1 File systems

1. Name two factors that are important in the design of a file system.

Two factors that are important in the design of a file system are performance and reliability and data integrity. Performance relates to the efficiency of accessing data and storage. Reliability and data integrity references the need for data to be stored and retrieved accurately. Additionally it refers to protecting the data from data corruption.

2. Name some examples of file metadata.

Some examples of file metadata could be name, size, author, version and path.

2 Files and directories

1. Consider a Fast File System (FFS) like Linux's ext4.
    a. Explain the difference between a hard link and a soft link in this file system. What is the length of the content of a soft link file?

The difference between a hard link and a soft link in such a file system is that a hard link directly references the same inode as the original file, meaning that a change to the original file would result in a change to all the hard links connected to said file. On the other hand a soft link is more like a reference to a file path, as in that it is its own file containing the path to the given file rather than direct references to the inodes. As such a soft link can exist even if the file it references gets moved or deleted, at which point it becomes a "dangling link". The lengths of the content of a soft link is not a fixed size but rather equal to the path of the file it references.

    b. What is the minimum number of references (hard links) for any given folder?

The minimum number of hard links for a given folder are 2. These hard links are The directory entry in its parent directory, which references the directory name within its parent directory. The second of these hard-links are the special "." link. This is a link within the directory that references the directory itself.

c. Consider a folder /tmp/myfolder containing 5 subfolders. How many references (hard links) does it have? Try it yourself on a Linux system and include the output. Use ls -ld /tmp/myfolder to view the reference count (hint, it's the second column in the output).

The folder ends up with 7 hard links.

```
sverre@Sverre:~/os/2023$ ls -ld tmp/myfolder/
drwxr-xr-x 7 sverre sverre 4096 Nov 10 10:18 tmp/myfolder/
sverre@Sverre:~/os/2023$
```

d. Explain how spatial locality is achieved in a FFS.

Spatial locality in a FFS is achieved through the use of block allocation strategies, chaching, read-ahead, defragmentation and intelligent I/O scheduling. All of these result in data being grouped closer together resulting in faster and more efficient file operations.

2. NTFS - Flexible tree with extents
    a. Explain the differences and use of resident versus non-resident attributes in NTFS.

The difference between and use of resident versus non-resident attributes in NTFS are that resident attributes store its contents directly in the MFT record, whilst on the other hand the non-resident attributes store extent pointers in its MFT record and stores its contents in those extents.

b. Discuss the benefits of NTFS-style extents in relation to blocks used by FAT or FFS.

The benefits of NTFS-style extents in relation to blocks used by FAT or FFS are reduced fragmentation, improved file allocation, minimized wasted space, better performance and support for sparse files. All of these advantages have made NTFS more efficient for modern computing. The older FAT and FFS systems are not as efficient in these aspects, and with the larger files in modern computing their limitations have become more obvious

3. Explain how copy-on-write (COW) helps guard against data corruption.

Copy-on-write (COW) helps to guard against data corruption by not writing to files directly but rather by making a copy of the file and writing to it. It then redirects any traffic from the original file to the new copy until all the files that are still queued up to use the original file are finished,

the system then overwrites the original file with the copied file and diverts traffic back to the original file.

Security

1. Authentication
    a. Why is it important to hash passwords with a unique salt, even if the salt can be publicly known?

It is important to hash passwords with a unique salt, even if the salt can be publicly known because it is an important aspect of enhancing the security of password storage. Additionally it protects against a wide variety of different attacks, such as brute-force attacks and dictionary attacks.

    b. Explain how a user can use a program to update the password database, while at the same time does not have read or write permissions to the password database file itself. What are the caveats of this?

A user can use a program to update the password database, whilst they at the same time don't have read or write permissions to the password database file itself by interacting with a program that has access to modifying the password database. This approach is done to improve security, but if implemented wrong introduces a series of risks such as; elevated security concerns, privilege escalation risk and secure communication issues.

2. Software vulnerabilities
    a. Describe the problem with the well-known gets() library call. Name another library call that is safe to use that accomplishes the same thing.

The problem with the well-known gets() library call is that it does not perform any bounds checking on the data it reads. This can result in overwriting memory outside of its own bound, which in turn can lead to data corruption in addition to exploitable security risks. It is much safer to use the fgets() library method. The fgets() method allows you to specify how many characters to read.

    b. Explain why a microkernel is statistically more secure than a monolithic kernel.

Microkernel is statistically more secure than a monolithic kernel thanks to its focus on design principles that emphasize modularity, isolation and minimalism.