1. When a user wants to access an old file (X) through an existing link, they will end up accessing the new file (Y) instead. This occurs because the access protection for file X is used, rather than the one associated with file Y. To prevent this issue, it is essential to remove all links to a deleted file.

To achieve this, there are a few options available. One approach is to keep a comprehensive list of all links that point to a file. As the file is removed, each of these links is removed from the list. Another strategy is to retain the links but remove them as soon as an attempt is made to access the deleted file.

A third method involves maintaining a file reference list, or counter, which tracks the number of links or references to a file. Only after all links or references to the file have been deleted from the list, will the file be deleted. By implementing one of these methods, you can avoid the problem of users accessing the wrong file through outdated links.

2. To enable certain operations, such as ensuring that a file is not deleted until all processes accessing it have closed, the operating system can use a central open-file table. This feature would be impossible to implement without such a table.

For instance, if a file is being accessed by one or more processes and then gets deleted, the operating system must check if any process is still accessing it before removing the file from the disk. This check can only be made if there is a centralized accounting system that keeps track of the number of processes accessing the file.

Additionally, when two processes are accessing the same file, a separate state must be maintained to keep track of which parts of the file are being accessed by each process. The operating system must create separate entries in the central open-file table for each of these processes. This ensures that each process can access the file independently without interfering with the other.

Therefore, by using a central open-file table, the operating system can perform critical checks and ensure that processes can access files without interference, thereby improving system efficiency and security.

3. When multiple users are updating the same file simultaneously, having a single copy of the file may lead to incorrect information being presented to the users and the file being left in an inconsistent state.

On the other hand, creating multiple copies of the file to address this issue may result in storage waste and inconsistency between the various copies. The copies may not be consistent with each other, leading to conflicting and inaccurate data.

Therefore, maintaining multiple copies of a file can have a significant impact on the consistency and accuracy of the data, while having a single copy can result in incorrect information and an inconsistent file state. To address this issue, file locking mechanisms and version control systems can be implemented to ensure that multiple users can access and modify a file without conflicting with each other, while maintaining a single copy of the file. This approach can help avoid inconsistencies and inaccuracies while minimizing storage waste.

4. Disk allocation, address mapping, and access concept can be used to distinguish between the three primary strategies for file storage: contiguous, linked, and indexed techniques.

The contiguous technique stores files in sequential order and keeps them close together on the disk. As a result, sequential traversal is used when reading the file. This technique is useful for situations where the entire file is needed to be accessed, but it can be less efficient for accessing specific parts of a file. When files are accessed at random, the contiguous approach can be helpful in identifying a specific block by utilizing an adjacent disk search.

The linked technique utilizes links that connect one block to another, making it easy to access the file. However, random access can be less efficient with this technique because of the linkages between blocks. When searching for a specific block randomly in this scenario, working with links to reach the seek point is still a necessary step.

The indexed technique is useful because it employs an index, which makes it simple to perform sequential access. When selecting blocks randomly, the indexed technique is particularly effective because it enables the selection of blocks at random. Therefore, the indexed technique can be used for both sequential and random access.

In summary, each of these techniques has its advantages and disadvantages. The contiguous technique is efficient for sequential access, the linked technique is useful for accessing files through links, and the indexed technique is effective for both sequential and random access. By considering the type of access required for a given file, the most appropriate technique can be chosen to ensure efficient and effective access.

5. For the contiguous allocation strategy, the logical block number is divided by the physical block size, which is 512 bytes in this case. The quotient is stored in X, and the remainder in Y. To get the physical block number, X is added to the starting file address Z. The displacement within the block is given by Y. To access logical block 4, only one physical block needs to be read from the disk.

For the linked allocation strategy, the logical block number is divided by 511, which is the maximum number of pointers that can be stored in a single block. X and Y are again the quotient and remainder respectively. To access a particular logical block, we need to follow the linked list from the beginning until we reach the block containing the desired data. Since we start at logical block 10 and want to access block 4, we need to chase down the linked list for six blocks, and then add one to Y to get the displacement within the last block. Thus, we need to read four physical blocks from the disk.

For the indexed allocation strategy, the logical block number is again divided by 512, with X and Y representing the quotient and remainder respectively. In this strategy, an index block is used to map logical block numbers to physical block numbers. To access a specific logical block, we first need to read the index block into memory. Then, we can find the physical block number associated with the logical block number by looking up the corresponding entry in the index block at position X. The displacement within the block is given by Y. To access logical block 4, we need to read two physical blocks from the disk.