Inline Data

The inline data feature was designed to handle the case that a file's data is so tiny that it readily fits inside the inode, which (theoretically) reduces disk block consumption and reduces seeks. If the file is smaller than 60 bytes, then the data are stored inline in `inode.i_block`. If the rest of the file would fit inside the extended attribute space, then it might be found as an extended attribute "system.data" within the inode body ("ibody EA"). This of course constrains the amount of extended attributes one can attach to an inode. If the data size increases beyond i_block + ibody EA, a regular block is allocated and the contents moved to that block.

Pending a change to compact the extended attribute key used to store inline data, one ought to be able to store 160 bytes of data in a 256-byte inode (as of June 2015, when i_extra_isize is 28). Prior to that, the limit was 156 bytes due to inefficient use of inode space.

The inline data feature requires the presence of an extended attribute for "system.data", even if the attribute value is zero length.

Inline Directories

The first four bytes of i_block are the inode number of the parent directory. Following that is a 56-byte space for an array of directory entries; see `struct ext4_dir_entry`. If there is a "system.data" attribute in the inode body, the EA value is an array of `struct ext4_dir_entry` as well. Note that for inline directories, the i_block and EA space are treated as separate dirent blocks; directory entries cannot span the two.

Inline directory entries are not checksummed, as the inode checksum should protect all inline data contents.