EXT4: Bit by Bit

Hal Pomeranz
Deer Run Associates

What’s New in EXT4?

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

Backwards Compatibility

- 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

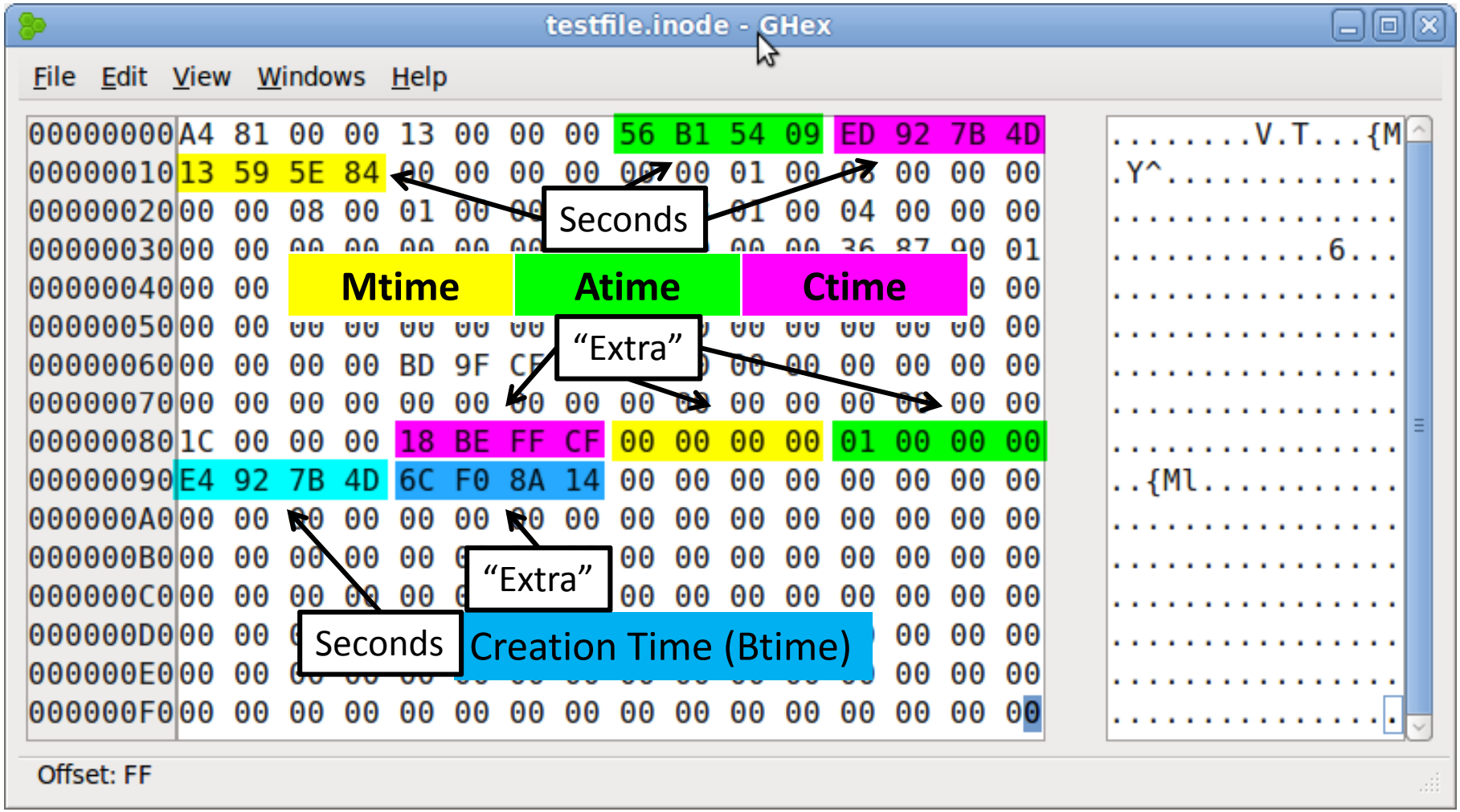
Let's Make a File!

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

No fractional seconds!

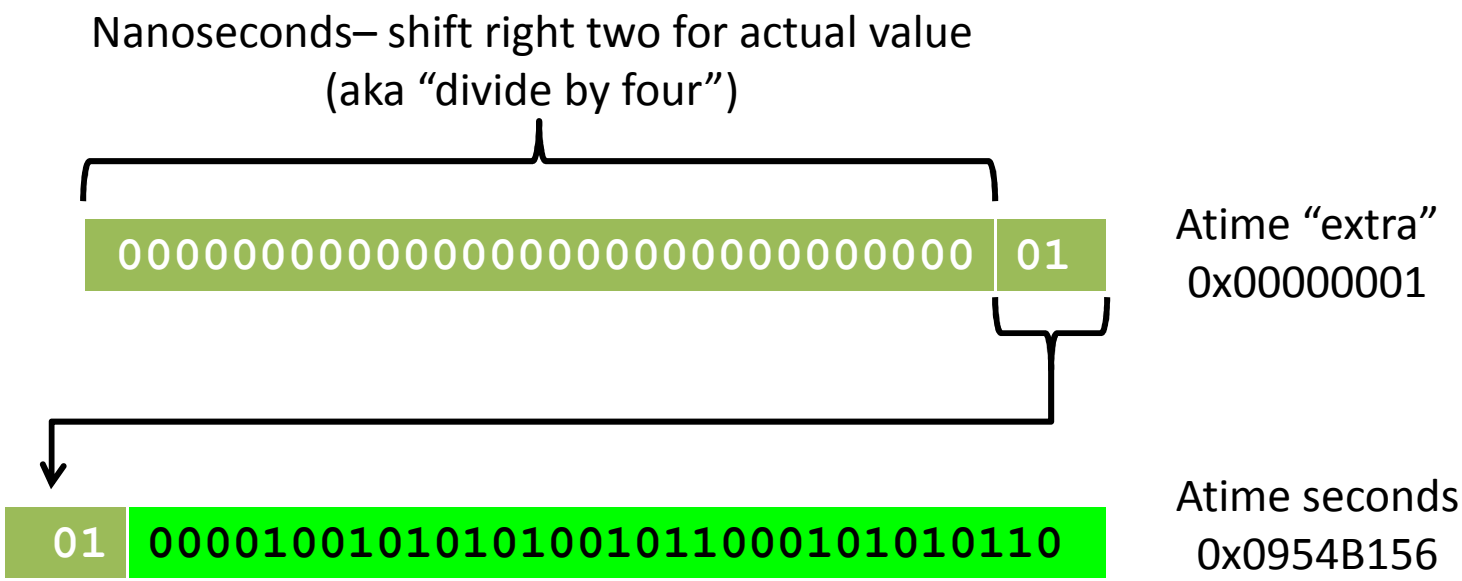
	stat	istat	debugfs
Access	2111-01-23 19:17:42.0	1974-12-17 12:49:26	1974-12-17 12:49:26.0
Modify	2040-05-16 03:08:19.0	2040-05-16 03:08:19	2040-05-16 03:08:19.0
Change	2011-03-12 07:36:13…	2011-03-12 07:36:13	2011-03-12 07:36:13…
Create	N/A	N/A	2011-03-12 07:36:04…

Timestamps In The Inode

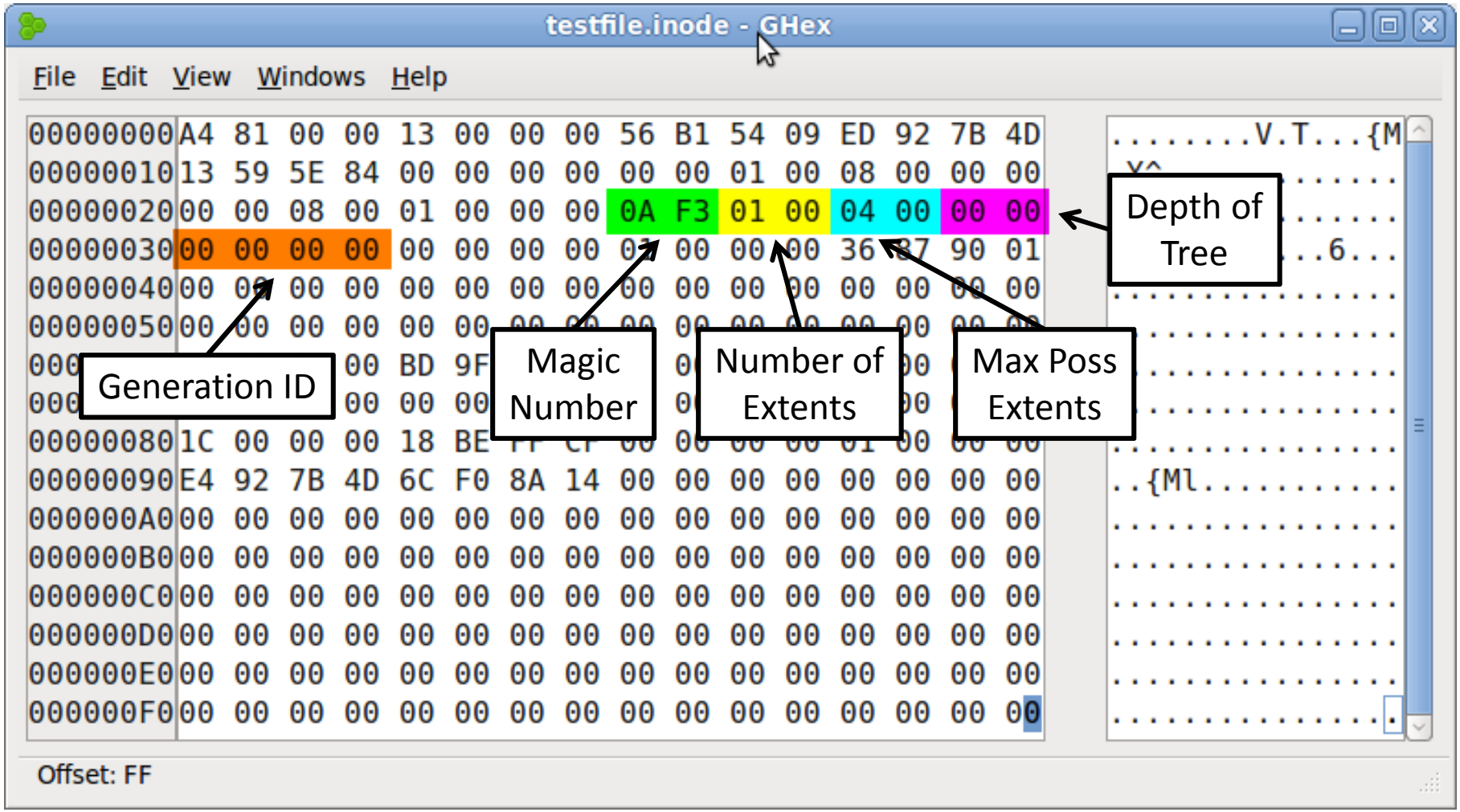


“Extra” – Not Just Nanoseconds!

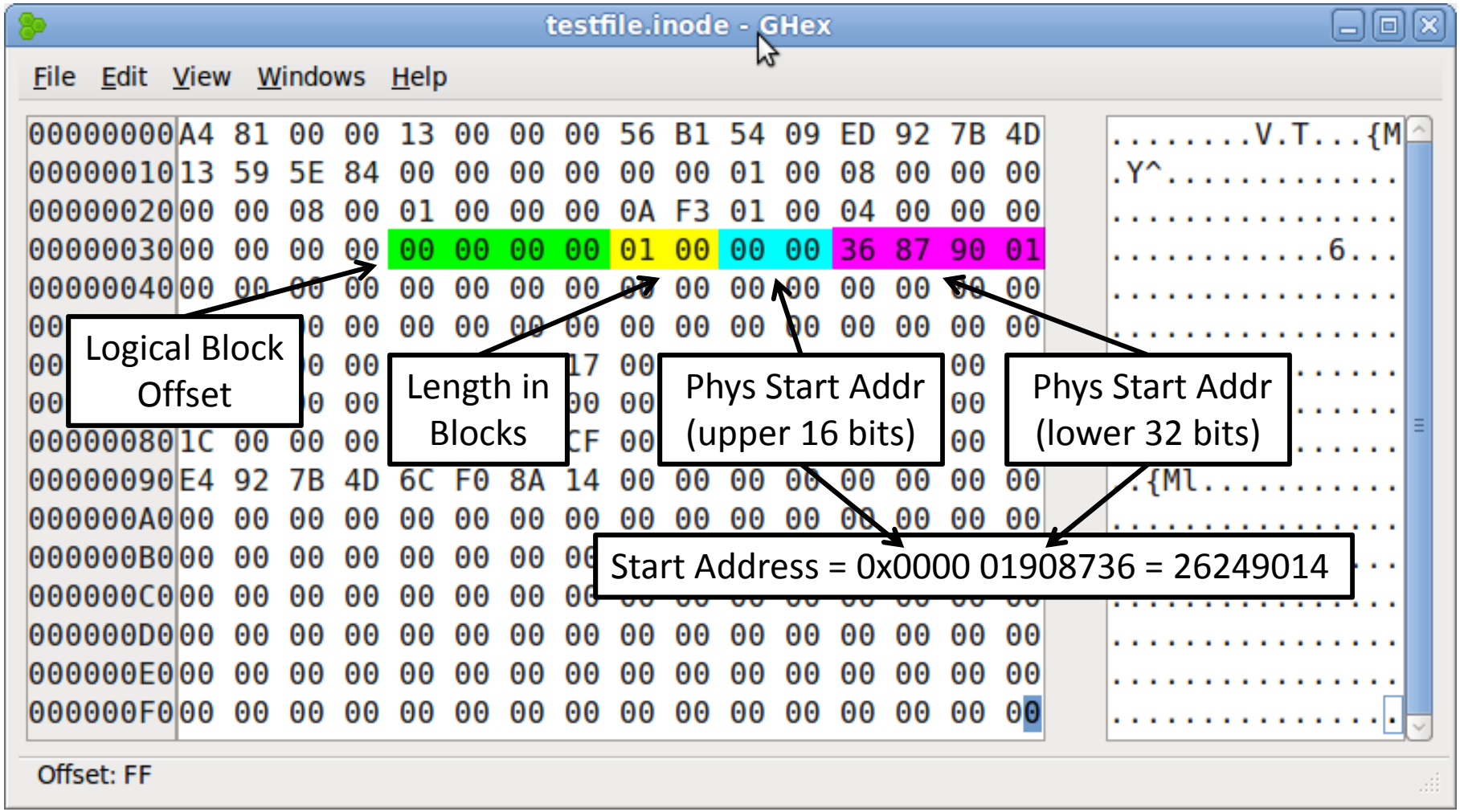
- Only need 30 bits for nanosecond resolution
- Low-order two bits used to extend seconds field



Extent Header (Bytes 40-51)



Extent Structure



Limitations

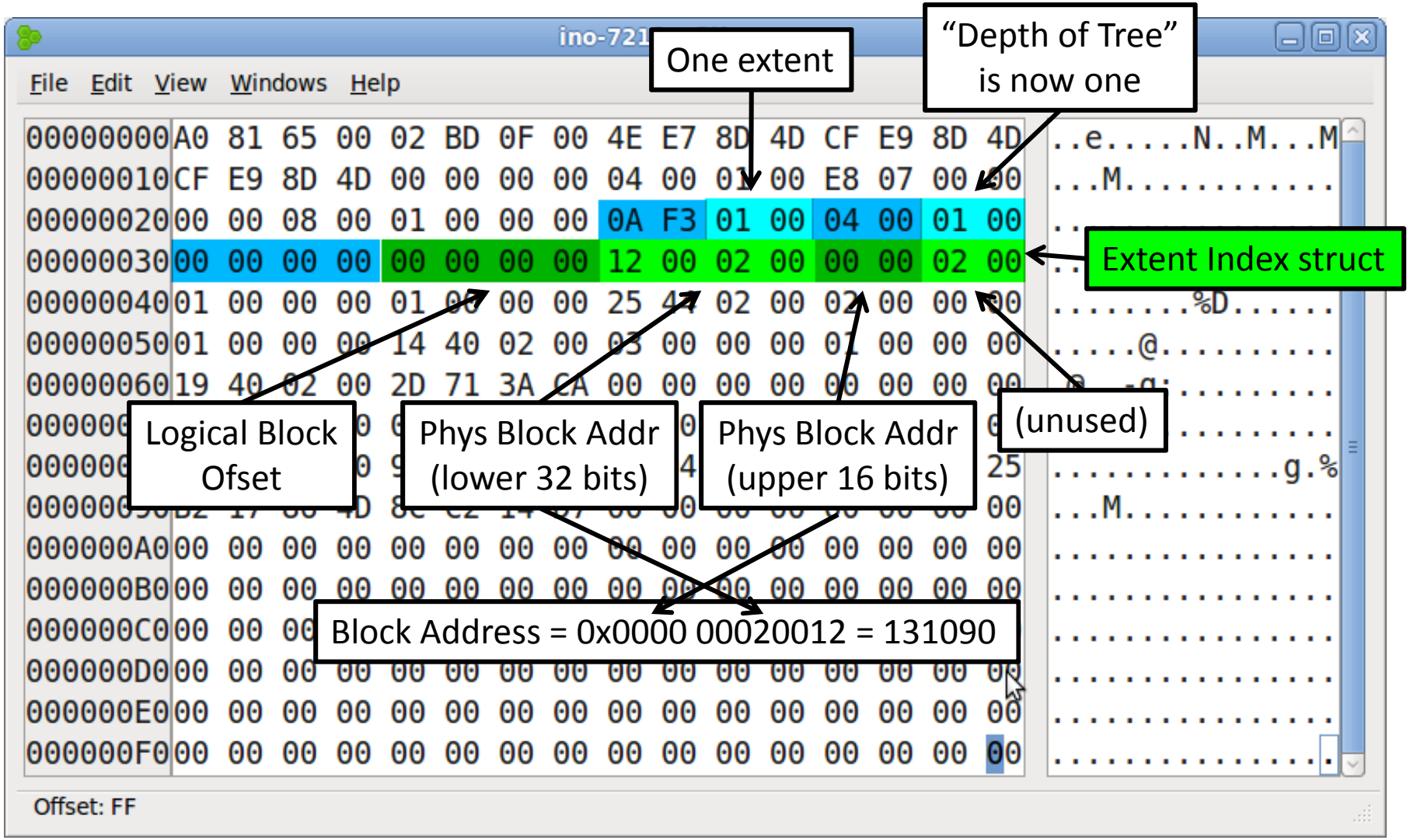
- Only 15 bits for extent length (high bit reserved)
  - *Max extent size is 128MB* (assuming 4K blocks)

- Only 4 extents per inode

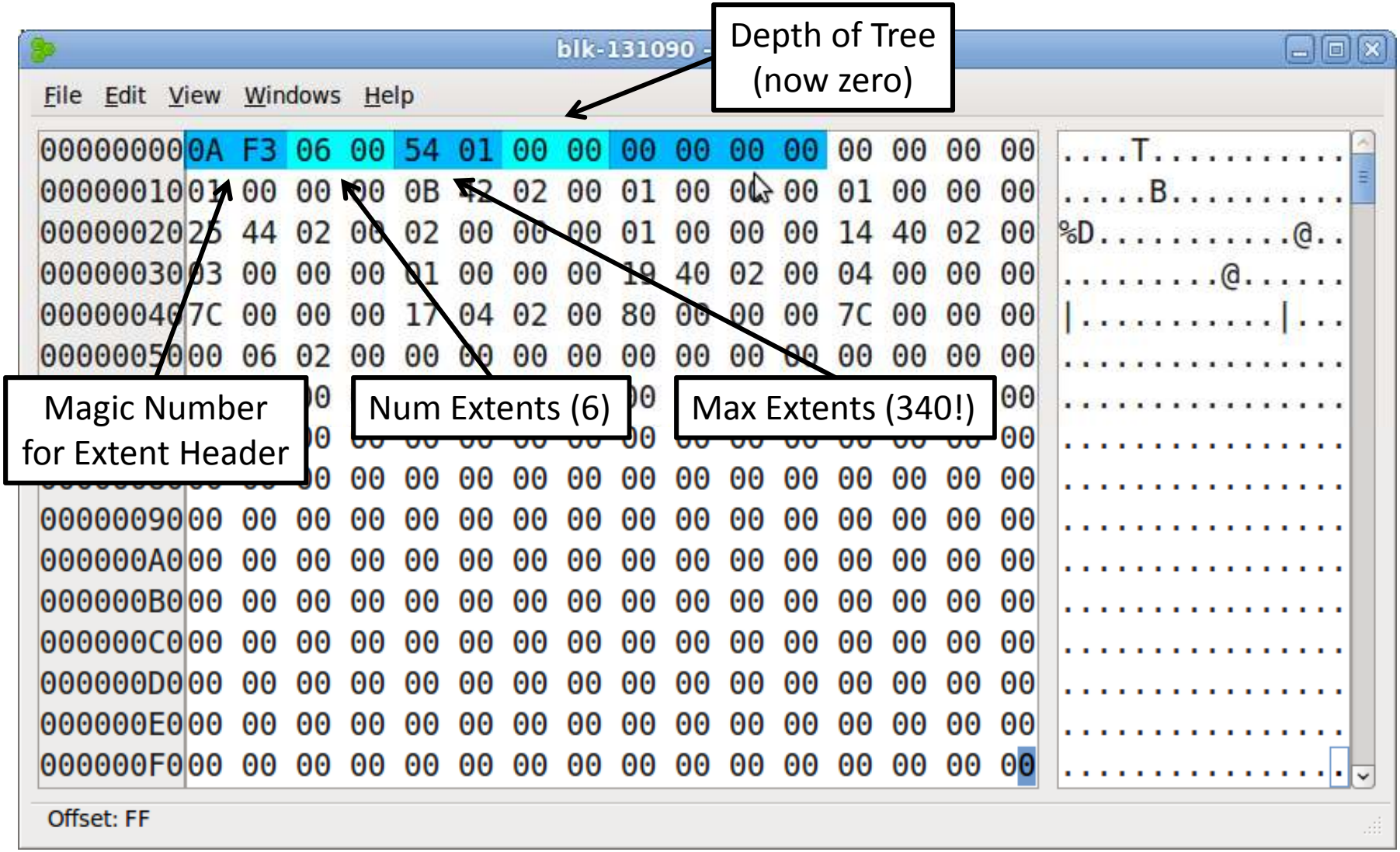
What about large files (> 0.5GB)?

What about heavily fragmented files?

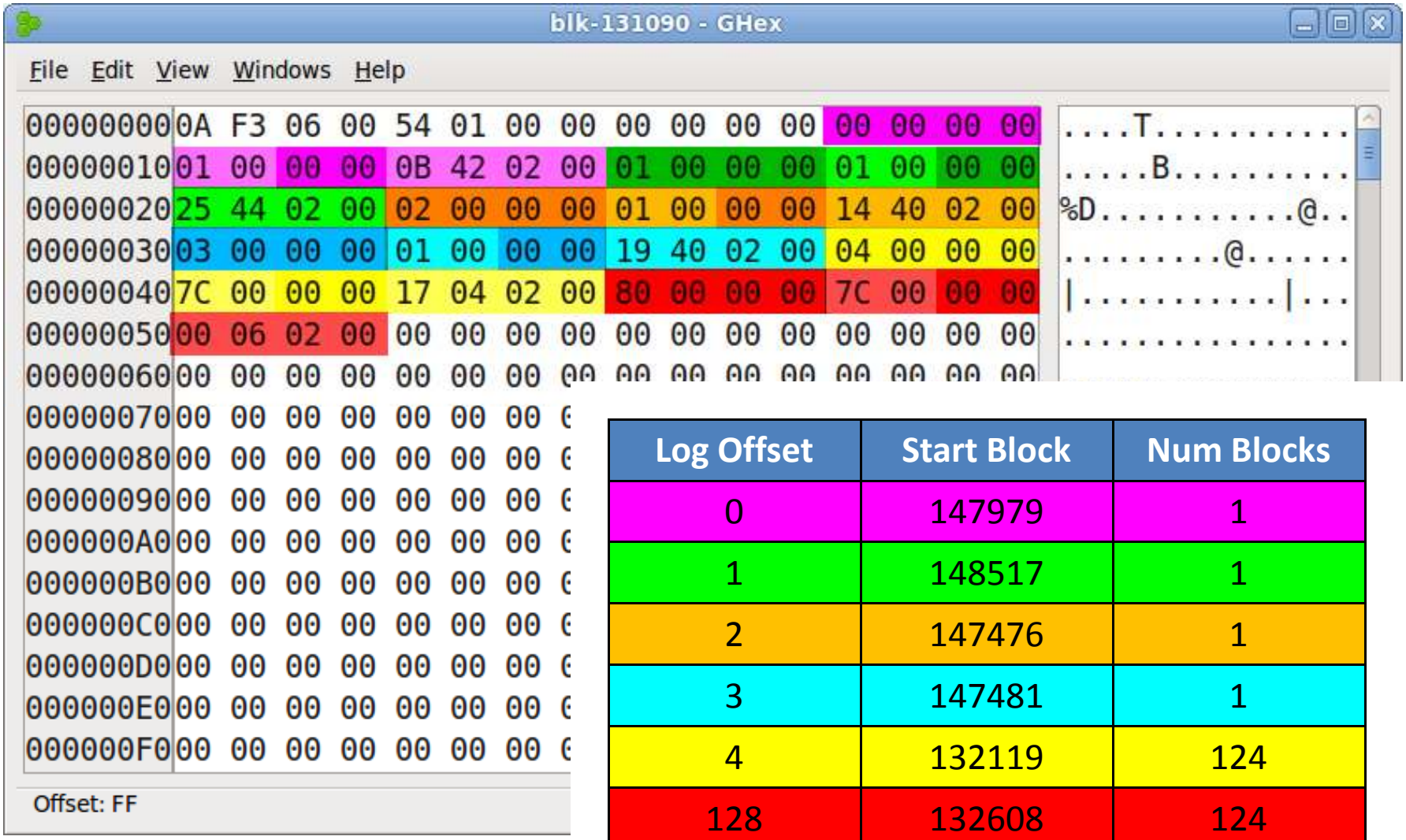
Extent Trees



Block 131090 (Bytes 0-255)



Block 131090 - Extents



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

```
# 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 About Inode Residue?

- 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?

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

Post-Deletion Timestamps

ino-7210-postdelete - GHex

File Edit View Windows Help

```
00000000 A0 81 65 00 00 00 00 00 F6 41 8E 4D 25 43 8E 4D  ..e......A.M%C.M
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 00 00 00 00 00 00 02 00  ................
00000040 01 00 00 00 01 00 00 00 02 00 00 00 00 00 25 44  ..............%D
00000050 01 00 00 00 14 40 00 00 01 00 00 00 00 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 36  .......w..Xt.[^6
00000090 B2 17 86 4D 8C C2 14 D7 00 00 00 00 00 00 00 00  ...M............
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  ................
```

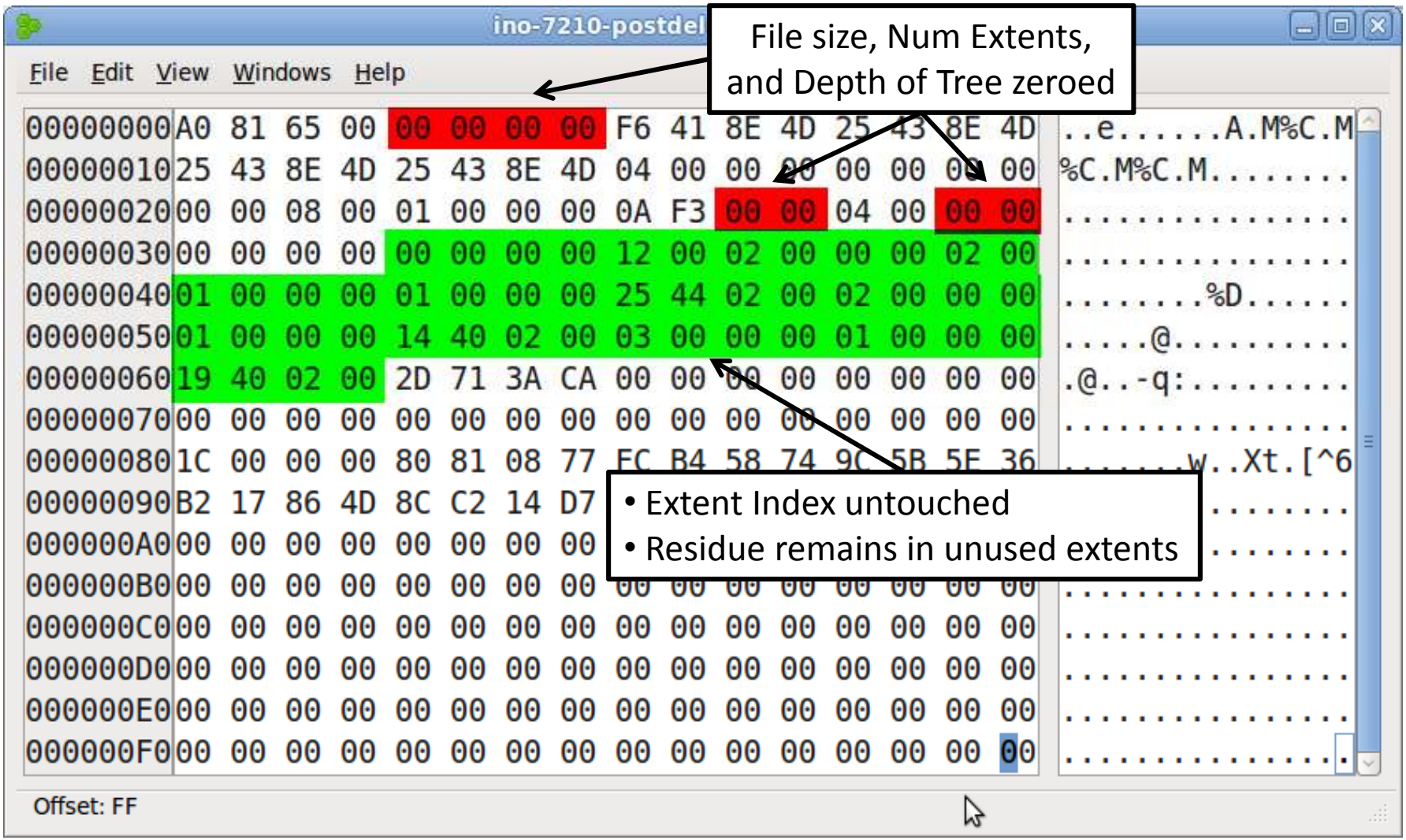
[CMD]time set to time file deleted

Atime unaltered

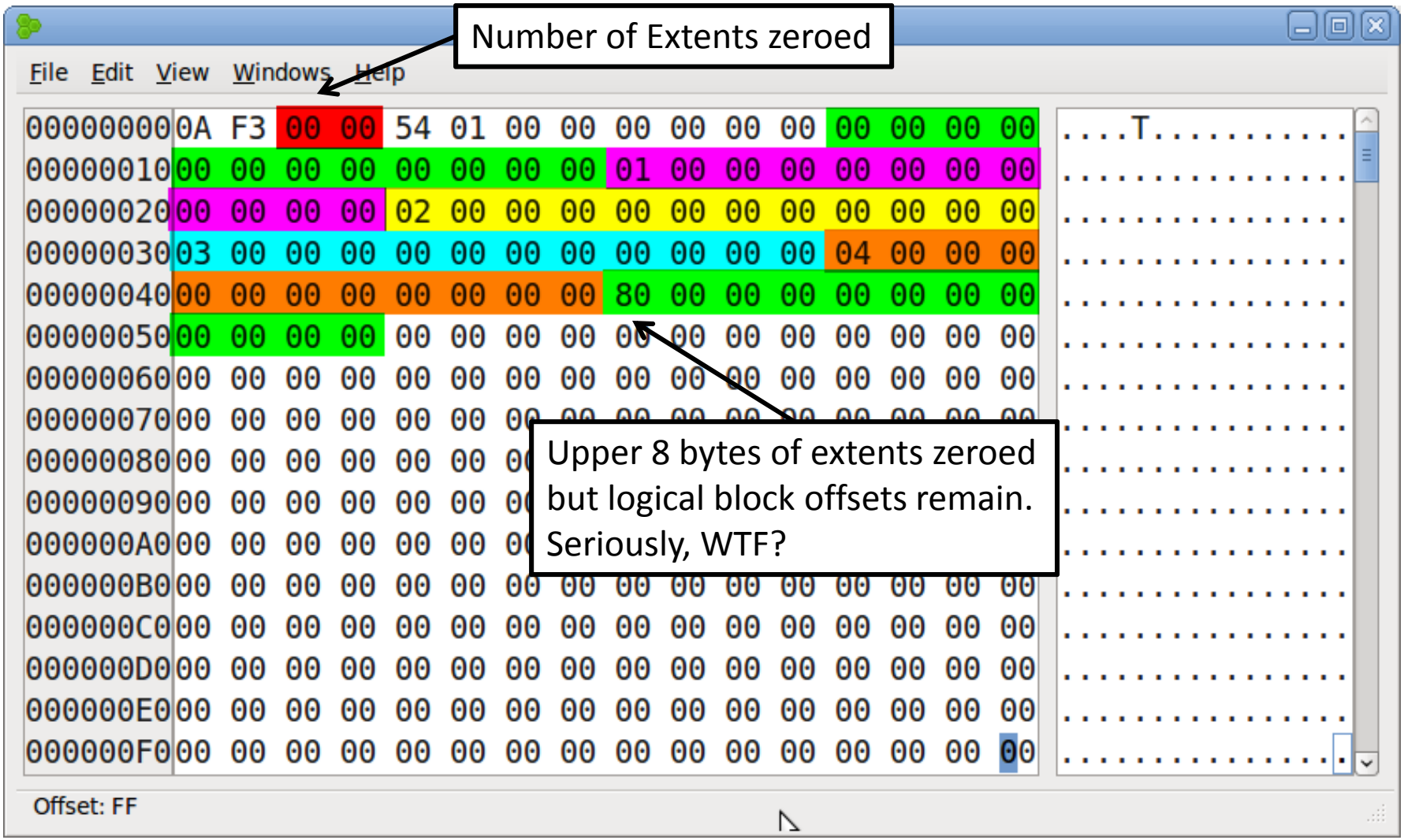
Btime unaltered

Offset: FF

Post-Deletion Extent Structs



Block 131090 Post-Deletion



Post-Deletion Summary

- 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]

Wrapping Up

- 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