# Lecture 12

## Storage space management

- How does the FS monitor free space on disks ?
- How is disk space allocated to files ?
- What kind of data structures are used by the FS ?
- Is there any mechanism to preserve data consistency across the entire file system ?
- What mechanisms are used to speed up the access to files ?

# File system metadata

- Most file systems have a *header*. The header describes the file system as a whole and gives a starting point for locating files in the system – it includes the total size of the file system, the amount of free space, the date of the last mount, the location of the free space data structures and the starting point for any name lookup.

- Regarding its location, the header can have a fix location (e.g., first block of the disk), or the first block or other known location stores a pointer to a common file that has the FS metadata.

## Free space management

- The allocation of space in a file system is done in fixed-sized blocks (one block is a multiple of a sector, from 512 B to 8192 B).
- For keeping the track of free blocks, the *free bitmap* can be used (0 if the block is allocated).
- Another solution is to use *free lists* (linked lists), embedded in the free blocks.
- Finally, a *simple list* stored in a free block can be used; if it extends over several blocks, pointers will be used.

# Regular files

- The simplest solution for block allocation is to have all blocks allocated to a file contiguously. If the file grows, either there is enough free space after the last block or the entire file needs to be copied in another region of the disk. As both solutions may create difficulties, files are allowed to spread out, but keeping the blocks as close as possible.
- A linked list is a natural structure to represent the blocks allocated to a file. The list itself can be managed as a separate data structure, or each pointer to the next block is stored in the current one. If the list is sorted, it should contain only block numbers. Obviously, if the size of the structure is larger than the block size, the list will be distributed on more blocks that point to each other.
- A variation to the last approach is to use an array of pointers, one for each data block in the file system – each entry gives the next block of the file.
- To speed up the access time, the index blocks can be organised in a tree: with 512-byte blocks, the root will store 128 pointers addressing 128 index blocks that, on their turn, will each address 128 data blocks,…

- What about sparse files ?

## Directories

- The directory maps a subset of names to file metadata – size of the file, file ownership, security characteristics, and the location of the file blocks. All this information can be stored in the directory entry, or the directory entry can point to a data structure.
- The directory is implemented as a file as well, but with a particular structure – name of the file, starting block and size can be the only elements of the entry.

## Directory models

- *Single-level*: all files in the same directory; they must have unique names.
- *Two-level*: a separate directory for each user; there is the master file directory (MFD) and one user file directory (UFD) for every user.
- *Tree-structured*: users can create their own subdirectories; one bit in each directory entry defines the entry as a file (0) or as a directory (1).
- *Acyclic-graph*: a graph with no cycles, allows directories to share subdirectories and files.

## Implementation of sharing

- In UNIX systems: a new directory entry called *link* is created – this is a pointer (absolute or relative path name) to another file or subdirectory. The link is resolved by using the path name to locate the real file.
- Another implementation is to duplicate the information about the shared resource in all directories sharing it. All directory entries about that file/subdir are the same. This solution raises the problem of consistency – a file can have multiple absolute path names !

## Consistency checking

- FS checks to see if free blocks are indeed free. This is done periodically or when a mount operation is executed.
- As consistency checking is time consuming, log files can be used. For example, when a file is deleted, several operations are executed :
  - remove the directory entry;
  - free the data blocks;
  - free the metadata structure.
- If the system crashes during any of these operations, the file system is left in an inconsistent state. As in databases, the delete operation should be either entirely executed or not. The journal contains enough information to control the operation.

## Block caching

- Although applications process individual data from files, the transfer between the disk and the main memory is on blocks. Therefore, the OS reserves memory space for file caches (buffers).
- A kernel thread will take care of data synchronisation: each block in the cache marked modified is queued to be written on the disk; then, the block is marked as clean.
- Synchronisation is also forced when a file system is unmounted.
- As the cache space is limited, the problem of block replacement needs to be solved. The strategy used to replace a block is least recently used (LRU).