EXT4: Bit by Bit

Hal Pomerantz
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!

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

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

No fractional seconds!
 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

Start Address = 0x0000 01908736 = 26249014
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

**Extent Index struct**

- **Logical Block Offset**
- **Phys Block Addr (lower 32 bits)**
- **Phys Block Addr (upper 16 bits)**

**Block Address = 0x0000 00020012 = 131090**

One extent

"Depth of Tree" is now one

(unused)
Block 131090 (Bytes 0-255)

- Magic Number for Extent Header
- Num Extents (6)
- Max Extents (340!)
- Depth of Tree (now zero)
Block 131090 - Extents

<table>
<thead>
<tr>
<th>Log Offset</th>
<th>Start Block</th>
<th>Num Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>147979</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>148517</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>147476</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>147481</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>132119</td>
<td>124</td>
</tr>
<tr>
<td>128</td>
<td>132608</td>
<td>124</td>
</tr>
</tbody>
</table>
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" >newmerr
# md5sum newmerr /var/log/messages
8e8c9445d8ff3e17a22ef5a3034422a9  newmerr
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

[CMD] time set to time file deleted

Atime unaltered

Btime unaltered
# Post-Deletion Extent Structs

![Post-Deletion Extent Structs](image.png)

- **File size, Num Extents, and Depth of Tree zeroed**
- **Extent Index untouched**
- **Residue remains in unused extents**
Block 131090 Post-Deletion

Number of Extents zeroed

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