CS 537 Notes, Section #27: File System Crash Recovery

Unix File System Crash Recovery

Computers can crash at any time, and we want the file system to behave sensibly in the face of crashes. The key idea is called *consistency*:

- The file data and the various control structures (descriptors, bitmaps) must be in agreement.
- Since crashes can occur at any time, not all updates to the disk may be completed.
- We must insure that when the system reboots, it can return its file system to some sensible state.
- The key constraint is that any file system write operation, in progress at the time of the crash, either completely finishes or appears as if it never happened. This is called *atomicity* by the database folks.

Insuring consistency requires two things:

- Updates to the file system data structures must be done in the write order (and there is *only* one right order)!
- The proper steps must be taken at reboot time to bring the system back in to a consistent state.

There are three basic updates that happen when data is written to a file.

1. A block (or blocks) is allocated from the free list (bit map).
2. Data is written to the newly allocated block.
3. The inode is updated to include the new data.

These operations must be done in the above order. If they are not, then it is possible to have a data block included in a file that might have garbage (uninitialized data) in the block.

After rebooting, the recovery utility program on Unix, called "fsck", is going to traverse the entire directory structure of the disk to insure that all free blocks are in the free list.

Recovery after a crash follows these steps:

1. Allocate a temporary bit map, initialized to indicate that all disk blocks are free.
2. Start at the inode for the root directory.
3. Traverse the directory:
    1. For each disk data block in the directory file, marks its blocks as "allocated" in the bit map.
    2. For each data file in this directory, marks its data blocks as "allocated" in the bit map.

3. For each directory in this directory, perform the "Traverse the directory" steps above.

At the completion of the algorithm, you can compare the actual bit map to the temporary one to to find blocks that were allocated, but never made it into a file.

Windows File Sysem Crash Recovery

NTFS assures that the file system will remain consistent by use of a *write log*. This technique is similar to that used in a database system.

As in other file systems, consistency means that a write (or group of writes) to a file either complete or do not happen at all. It is not possible for a data block to be in an undefined state (e.g., allocated, but not written).

- The log is one of those standard files stored at the beginning of the MFT. It is called, cleverly enough, the *log file*.
- A simplified version of the steps to write data to a file look like:
  1. A file update is written to the *in-memory* log buffer.
  2. Updates to the *in-memory* file data and associated file system structures are made.
  3. The log changes are flushed to disk.
  4. The file data and structure changes are flushed to disk.
- If the system crashes during a file update, it is sufficient to go through the log an re-do each operation specified in the log.
- The system occassionally creates checkpoints, so that it does not have to back to the beginning of the log for recovery. Checkpoints have two main benefits:
  - Log files can be truncated, reducing the space needed for the log.
  - Recovery time is faster if fewer log records need to be processed.
