Jeremy Holloway
CPSC-3220-001
6/17/2019

Pre-class Assignment #20

1. Define the following terms:
-defragment: coalesce scattered disk blocks to improve spatial locality, by reading data from its present storage location and rewriting it to a new, more compact, location.
-file allocation table: An array of entries in the FAT file system stored in a reserved area of the volume, where each entry corresponds to one file data block, and points to the next block in the file.
-multi-level index: A tree data structure to keep track of the disk location of each data block in a file.
-inode: In the Unix Fast File System (FFS) and related file systems, an inode stores a file's metadata, including an array of pointers that can be used to find all of the file's blocks. The term inode is sometimes used more generally to refer to any file system's per file metadata data structure.
-indirect block: A storage block containing pointers to file data blocks.
double indirect block: A pitfall in concurrent code where a data structure is lazily initialized by first, checking without a lock if it has been set, and if not, acquiring a lock and checking again, before calling the initialization function. With instruction reordering, double checked locking can fail unexpectedly.
triple indirect block: A storage block containing pointers to double indirect blocks.
extent: A variable-sized region of a file that is stored in a contiguous region on the storage device.
-master file table: In NTFS, an array of records storing metadata about each file.
-attribute record: In NFTS, a variable size data structure containing either file data or file metadata.
-partition: separate, usually smaller, virtual storage devices that can be formatted as a separate file system.

2. Why do you need a specialized API to write directory files?

They require a specialized API because they must control the contents of the directory files. File systems must prevent prevent application from corrupting these crucial files.

3. Why is a linear list not an adequate data structure for implementing a directory?

Simple lists are fine if the data set is small but when there are thousands of files a simple list structure becomes too sluggish.

4. Should you store file metadata in a directory entry or in the file itself? Explain your rationale in choosing between the two locations.

If you stored file metadata in a directory entry, whenever metadata is updated you would have to located and update every file entry. Metadata should be stored in the file's data itself to make it easier to modify.

5. Why is a linear list not an adequate data structure for implementing a file?
A file needs an index to provide a way to locate each block of the file.

6. How does the file allocation table support both the linking of file blocks and the tracking free space?
Each file is a linked list of blocks, supporting the linking of file blocks. The file system can find a free block by scanning through FAT allocation table to find a zeroed entry.

7. Why does the file allocation table have difficulty supporting random accesses?
Requires sequentially traversing the file's FAT entries until the desired block is reached.

8. Give at least one reason why the tree structure in an FFS inode is asymmetric.
To efficiently support both large and small files.

9. Under what conditions can a file's data be "resident" in NTFS?
When an attribute is small enough to fit in an NFT record.

10. What is a copy-on-write file system?
File systems that write new versions to new locations when updating an existing file to optimizes writes by transforming random I/o updates to sequential ones.