HW 6

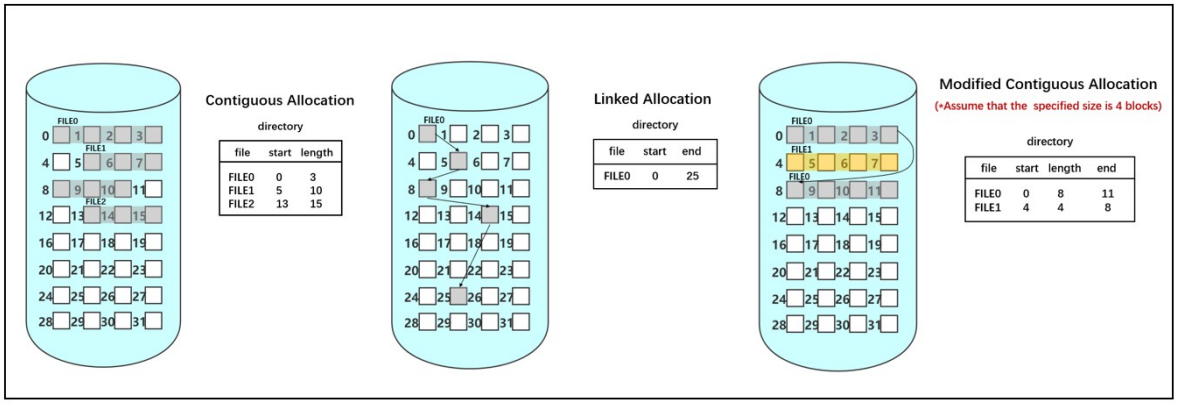
Chapter 14

14.4

Question: One problem with contiguous allocation is that the user must preallocate enough space for each file. If the file grows to be larger than the space allocated for it, special actions must be taken. One solution to this problem is to define a file structure consisting of an initial contiguous area of a specified size. If this area is filled, the operating system automatically defines an overflow area that is linked to the initial contiguous area. If the overflow area is filled, another overflow area is allocated. Compare this implementation of a file with the standard contiguous and linked implementations.

Answer:

Firstly, we may need use illustration to make this question more clearly.



Comparing to standard contiguous implementation:

According to the illustration, we can obviously find that with this implementation of a file. the external fragmentation problem of normal contiguous allocation is solved, and as we can allocate overflow area, another contiguous allocation problem which is about the space extension is solved as well. But as these shortages are gone, the internal fragmentation problem appears. Because we allocate an initial contiguous area of a specified size, for the files can not hold all these space, internal fragmentation will come. And if we fix this specified size too large, many space will waste, while the size is too small, many link will be established which will affect the performance of the file system.

Comparing to standard linked implementation:

Comparing to the linked allocation, the straight change is that we can reduce the link number so that we can make the performance better in certain situation that we set the specified size bigger than one block. Linked allocation requires that system need to jump with link information every time that we need to read the next block, but in this implementation, only when the block is not in the specified size, we need to do this operation.

And if we set the specified size as 1, this implementation transfer to linked allocation.

14.14

Question: Consider a file system on a disk that has both logical and physical block sizes of 512 bytes. Assume that the information about each file is already in memory. For each of the three allocation strategies (contiguous, linked, and indexed), answer these questions:

a. How is the logical-to-physical address mapping accomplished in this system? (For the indexed allocation, assume that a file is always less than 512 blocks long.)

b. If we are currently at logical block 10 (the last block accessed was block 10) and want to access logical block 4, how many physical blocks must be read from the disk?

Answer: Let **Z**(block number) be the starting file address. And we assum that the logical address we want to access is **logicAd**, the physical address is **physicalAd**.

a. **Contiguous Allocation:**

If this system uses contiguous allocation strategy, we can know that block number of physical address is $(Z + \lfloor logicAd/512bytes \rfloor)$ and the offset in the block is $(logicAd \ \% \ 512bytes)$. The mapping process is shown below.

$$\because considering\ that\ the\ starting\ block\ number\ is\ Z$$
$$\because physicalAd\ cover\ \lfloor logicAd/512bytes \rfloor\ blocks$$
$$\therefore physicalAd's\ block\ number = Z + \lfloor logicAd/512bytes \rfloor$$
$$offset\ in\ the\ block = logicAd\ \%\ 512bytes$$

Linked Allocation: Assume **space of pointer** in every block is **1 byte**

If this system uses linked allocation strategy, we need to read $(\lfloor logicAd/(512bytes - 1\ byte) \rfloor + 1)$ blocks from starting block(including starting block) to find the physical block and the offset in the block is $(logicAd \ \% \ (512bytes - 1\ byte) + 1\ byte)$.

$$\therefore block\ jump\ number = \lfloor logicAd/(512bytes - 1\ byte) \rfloor (attention: it\ is\ not\ physicalAd's\ block\ number)$$
$$offset\ in\ the\ block = logicAd\ \%\ (512bytes - 1\ byte) + 1\ byte$$

Indexed Allocation:

If this system uses indexed allocation strategy, it need to access the indexed blocks, find the physical block number corresponding to the logical block number, and the offset in the block is $(logicAd \ \% \ 512bytes)$.

$$offset\ in\ the\ block = logicAd\ \%\ 512bytes$$

b. **Contiguous Allocation:** 1

If this system uses contiguous allocation strategy, it can accomplish random accessing as it does not need to access other blocks, which means that it can directly access logical block 4. So, there are only one block that is read from disk.

Linked Allocation: 4

If this system uses linked allocation strategy, it can not access blocks in random. The walking process is unavoidable, so it will walk to block 4. There are 4 blocks must be read from the disk.

Indexed Allocation: 2

If this system uses indexed allocation strategy, it need access index block before accessing block 4. So there are 2 blocks must be read from the disk.