IDATT2202 OBL4-OS

Nicolai H. Brand

November2022

1 File systems

1.1

- A file system should be convenient to use for the end-user. This is done through a concept called "named data" which means that the internal representation of a file is not what the user sees. Rather, the user accesses data through human-readable names.

- Another important consideration is performance. This can be done by grouping files in a directory close to each other. It is likely that once a user opens one image in the images directory they may want to open another one. Having all images in the images directory scattered across the physical disk could be less efficient than having them close together.

1.2

The name of the file. Who has access to read/write. Most file systems also store the date and time of creation and last edit. It could also beneficial to store the size of the file so it does not need to be calculated on every request for the file size.

2 Files and directories

2.1

a) A soft link, usually called a symbolic link or just symlink, is a mapping from one file to another. Conceptually it is similar to a pointer. The length of the content of a symlink will be the length of whatever the symlink is pointing to. The length of the actual symlink itself will be the length of the path it points to.

A hard link is a mapping/link/reference to actual physical data.

b) Two. The directory itself will require one hard link mapping for its path. In addition, the ext4 file system stores a special hard link in every directory to itself. A directory does not need to contain any entries, and so the empty

directory will have two hard link. One for the path to the directory, and one for itself itself.

c) After creating the sub-directories and running $ls-ld$ inside the root folder I get seven hard links. This makes sense. Five hard links to the sub-directories (irrespective of how many hard links any sub-directory itself has) and the two hard links described above.

d) FFS uses, among other things, block groups as a locality heuristics. Block groups means that files that are grouped together in the user interface (for example the files inside a directory) are grouped together in the actual physical data.

2.2

a) A resident attribute means that the attribute in question is the actual extent that the file is stored in. A non-resident attribute means that the extent is not stored in the attribute, and that the attribute itself is a pointer to the actual extent.

b) A block in FAT and FFS have a fixed size, usually 4096 bytes or 4kB. If a file is less than that the file system would still allocate 4096 bytes for that file. This results in internal fragmentation where some files take up way more space than they need. An extent will only take up the space it needs.

2.3

If a file is modified, it does not directly change the underlying data of the file being modified. Instead, the file systems creates a copy of the modified file, hence the name copy-on-write. This means that the file system can guard itself against data corruptions if something where to go bad during the new write.

3 Security

3.1

a) Adding a unique salt to the password being hashed means that the same passwords creates different unique hashes. This means that a malicious actor can not use a rainbow table attack. A rainbow table can contain computed hashes of common passwords for common hashing algorithms. A user with a weak or common password would therefore be vulnerable to a rainbow table attack. When the password is salted, despite the salt being known in advance, a malicious actor would have to re-compute all the hashes with the unique salt for every password it wants to crack. This proves to be very time consuming.

b) A user could open and modify database if it has access switch into the root user. A similar approach would be to give the user access to modify the database itself as if it was the root user with setuid using the chmod utility. This comes with security risks as the user gets temporary and limited root privileges and a bad behaving programs could take advantage of this.

3.2

a) The gets() function takes in one parameter; a buffer. The size of the buffer is decided by the user, however, the function does not make sure the data read fits into that buffer. Therefore, you may get a buffer overflow which is a well-known security vulnerability that can be maliciously exploited. The fgets() function takes in the buffer to store the data, but also the size of the buffer. fgets() will therefore never attempt to store more data in a buffer than there is space for (assuming the caller provides the correct buffer size value).

b) The philosophy of a microkernel is to move as much behaviour as possible outside the kernel. The course book claims that something around 90% of crashes in the kernel are caused by device drivers. In fact, most of the Linux kernel are device drivers. The total lines of code and the amount of security vulnerabilities a program has tend to be correlated. Therefore, less code tends to result in less room for security vulnerabilities to creep in.