Filesystem Activity

So for this activity we want you to discuss the various pros and cons of a couple different file allocation approaches. We're going to simplify things to try to highlight the most key issues.

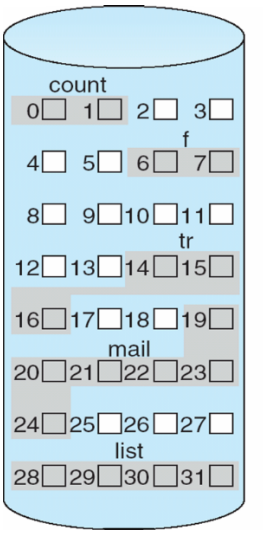
In this system, we're going to imagine our disk as a collection of fixed sized blocks. We'll do all our allocation at the block level i.e. you can't devote part of a block to one file and one to another. Each block will have a block id which is a number that uniquely identifies the block. File sizes will also be in terms of number of blocks.

We're also going to only worry about reading or writing file data – we're not going to worry about finding a file in the directory structure. We're just going to assume that somehow the directory lookup has already occurred and we've found some magical file record. Exactly what data is in that record will depend on our approach but it's important to note that these records are of fixed size. Files themselves will be of variable size but if you want variable-sized metadata you have to store it in the file's blocks.

For each of these approaches we describe the approach and then list a couple desirable characteristic of a filesystem. You team's task will be to understand the approach and then decide if this particular system is "good" or "bad" as far as this characteristic goes.

Approach: Contiguous allocation

In this approach each file has its data stored contiguously on the disk. So if a file is 3 blocks and starts at block 4, the blocks used are 4-6. The file record stores the size of the file and the file's starting block – from that you can instantly figure out where any particular file block must be on the disk.

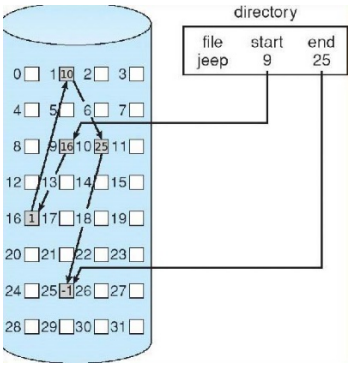


Characteristics:

Characteristic	Good or bad?	Why?
Fast to navigate to any particular position in the file (e.g. if I want to lookup data in the 100th block, it should be easy to do so)		
Needs little metadata (i.e. amount of space devoted to non-file data is small)		
Can store very large files (i.e. files on the order of the size of the disk)		
Can append data to existing files easily		

Approach: Linked Allocation

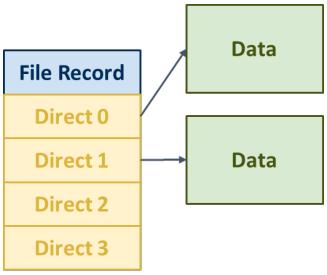
In this approach each block ends with a block id of the next block id in the sequence. The file record stores the block id of the starting block and the block id of the ending block (for fast appending).



Characteristic	Good or bad?	Why?
Fast to navigate to any particular position in the file (e.g. if I want to lookup data in the 100th block, it should be easy to do so)		
Needs little metadata (i.e. amount of space devoted to non-file data is small)		
Can store very large files (i.e. files on the order of the size of the disk)		
Can append data to existing files easily		

Approach: Indexed Allocation

The file record contains a (fixed size) list of block ids. The first entry represents the first block of file data, 2nd entry represents second, etc. For smaller files, some entries may be -1.



Characteristic	Good or bad?	Why?
Fast to navigate to any particular position in the file (e.g. if I want to lookup data in the 100th block, it should be easy to do so)		
Needs little metadata (i.e. amount of space devoted to non-file data is small)		
Can store very large files (i.e. files on the order of the size of the disk)		
Can append data to existing files easily		

We can get some of the good characteristics of both worlds if we combine aspects of these approaches. What do you think a good system would be?