Task 4 – File systems

Ingebrigt Hovind

1. **File systems**
    1. Two factors that are important in the design of a file system is speed and reliability.
    Speed is important because the file system needs to be able to read files quickly in order to increase the responsiveness of the program that will be running it
    Reliability is important because the file system has to be able to store files for long periods of time. The end user does not want to lose data because of crashes or power outages.
    2. File metadata can be: Date/time the file was created, the size of the file, the file format, the creator of the file or when the file was last modified.
2. **Files and directories**
    1.
        a. A hard link is a direct mapping between a name and an underlying file. A soft link is a mapping between one file name to another file name, the file system then uses the hard link of this file to open the file.
        A soft file has no content in and of itself other than the location of the file it is referencing plus some other info about the link itself. So this content would also have no length apart from the path of the file and some other info relating to the soft-link file itself
        b. A minimum number of references of any given folder would be 2, but this is dependent on the file system. On a Unix system two pointers would be created for each new directory. One would point to the folder itself and one would point to the parent folder, the root folder also has two pointers, but both points to the root folder itself, as it lacks a proper parent.
        c. It would have 7 hard links. 1 hard link would be to the myfolder directory itself, 1 hard link would be to the temp directory (the parent of the directory), then there would be 5 hard links to each of the sub-folders.

```
ingebrigt@Stationary:~/Documents/uni-2/Operativsystem
4$ ls -ld temp/myfolder
drwxrwxr-x 7 ingebrigt ingebrigt 4096 nov.  16 19:16
```

        d. FFS uses a tree-based multi-level index for its index structure, it uses a collection of locality heuristics to get good spatial locality.
        These locality heuristics are block group placement and reserve space.
        Block group placement is placing the data where a file's data blocks, a file's data, metadata and different files from the same directory are all accessed together.
        Reserved space is that FFS reserving a fraction of the disk space and treats the disk as full when the rest of the disk has been written to. This allows the file system to avoid having to scatter new writes all across an almost full disk

    2. NTFS
        a. A resident attribute is an attribute which is small enough to fit in a 1 KB MFT-trecord in a NTFS file system.
        A non-resident attribute is too large to fit its contents directly in an MFT record, instead pointers to the extent of the data is stored and the record is stored inside those extents.

b. The block sizes can be of a variable size when using NTFS. This means that large files are more easily stored in a single extent, while smaller files do not have to take up an excessive amount of space, as their extents can be small. This also means that there is no limit to file size with NTFS, but there is with FAT, as the linked list used to link blocks together has a limited size
   NTFS' benefits over FFS are also its flexibility, as it doesn't have to index by fixed blocks, but does this with variable sized extents.

3. Copy-on-write file systems do not overwrite data, instead copying the data to a new location and making the change there. This adds a layer of redundancy protecting against corruption, as if the write corrupts the data in unintended ways, then the file system can revert back to the old version which is still on disk

3. Security

1. Authentication
   a. Because if passwords are hashed without salt then the stored hashes will be equal for equal passwords. This means that if a malevolent actor has access to the hashed passwords and cracks one password, then they can find out which passwords in the database are the same as the one they cracked.
      When passwords are hashed without salt a malevolent actor could also hash common passwords in order to find all the users which used one of these, when they are hashed with salt the guessed password has to be hashed once for every single password in the database in order to achieve the same result.
      The salt can be publicly known because it does not aid in figuring out the original password. It only aids when checking the hash of a guessed password.
   b. Allowing a user to write to a file to which they normally don't have access could be done by a setuid program. Such a program would run with the privileges of the user that created the file, not the privileges of the user that ran it. However, problems arise whenever the person providing authority is different from the person deciding how that authority will be used. Therefore, care must be taken when designing setuid programs. Known bugs such as buffer overflow should be avoided and the program should be as small as possible to avoid the possibility of them being exploited.
2. Software
   a. With gets you have to know exactly how many characters you will be reading so that you can make the buffer large enough. Because C does no bound checking, it may still succeed if it attempts to read to many characters, but it would overwrite parts of the stack.
   b. In a microkernel less processes have privileged access than in a monolithic kernel. This minimizes the amount of possible failure points, making it more secure. Microkernels' source code is often also much smaller smaller than that of a monolithic kernel, making the chance for bugs smaller.