.
Each file will be having nine access permissions for read, write and execute, for the owner, group and others in rwx rwx rwx format.
In Unix we can create links to some files or directories. So we need to have a count of how many links are pointing to the same inode, so that if we delete one of the links the actual data is not gone.
Size of the file represents the actual size of the file.

In Unix, we have a kind of indexing to access the actual data blocks that contains data. We have an array of which (in each inode) first ten elements indicate direct indexing. The next entry is single indirect, then comes double indirect and then triple indirect. By direct indexing we mean that, the value in the array represents the actual data block. If the file needs more than 10 blocks, it uses single indirect indexing, means this is an index to the a block which contains an array of disk block numbers which in turn represent the actual disk block. If all these are exhausted, then double indirect indexing is used and then triple indirect. For further clarifications, refer figure.

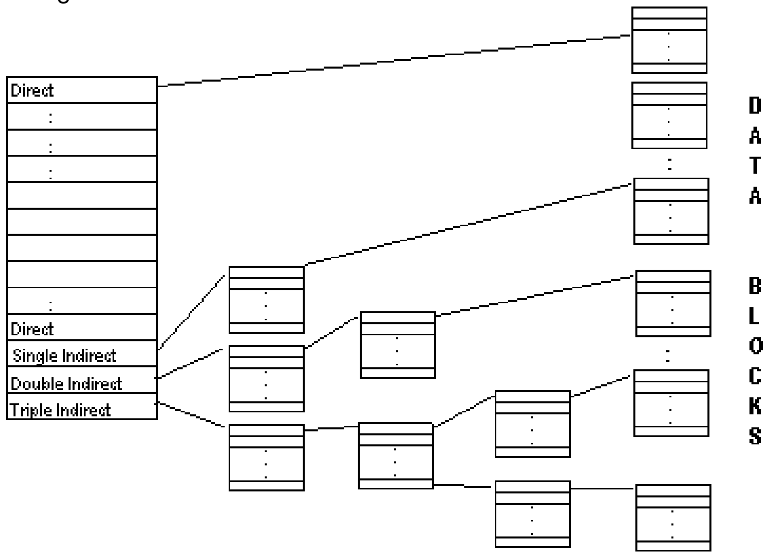


Figure 4: Inode DataBlock representation

The **data blocks** contain the actual data contained in the files or directories. In Unix, a directory is a special file. A directory file contains names of the subdirectories and files present in that directory and its corresponding inode number.

Quelle: http://www.angelfire.com/myband/binusoman/Unix.html

Inode

An inode is a data structure on a traditional Unix-style file system such as UFS. An inode stores basic information about a regular file, directory, or other file system object.

An important part of a file system is the data structures that contain information about the files. Each file is associated with an inode (identified by an inode number, often referred to as an i-number or inode).

Inodes basically store information about files and folders, such as (user and group) ownership, access mode (read, write, execute permissions) and file type. On many types of file systems the number of inodes available is fixed at file system creation, limiting the maximum number of files the file system can hold. A typical space allocation for inodes in a file system is 1% of total size.

The inode number indexes a table of inodes in a known location on the device; from the inode number, the kernel can access the contents of the inode, including the location of the file allowing access to the file.

http://en.wikipedia.org/wiki/Inode

UFS Inodes

The inode information is kept in the cylinder information block. An inode contains all the information about a file except its name, which is kept in a directory. An inode is 128 bytes long. One inode is created for every 2K of storage available in the filesystem. This parameter can be changed when mkfs(1M) is used to create the filesystem.

http://uw713doc.sco.com/en/FS_admin/_ufs_Inodes.html

Chmod

Chmod (Change Mode) is a UNIX command that changes the file system modes of files and directories.

http://en.wikipedia.org/wiki/Chmod

Ls
Ls is a command to list files in Unix and Unix-like operating systems.
http://en.wikipedia.org/wiki/Ls

SUN ZFS
ZFS is a combined file system and logical volume manager designed by Sun Microsystems.
http://en.wikipedia.org/wiki/ZFS