OBL4-OS

1. File systems

1. Two of several factors that are important in the design of a file system is reliability and high performance. When it comes to reliability a user's data should be safely stored even if a machine's power is suddenly turned off or its operating system crashes. Data is so important that users expect and need data to survive even if the devices to store them are damaged. As an example, many modern systems today continue to work even if one of the magnetic disks storing the data malfunctions (Operating Systems, page 491).

   When it comes to high performance, programs that are using data must be able to access it, and it must be fast. As an example, users want programs start-up to be nearly instantaneous (Operating Systems, page 492).

2. Some examples of metadata are file size, file type, file creator, creation time, modification time, access time, permissions, and so on (Operating Systems, page 496).

2. Files and directories

1. Consider a Fast File System (FSS).
   a. A hard link is a file directory entry that map different path names to the same file number, while a soft link or a symbolic link, is a directory entry that map one name to another name (Operating Systems, page 551-552).

      Considered an FSS, a hard link is a pointer to an Inode location for a file. Each hard linked file is assigned the same Inode value as the original, therefore they reference the same physical file location. One downside with hard links is that they are unable to cross different file systems (GeeksForGeeks, 2022).

      A soft link is similar to the file shortcut feature which is used in Windows Operating Systems. Each soft linked file contains a separate Inode value that points to the original file. Like hard links, any changes to the data in either file is reflected in the other. The difference is that soft links can be linked across different file systems, but if the original file is deleted or moved, the soft linked file will not work anymore. (GeeksForGeeks, 2022).

      The size of the content of a soft link is equal to the length of the path of the original file. (GeeksForGeeks, 2022).

b. The minimum number of references (hard links) for any given folder is 2. This is for a reference to the parent directory(..) and itself(.) (Unix.StackChange, 2013).

c.

```
jenscaa@jenscaa:~/os/2023/o4/tmp$ ls -ld myfolder
drwxr-xr-x 7 root root 4096 Nov  7 18:26 myfolder
```

Figure 1: These lines of codes show the command ls -ld myfolder which contains 5 subfolders.

The folder /tmp/myfolder has 7 hard links. The figure above demonstrates this considered I added 5 subfolders to it.

d. In an FFS spatial locality is achieved through several mechanisms. Particularly FFS's index structure, called multilevel index, which is a structured tree that allows FFS to locate any block of a file which it efficient for both large and small files (Operating Systems, page 558). To efficiently support both large and small files with a fixed tree structure, FFS's tree structure is asymmetric. This allows FFS to store successive groups of blocks at increasing depth dependent of the file's sizes. Small files are stored in a small-depth tree, the bulk of medium files are stored in a medium -depth tree, and the bulk of large files are stored in a larger-depth tree (Operating Systems, page 562).

One other mechanism is FFS's free space management. In simplicity it creates a bitmap with one bit per storage block. The i'th bit in the bitmap indicates whether the i'th block is free or in use. The position of FFS's bitmap is fixed when the file system is formatted, so it is easy to find the bitmap that identifies free blocks near any location of interest.

One other mechanism is the locality heuristic Block Group Placement. To achieve spatial locality, FFS places data to optimize for a common case where a file's data blocks, a file's data and metadata, and different files from the same directory are accessed together. Since everything cannot be near each other, FFS lets different directories be far from each. (Operating Systems, page 565). Essentially the FFS divides a disk into block groups so that the seek time between any blocks will be small. Then it puts a directory with its files in the same block group. When a new file is created, FFS knows the inode number of the new file's directory, and can from that determine the range of inode numbers in the same block group. FFS chooses an inode from that group if one is free. Othewise it gives up locality and selects an inode number from a different block group.

Within a block group FFS uses a first-free heuristic. This means that when a new block of a file is written, FFS writes the block to the first free block in the file's block group. This has the side effect by giving

up some locality, since it might write into small holes near the start of a block group rather than writing a sequence of contiguous free blocks somewhere else. This is acceptable for small files, and for big files, it will usually tend to have its first few blocks scattered through holes before the blocks are contiguous, and thus provides some spatial locality (Operating Systems, page 566).

2. NTFS (*The Microsoft* New Technology File System) – Flexible tree with extents.
    a. The differences between resident and non-resident attributes in NTFS is that a resident attribute stores its contents directly in the MFT record (*Master File Table*), while a non-resident attribute stores its pointers in its MFT record and sores its contents in those extends (Operating Systems, page 568).

    b. Compared to blocks which are fixed sized regions, extents are a variable-sized region of files that are each stored in a contiguous region on the storage device (Operating Systems, page 568).

    The benefits of NTFS-style extents in relation to blocks used by FAT or FSS, is that the NTFS has an important feature for optimizing its best fit placement, which allows an application to specify the expected size of a file at creation time. Compared to NTFS, FSS and FAT allocates file blocks as they are written, without knowing how large the file will eventually grow. In this way NTFS reduces fragmentation compared to Fat and FFS and improves disk access performance (Operating Systems 571-572).

3. Copy-on-write (COW) can help guard against data corruption by not overriding existing data or metadata, but instead write new versions to new location. This is done in order to optimize writes by transforming random I/O updates into sequential ones (Operating Systems, page 573). An example of this is when appending a block to a file. A traditional update-in-place file system might seek to update its free space bitmap, the file's inode, the file's indirect block, and the file's data block. On a COW file system, it might find a sequential run of free space and write all to that free location (Operating Systems, page 573). In this way a file system is less prone to data corruption, but the trade-is the overhead in creating copies.

3. Security

1. Authentication.
    a. It is important to hash passwords with a unique SALT because of the work that an attacker does to crack one user's salted password cannot be reused against another user (Security.Stackexchange, 2017). Even if the SALT is publicly known it will still be a challenge. Consider an

attacker that uses a rainbow table with precomputed password-hashes for commonly used passwords. The attacker would have to add a second layer in this table or a separate rainbow table because a unique SALT produces different hashes with the same passwords, and thus making it harder and time consuming for an attacker to crack.

In summary SALTs can be public because they offer a big security improvement even if the attacker knows them. The best measurement for security is to keep the SALT a secret, which is then called *pepper* (Security.Stackexchange, 2017).

b. A user can use a setuid program (a program that that runs with escalated permissions) to update the password database. In this way the user can make updates even though if the user does not have read or write permissions to the password database file itself (Computerhope, 2023).

The caveats of this are that any user can use this program. It is therefore most important to implement secure user authentication within the setuid program. To avoid security risks the setuid program must written carefully and only be limited to perform specific operations within the password database. It is important that the program does not have broad or unrestricted access to the system.

2. Software vulnerabilities.
   a. The problem with the gets() library call is that it does not perform any array bound testing on the input. gets() keeps on reading until it sees a newline character, making it susceptible to buffer overflow (GeeksForGeeks, 2023).

   Another library call that is safe to use that accomplishes the same thing is the fgets() call. To avoid buffer overflow, fgets() should be used instead of gets() as fgets() limits the numbers of characters read (GeeksForGeeks, 2023).

   b. The reason why a microkernel is statistically more secure than a monolithic kernel, is that if any service fails in a monolithic kernel-based operating system, it will lead to an entire system failure. In a monolithic kernel all services run in kernel space, meaning that they all share the same memory space. This is a vulnerability that can be compromised. In a microkernel architecture, many system services that run in a monolithic kernel, are run in user space. This separation improves protection and prevents possible malicious code to enter the kernel. (GeeksForGeeks, 2023).

Sources:

- Anderson, T. & Dahlin, M. (2014, August 21.). *Operating Systems: Principles & Practice* (2. Edition). Recursive Books.
- GeeksForGeeks. (2022, June 21.). *Soft and Hard links in Unix/Linux*. GeeksForGeeks. Retrieved from: https://www.geeksforgeeks.org/soft-hard-links-unixlinux/
- (User) Gilles 'SO- stop being evil'. (2013, November 17.). *Why does a new directory have a hard link count of 2 before anything is added to it?* Unix.Stackexchange. Retrieved from: https://unix.stackexchange.com/questions/101515/why-does-a-new-directory-have-a-hard-link-count-of-2-before-anything-is-added-to
- Bodewes, M. (2016, August 3.). *Why can salts be public?* Security.Stackexchange. Retrieved from: https://security.stackexchange.com/questions/131731/why-can-salts-be-public
- Computer Hope. (N/A). *Setuid*. Computer Hope. Retrieved from: https://www.computerhope.com/jargon/s/setuid.htm#setuid
- GeeksForGeeks. (2023, October 30.). *gets() is risky to use!* GeeksForGeeks. Retrieved from: https://www.geeksforgeeks.org/gets-is-risky-to-use/
- (User) akash1295. (2023, May 5.). *Monolithic Kernel and key differences from Microkernel*. GeeksForGeeks. Retrieved from: https://www.geeksforgeeks.org/monolithic-kernel-and-key-differences-from-microkernel/