

EXT4: Bit by Bit

Hal Pomeranz
Deer Run Associates



What's New in EXT4?

CEIC[®] 2011

- 48-bit address space
- Uses extents instead of indirect block chains
- 64-bit nanosecond resolution timestamps
- File creation time timestamp



- Backwards compatibility was a design goal
- Inodes expanded to 256 bytes:
 - Much of the first 128 bytes unchanged from EXT[23]...
 - ... except that block pointers replaced by extents
 - Extended timestamps, etc in upper 128 bytes



Let's Make a File!

CEIC[®] 2011

```
# echo Time for knowledge >testfile  
# touch -a -t 211101231917.42 testfile  
# touch -m -t 204005160308.19 testfile
```

No fractional seconds!

	stat	istat	debugfs
Access	2111-01-23 19:17:42.0	1974-12-17 12:49:26	1974-12-17 12:49:26.0
Modify	2040-05-16 03:08:19.0	2040-05-16 03:08:19	2040-05-16 03:08:19.0
Change	2011-03-12 07:36:13...	2011-03-12 07:36:13	2011-03-12 07:36:13...
Create	N/A	N/A	2011-03-12 07:36:04...



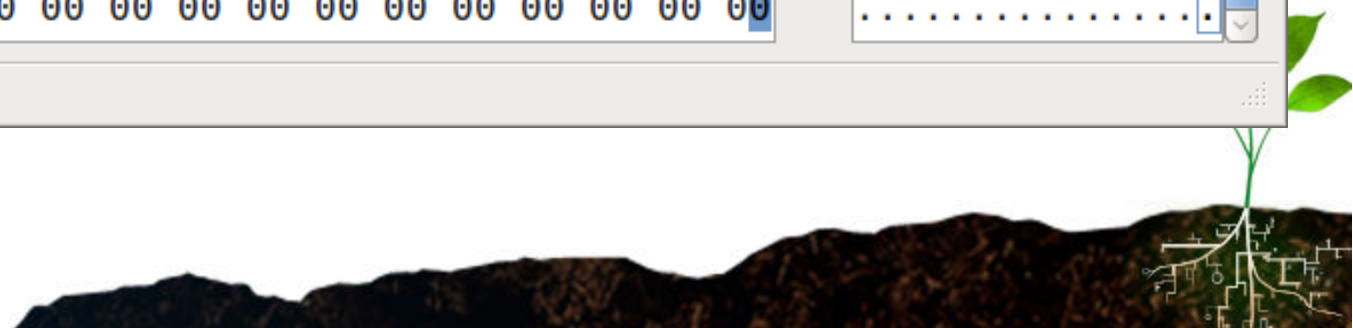
Timestamps In The Inode

CEIC[®] 2011

The screenshot shows a GHex window titled "testfile.inode - GHex" with a menu bar (File, Edit, View, Windows, Help). The main area displays a hex dump of inode data. Annotations with arrows point to specific fields:

- Mtime** (Modification Time): Yellow box highlighting bytes 13 59 5E 84 at offset 00000010.
- Atime** (Access Time): Green box highlighting bytes 56 B1 54 09 at offset 00000008.
- Ctime** (Change Time): Pink box highlighting bytes ED 92 7B 4D at offset 0000000C.
- Creation Time (Btime)**: Blue box highlighting bytes E4 92 7B 4D at offset 00000090.
- Seconds**: Two boxes, one pointing to the Mtime field and another pointing to the Btime field.
- "Extra"**: Two boxes, one pointing to bytes BD 9F CF at offset 00000060 and another pointing to bytes 6C F0 8A 14 at offset 00000094.

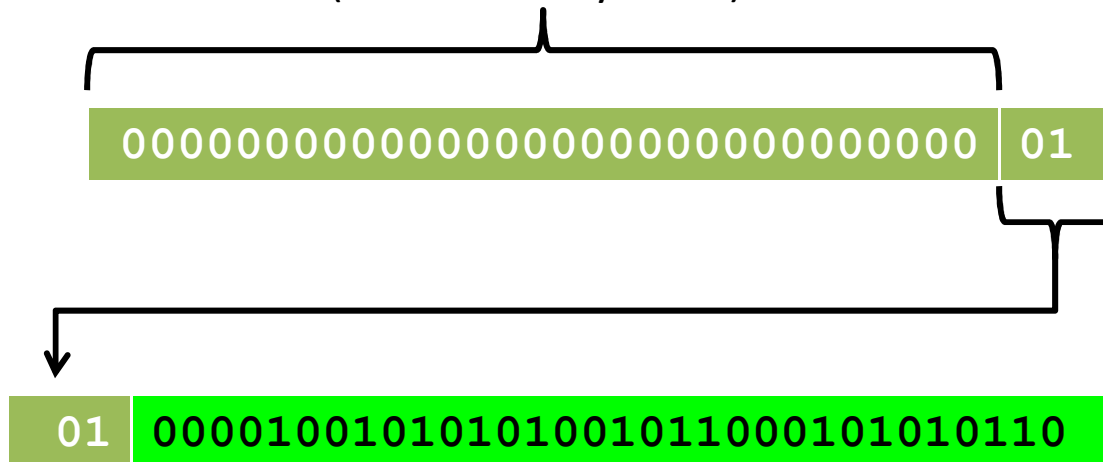
Other visible data includes bytes 1C 00 00 00 at offset 00000080 and 18 BE FF CF at offset 00000084. The right pane shows ASCII characters: ".....V.T...{M", ".Y^.....", ".....6...", and "..{ML.....". The status bar at the bottom left shows "Offset: FF".



“Extra” – Not Just Nanoseconds!

- Only need 30 bits for nanosecond resolution
- Low-order two bits used to extend seconds field

Nanoseconds– shift right two for actual value
(aka “divide by four”)



Atime “extra”
0x00000001

Atime seconds
0x0954B156



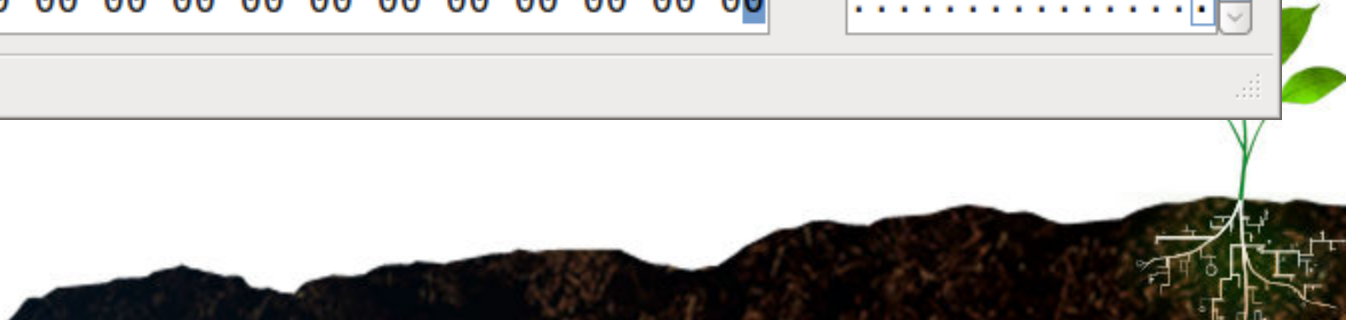
Extent Header (Bytes 40-51)

The screenshot shows a GHex window titled "testfile.inode - GHex" displaying hex data. The data is organized into columns with addresses on the left. Annotations with arrows point to specific fields:

- Generation ID:** Points to the first four bytes of the extent header (00 00 00 00).
- Magic Number:** Points to the next four bytes (0A F3 01 00).
- Number of Extents:** Points to the next two bytes (04 00).
- Max Poss Extents:** Points to the next two bytes (00 00).
- Depth of Tree:** Points to the next two bytes (36 87).

Address	Hex	Hex	Hex	Hex	Hex	Hex	Hex	Hex	Hex	Hex	Hex	Hex	Hex	Hex	Hex
00000000	A4	81	00	00	13	00	00	00	00	56	B1	54	09	ED	92
00000010	13	59	5E	84	00	00	00	00	00	00	01	00	08	00	00
00000020	00	00	08	00	01	00	00	00	00	0A	F3	01	00	04	00
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	36	87
00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000070	00	00	00	00	BD	9F	00	00	00	00	00	00	00	00	00
00000080	1C	00	00	00	18	BE	FF	CF	00	00	00	00	01	00	00
00000090	E4	92	7B	4D	6C	F0	8A	14	00	00	00	00	00	00	00
000000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Offset: FF



Extent Structure

testfile.inode - GHex

File Edit View Windows Help

00000000	A4	81	00	00	13	00	00	00	00	56	B1	54	09	ED	92	7B	4DV.T...{M
00000010	13	59	5E	84	00	00	00	00	00	00	00	01	00	08	00	00	00	.Y^.....
00000020	00	00	08	00	01	00	00	00	0A	F3	01	00	04	00	00	00	00
00000030	00	00	00	00	00	00	00	00	00	00	00	36	87	90	01	00	006...
00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000080	1C	00	00	00	00	00	00	00	CF	00	00	00	00	00	00	00	00
00000090	E4	92	7B	4D	6C	F0	8A	14	00	00	00	00	00	00	00	00	00{ML.....
000000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Logical Block Offset

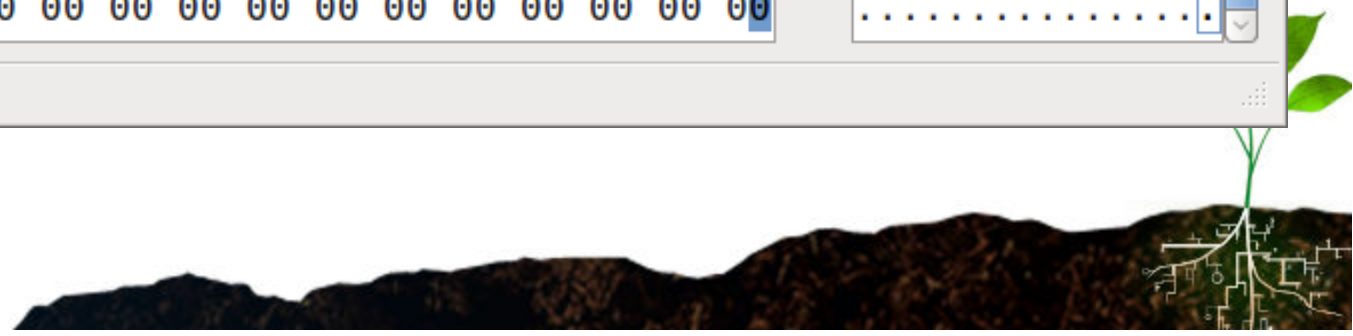
Length in Blocks

Phys Start Addr (upper 16 bits)

Phys Start Addr (lower 32 bits)

Start Address = 0x0000 01908736 = 26249014

Offset: FF



- Only 15 bits for extent length (high bit reserved)
 - *Max extent size is 128MB* (assuming 4K blocks)
- Only 4 extents per inode

What about large files (> 0.5GB)?

What about heavily fragmented files?



Extent Trees

The screenshot shows a hex editor window titled 'ino-721'. The main content is a hex dump of an extent index structure. The following table represents the data shown in the hex editor:

Offset	Hex	ASCII
00000000	A0 81 65 00 02 BD 0F 00 4E E7 8D 4D CF E9 8D 4D	..e....N..M...M
00000010	CF E9 8D 4D 00 00 00 00 04 00 00 00 E8 07 00 00	...M.....
00000020	00 00 08 00 01 00 00 00 0A F3 01 00 04 00 01 00
00000030	00 00 00 00 00 00 00 00 12 00 02 00 00 00 02 00
00000040	01 00 00 00 01 00 00 00 25 47 02 00 02 00 00 00%D.....
00000050	01 00 00 00 14 40 02 00 03 00 00 00 01 00 00 00@.....
00000060	19 40 02 00 2D 71 3A CA 00 00 00 00 00 00 00 00	...s-g:.....
00000070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000080	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000090	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Annotations and Callouts:

- One extent**: Points to the entire extent index structure.
- "Depth of Tree" is now one**: Points to the extent index structure.
- Extent Index struct**: Points to the extent index structure.
- Logical Block Offset**: Points to the first four bytes of the extent index structure.
- Phys Block Addr (lower 32 bits)**: Points to the next eight bytes of the extent index structure.
- Phys Block Addr (upper 16 bits)**: Points to the next four bytes of the extent index structure.
- (unused)**: Points to the last four bytes of the extent index structure.
- Block Address = 0x0000 00020012 = 131090**: A callout box showing the calculation of the block address from the physical block address fields.

Offset: FF

Block 131090 (Bytes 0-255)

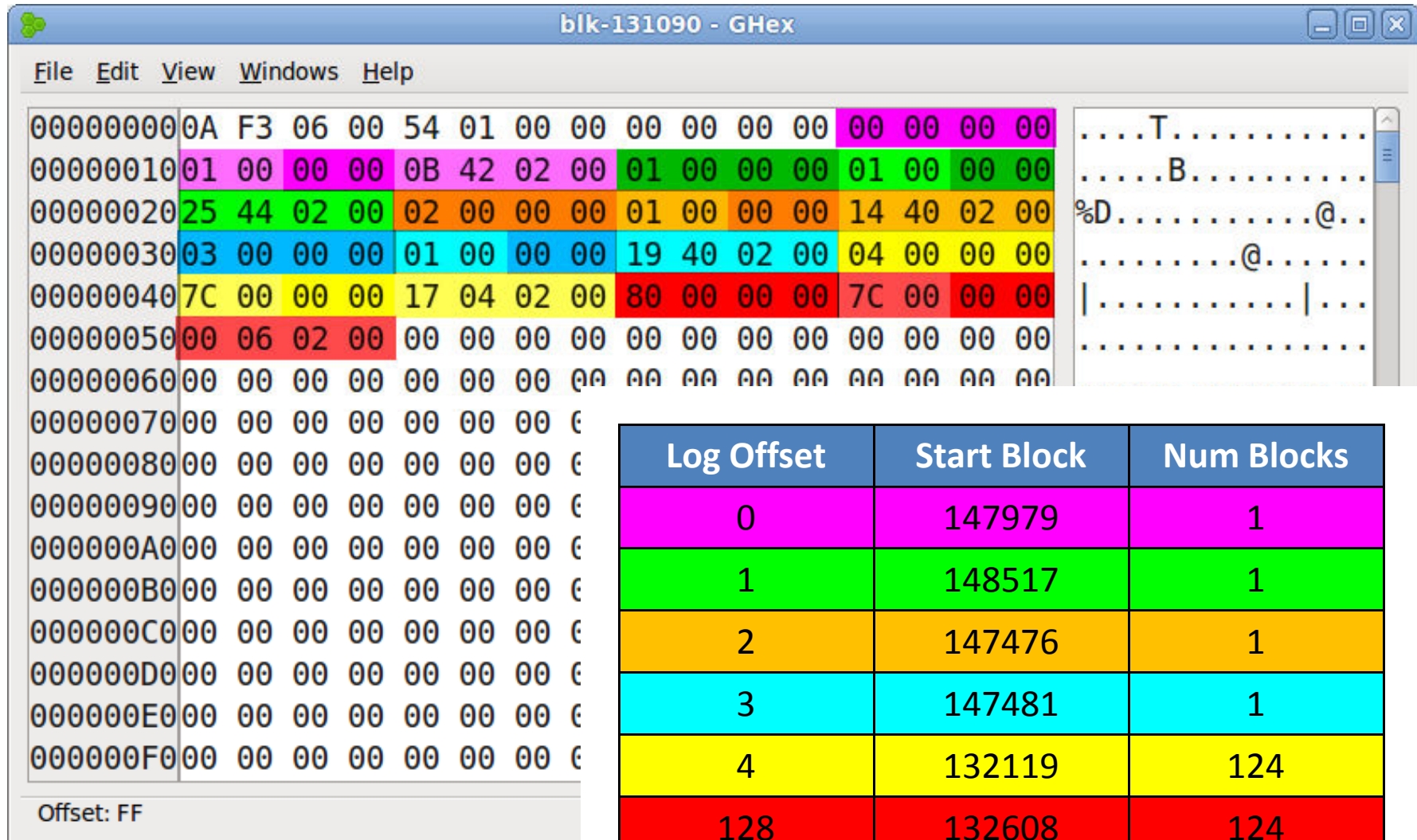
The image shows a hex editor window titled "blk-131090" with a menu bar (File, Edit, View, Windows, Help). The main area displays a hex dump of memory. The first row (offset 00000000) is highlighted in cyan and contains the hex values: 0A F3 06 00 54 01 00 00 00 00 00 00 00 00 00 00. Annotations with arrows point to specific fields in this row:

- "Magic Number for Extent Header" points to the first four bytes (0A F3 06 00).
- "Num Extents (6)" points to the fifth byte (54).
- "Max Extents (340!)" points to the sixth byte (01).
- "Depth of Tree (now zero)" points to the seventh byte (00).

The rest of the memory dump consists of zeroed-out bytes (00). On the right side of the window, there is a text pane showing a tree structure with nodes labeled "T", "B", and "%D".

Offset: FF

Block 131090 - Extents



The screenshot shows a GHex window titled 'blk-131090 - GHex'. The main pane displays hex data from offset 00000000 to 000000F0. The data is color-coded: 00000000-00000001 (magenta), 00000002 (green), 00000003 (orange), 00000004 (cyan), 00000005 (yellow), and 00000006-00000007 (red). The ASCII view on the right shows characters: '.....T.....', '.....B.....', '%D.....@..', '.....@.....', '|.....|...', and '.....'. The status bar at the bottom left shows 'Offset: FF'.

Log Offset	Start Block	Num Blocks
0	147979	1
1	148517	1
2	147476	1
3	147481	1
4	132119	124
128	132608	124



Testing Those Numbers

CEIC[®] 2011

```
# blkcat /dev/mapper/RD-var 147979 >ext1-blks
# blkcat /dev/mapper/RD-var 148517 >ext2-blks
# blkcat /dev/mapper/RD-var 147476 >ext3-blks
# blkcat /dev/mapper/RD-var 147481 >ext4-blks
# blkcat /dev/mapper/RD-var 132119 124 >ext5-blks
# blkcat /dev/mapper/RD-var 132608 124 >ext6-blks
# cat ext* | tr -d \\000 >newmess
# md5sum newmess /var/log/messages
8e8c9445d8ff3e17a22ef5a3034422a9  newmess
8e8c9445d8ff3e17a22ef5a3034422a9  /var/log/messages
```



- What was all that junk in the inode?
 - Extents 2-4 were populated but not used
 - “Unused” bytes in extent index had data in them
- EXT4 developers were ~~lazy~~ efficient:
 - Data in inode not zeroed when extent tree needed
 - Inode extents 2-4 match block 131090 extents 2-4
 - “Unused” bytes in extent index from old extent #1



What About File Deletion?

CEIC[®] 2011

- How are timestamps impacted?
- What about extent structures?
- Extent trees in data blocks cleaned up?



Post-Deletion Extent Structs

ino-7210-postdel

File Edit View Windows Help

File size, Num Extents, and Depth of Tree zeroed

00000000	A0	81	65	00	00	00	00	00	F6	41	8E	4D	25	43	8E	4D	...
00000010	25	43	8E	4D	25	43	8E	4D	04	00	00	00	00	00	00	00	%C.M%C.M.....
00000020	00	00	08	00	01	00	00	00	0A	F3	00	00	04	00	00	00
00000030	00	00	00	00	00	00	00	00	12	00	02	00	00	00	00	02
00000040	01	00	00	00	01	00	00	00	25	44	02	00	02	00	00	00%D.....
00000050	01	00	00	00	14	40	02	00	03	00	00	00	01	00	00	00@.....
00000060	19	40	02	00	2D	71	3A	CA	00	00	00	00	00	00	00	00	..@..-q:.....
00000070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00000080	1C	00	00	00	80	81	08	77	FC	B4	58	74	9C	5B	5E	36w..Xt.[^6
00000090	B2	17	86	4D	8C	C2	14	D7								
000000A0	00	00	00	00	00	00	00	00								
000000B0	00	00	00	00	00	00	00	00								
000000C0	00	00	00	00	00	00	00	00								
000000D0	00	00	00	00	00	00	00	00								
000000E0	00	00	00	00	00	00	00	00								
000000F0	00	00	00	00	00	00	00	00								

Offset: FF

- Extent Index untouched
- Residue remains in unused extents

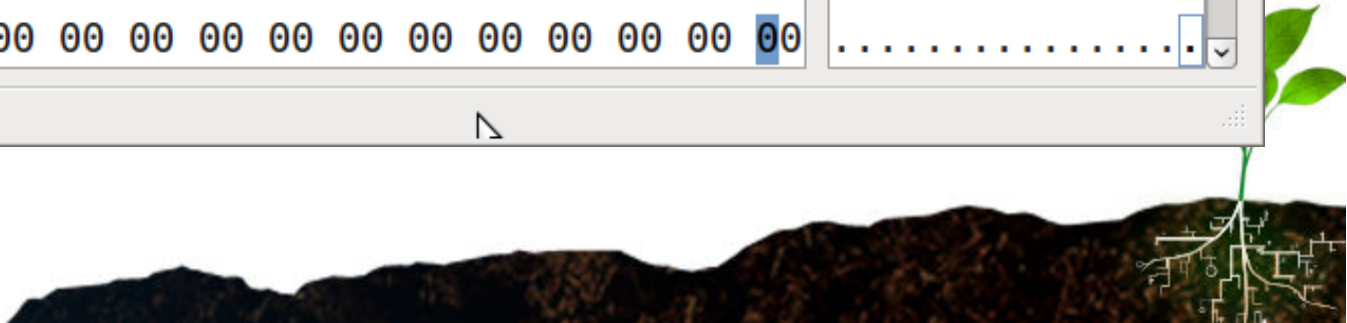
Block 131090 Post-Deletion

Number of Extents zeroed

00000000	0A F3	00 00	54 01 00 00 00 00 00 00	00 00 00 00
00000010	00 00 00 00	00 00 00 00	01 00 00 00	00 00 00 00
00000020	00 00 00 00	02 00 00 00	00 00 00 00	00 00 00 00
00000030	03 00 00 00	00 00 00 00	00 00 00 00	04 00 00 00
00000040	00 00 00 00	00 00 00 00	80 00 00 00	00 00 00 00
00000050	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000060	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000070	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000080	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
00000090	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000000A0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000000B0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000000C0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000000D0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000000E0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000000F0	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00

Upper 8 bytes of extents zeroed but logical block offsets remain. Seriously, WTF?

Offset: FF



- Timestamps:
 - Deleted time (in [CMD]time fields)
 - Last access time* and original creation time
- Extents
 - Data block address in extent index(es) [if any]
 - Unused extent structs in inode [if any]
 - Logical block offsets in extent structs
 - [allows extent sizes to be inferred in some cases]



- Any final questions?
- Thanks for listening!

Hal Pomeranz hal@deer-run.com

hal@sans.org

<http://www.deer-run.com/~hal/>

<http://computer-forensics.sans.org/blog/author/halpomeranz/>

<http://www.sans.org/security-training/instructors/Hal-Pomeranz>

https://twitter.com/hal_pomeranz

