*A picture containing text

Description automatically generated*

Reliability: File systems must be reliable and stable at all costs. A file system should be able to detect corrupted files (prevent it, if possible) and should be able to recover from hardware failures. Checksums can be used for checking for corruption while atomic transaction can be used to prevent inconsistencies.

Naming: Directories and files make organizing a file system human readable. Through this feature a user/program can navigate and access/organize their own data easily.

Controlled sharing: File systems should be able to associate a file to an owner and store file read/write/execute privileges. This prohibits unauthorized file access and improves a file system’s overall security.

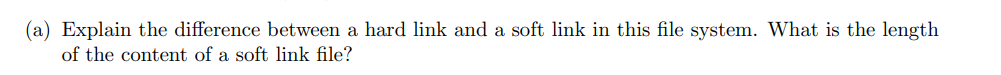
Performance: Since moving a disk arm or erasing a block of solid state memory is expensive in terms of speed, file systems try to make up for this by grouping placement of data so that such operations access large, sequential ranges of storage.



A files metadata includes information about the file such as its owner, security information (read/write/execute privileges of owner/other users), size, modification time etc.

Text

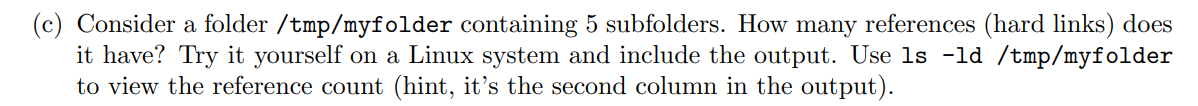
Description automatically generated with medium confidence



The mapping between a name and the underlying file name is called a hard link. A hard link acts as a mirror copy of the original file while a soft link acts as a pointer to the original file. A hard link is useful when duplicating a file is not satisfactory and you want changes on the hard linked files to be synchronized but deleting the original file doesn’t remove the hard linked file. A soft link acts as a shortcut and allows you to link between directories and can cross the file system (across disks, partitions etc.), unlike hard links which work on the same file system. The length of the soft link file is as long as the path the soft link is pointing towards (needs no other information than the path it’s pointing towards).



Two is the minimum number of references (hard links) for any given folder because you need one reference for the pathname that is mapped to an index number and another from current directory “.”



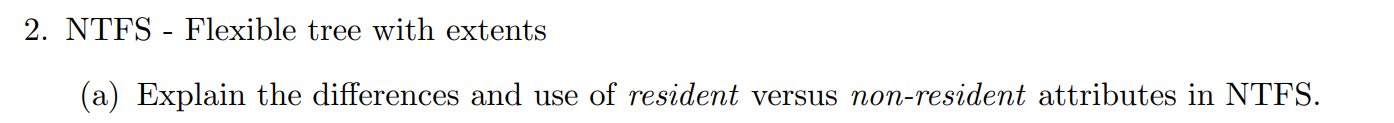
Text

Description automatically generated

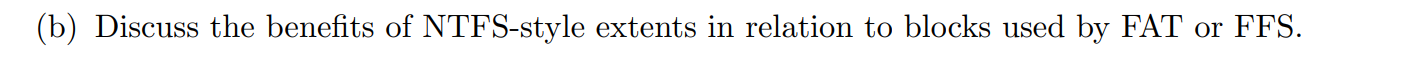
7 references, one for current directory “.”, one for the folder name and 5 for each sub directories (total 7)



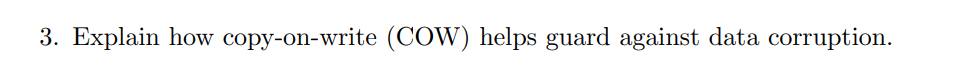
For the sake of efficiency, FFS groups files together based on their location. Files in the same directory are put in the same group and subdirectories are put in different block groups. This is so that the files get stored close to each other on the disk.



Since the NTFS file system views every file and folder as a set of file attributes, it stores these attributes in an MFT file record. Since the MFT also includes the data of a file, it can be problematic to store big files. Resident attributes are thus the file attributes that fit in the MFT file record. The files that were too big get relocated somewhere else in the same volume (non-resident attributes). This is done by creating additional MFST records and puts the “Attributes List” attribute to the first file’s MFT record for referencing the location of all the attribute records (a way of storing where non-resident attributes are located).



Extents are contiguous regions that can vary in size. The depth of the index tree can be minimized because the extents can be large. Blocks used by FAT and FSS have size limits, thereby making the index tree shallower when compared to NTFS index tree.

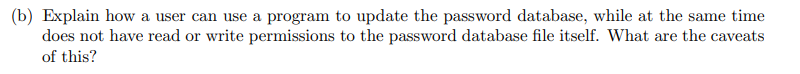


Copy-on-write helps to efficiently implement a “duplicate” or “copy” operation on modifiable resources. When updating an existing file, COW file systems do not overwrite existing data or metadata but instead write new versions to new locations. This update process only updates the i-node when everything has completed successfully to prevent only parts of the file being updated (corruption).

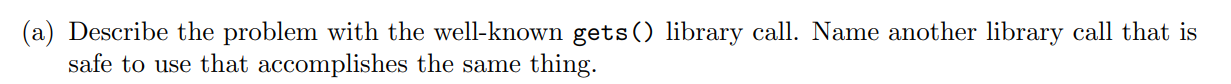
A picture containing chart

Description automatically generated

A password hasher is a one way function, and a hacker could try to reverse it. Salting passwords with unique salts adds an extra layer of protection since passwords are salted uniquely before getting hashed.



By setting uid permission to root (setuid root). The caveat of this is that this gives the program unrestricted access (must not allow the user to have read or write access to the file). Thus the program must be vulnerable free and trusted to not allow such violations.



gets() does not check bounds for memory region and takes input from the user directly into the buffer in memory. Large inputs could write well beyond the input’s intended size and overwrite already existing variables. Worst case scenario this opens the potential of arbitrary code execution.



A microkernel separately user services and kernel services in two different address spaces. Thus, making it less likely that a malicious/vulnerable code can execute in kernel mode.