Forensics II

Linux file systems Ext4, Btrfs and ZFS

Repetition begin

See forensic 1
http://users.du.se/~hjo/cs/dt1059/presentation/CF1_9_filesystems.pdf

ext4 is important since most Android devices run it ==
Actually most of the computers in the world!

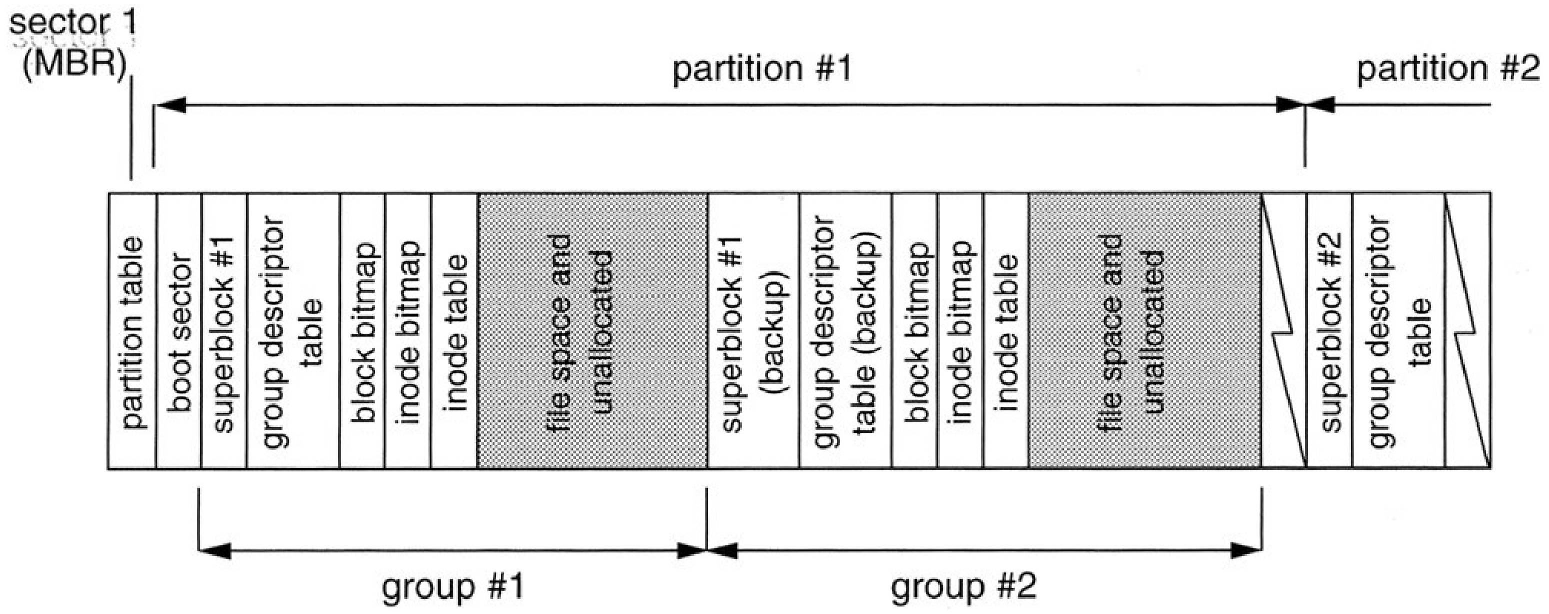
Slides from forensic 1 begin

UNIX filsystem UFS, ext2… osv.

- Använder datastrukturer som kallas för index noder i en tabell för att representera filer, bibliotek och symboliska länkar
- Inode fälten (128 byte) är av ett fixt antal och lagrar metadata
- Varje fil och mapp har ett associerat entry i inode tabellen
- Inodens nr som hanterar filen/mappen kan visas med ls -i
- Inode nr 1 används vanligen för att lagra bad blocks
- Inode nr 2 används alltid för root directory
- Bra program för lågnivå diskundersökning
  - The Sleuth Kit – fsstat och istat kommandot
  - Linux Disk Editor – lde
  - Tune2fs – visar filsystem info mm. för ext2/ext3
- Vissa icke traditionella UNIX filsystem har en ganska olik uppbyggnad på låg nivå tex. ReiserFS
  - Kräver sina egna program/verktyg

UNIX filsystem UFS, ext2… osv.

- Delar upp partitionen i ett antal block grupper för redundans, ca: 128MB (32k*4k) per grupp för file space
- Superblocket (1 kB) innehåller viktig filsysteminfo som block size, ant. block, block per grupp, last mounted mm.
  - Sparse superblock, group desc.
- Group descriptor håller reda på grupperna och var saker finns (bitmaps, inode table)
- Block/inode - bitmappar hanterar allokeringsstatus



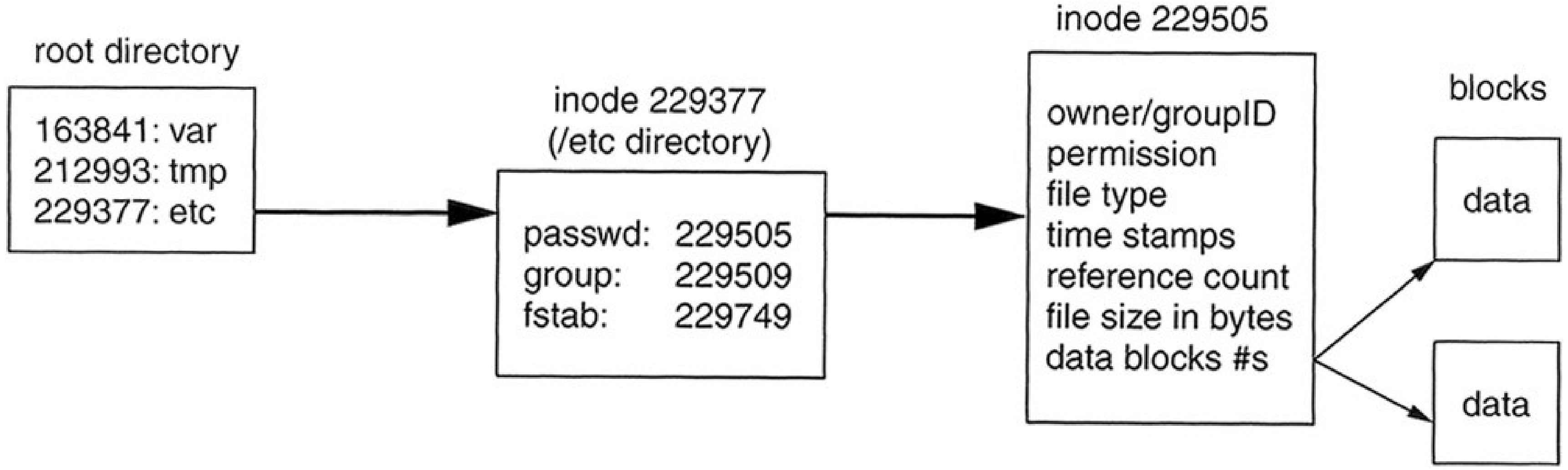
tune2fs kommandot

- Ger info om inoders index/fält- och block size
- Antalet inodes och blocks per grupp
- Mm. mm.

```
linuxbox:~# tune2fs -l /dev/hda2
tune2fs 1.40-WIP (14-Nov-2006)
Filesystem volume name:   <none>
Last mounted on:          <not available>
Filesystem UUID:          70be05e9-3e15-4456-be27-4153e420d320
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      filetype sparse_super
Default mount options:    (none)
Filesystem state:         not clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              14469312
Block count:              14460508
Reserved block count:     723025
Free blocks:              9935321
Free inodes:              14340692
First block:              0
Block size:               4096
Fragment size:            4096
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         32736
Inode blocks per group:   1023
Last mount time:          Sun Mar 28 21:06:36 2010
Last write time:          Mon Apr 26 21:31:38 2010
Mount count:              1
Maximum mount count:      37
Last checked:             Sun Mar 28 21:02:21 2010
Check interval:           15552000 (6 months)
Next check after:         Fri Sep 24 21:02:21 2010
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               128
```

UNIX filsystem, slå upp fil

- När systemet skall visa en viss fil tex. /etc/passwd går man först till superblocket för att hitta inod 2 (root directory)
- Man letar sedan upp mappen ”etc” i blocket som inodens info lagras i
- När ”etc” hittats går man till den inode som ”etc” pekar på och letar efter ”passwd” i dess info data block
- När ”passwd” hittats går man till ”passwd” inodens info och de data block ”passwd” inoden refererar till och kan slutligen läsa in själva fildatat



Inode 2 (root directory) -> block 5

Linux Disk Editor – lde

block 5   4096 stort

```
linuxbox:~# lde -i 2 /dev/hda2
Device "/dev/hda2" is mounted, be careful
User requested autodetect filesystem. Checking device . . .
Found ext2fs on device.
Warning: First block (0) != Normal first block (1)
-----------------------------------------------------------------------------
INODE: 2      (0x00000002)
drwxr-xr-x       root     root         4096 Sun Dec 24 01:10:00 2006
TYPE:                 directory
LINKS:                21
MODEFLAGS.MODE:        004.0755
SIZE:              4096
BLOCK COUNT:           8
UID:               00000 (root)
GID:               00000 (root)
ACCESS TIME:           Tue Apr 28 10:14:37 2009
CREATION TIME:         Sun Dec 24 01:10:00 2006
MODIFICATION TIME:     Sun Dec 24 01:10:00 2006
DELETION TIME:         Thu Jan  1 01:00:00 1970
DIRECT BLOCKS:         0x00000005


INDIRECT BLOCK:
DOUBLE INDIRECT BLOCK:
TRIPLE INDIRECT BLOCK:
```

```
linuxbox:~# lde -b 5 /dev/hda2
0x00005000  02 00 00 00 0C 00 01 02 : 2E 00 00 00 02 00 00 00  ................
0x00005010  0C 00 02 02 2E 2E 00 00 : 0B 00 00 00 14 00 0A 02  ................
0x00005020  6C 6F 73 74 2B 66 6F 75 : 6E 64 00 00 0C 00 00 00  lost+found......
0x00005030  0C 00 03 02 65 74 63 00 : 23 05 00 00 0C 00 04 02  ....etc.#.......
0x00005040  72 6F 6F 74 59 16 00 00 : 0C 00 03 02 74 6D 70 00  rootY.......tmp.
0x00005050  79 16 00 00 0C 00 04 02 : 62 6F 6F 74 8C 16 00 00  y.......boot....
0x00005060  10 00 07 07 76 6D 6C 69 : 6E 75 7A 00 8D 16 00 00  ....vmlinuz.....
0x00005070  0C 00 03 02 6C 69 62 00 : 29 19 00 00 0C 00 03 02  ....lib.).......
0x00005080  75 73 72 00 0F EE 00 00 : 0C 00 04 02 73 62 69 6E  usr.........sbin
0x00005090  B3 EE 00 00 0C 00 03 02 : 76 61 72 00 5F 16 01 00  ........var._...
0x000050A0  0C 00 03 02 62 69 6E 00 : B5 16 01 00 0C 00 03 02  ....bin.........
0x000050B0  64 65 76 00 A9 2A 01 00 : 0C 00 04 02 68 6F 6D 65  dev..*......home
0x000050C0  AC 41 01 00 0C 00 03 02 : 6D 6E 74 00 AE 41 01 00  .A......mnt..A..
0x000050D0  0C 00 04 02 70 72 6F 63 : AF 41 01 00 0C 00 03 02  ....proc.A......
0x000050E0  6F 70 74 00 B0 41 01 00 : 10 00 06 02 66 6C 6F 70  opt..A......flop
0x000050F0  70 79 00 00 B1 41 01 00 : 10 00 05 02 63 64 72 6F  py...A......cdro
0x00005100  6D 00 00 00 B2 41 01 00 : 10 00 06 02 69 6E 69 74  m....A......init
0x00005110  72 64 00 00 B3 41 01 00 : EC 0E 03 02 73 79 73 00  rd...A......sys.
0x00005120  00 00 00 00 E0 0E 05 02 : 2E 72 6F 6F 74 00 00 00  .........root...
0x00005130  00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00  ................
```

ls -i kommandot visar filens inod i inode table

Inode structure

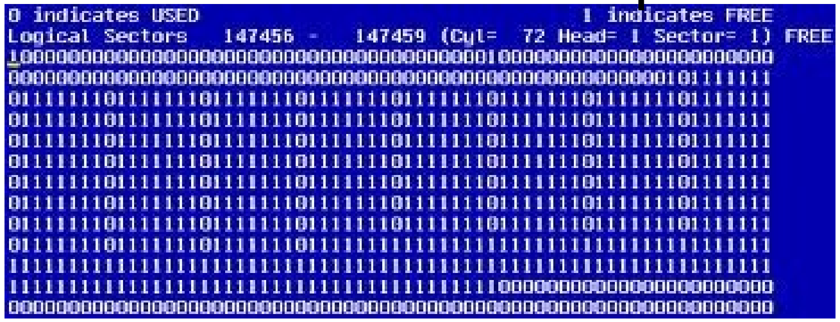
```
linuxbox:~# lde -i 1636804 /dev/hda2
Device "/dev/hda2" is mounted, be careful
User requested autodetect filesystem. Checking device . . .
Found ext2fs on device.
Warning: First block (0) != Normal first block (1)
---------------------------------------------------------------------------
INODE: 1636804 (0x0018F9C4)
-rwxr--r--     hjo     hjo     17923572 Mon Apr 13 11:45:58 2009
TYPE:                  regular file
LINKS:                 1
MODEFLAGS.MODE:        010.0744
SIZE:                  17923572
BLOCK COUNT:           35056
UID:                   01000 (hjo)
GID:                   01000 (hjo)
ACCESS TIME:           Mon Apr 13 11:45:58 2009
CREATION TIME:         Mon Apr 13 11:46:04 2009
MODIFICATION TIME:     Mon Apr 13 11:45:58 2009
DELETION TIME:         Thu Jan  1 01:00:00 1970
DIRECT BLOCKS:         0x0019500A 0x0019500B 0x0019500C
0x0019500D 0x0019500E 0x0019500F 0x00195010 0x00195011
0x00195012 0x00195013 0x00195014 0x00195015
INDIRECT BLOCK:        0x00195016
DOUBLE INDIRECT BLOCK: 0x00196B05
TRIPLE INDIRECT BLOCK:
```

= 512 byte block

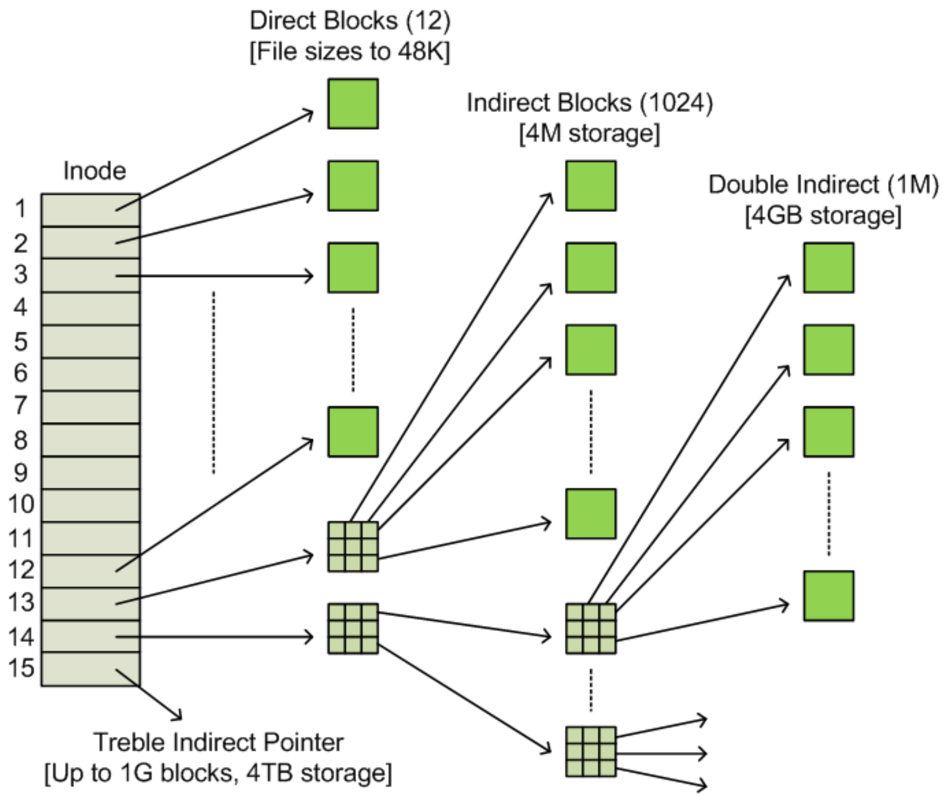
Byte Range	Description	Essential
0–1	File mode (type and permissions) (see Tables 15.11, 15.12, and 15.13)	Yes
2–3	Lower 16 bits of user ID	No
4–7	Lower 32 bits of size in bytes	Yes
8–11	Access Time	No
12–15	Change Time	No
16–19	Modification time	No
20–23	Deletion time	No
24–25	Lower 16 bits of group ID	No
26–27	Link count	No
28–31	Sector count	No
32–35	Flags (see Table 15.14)	No
36–39	Unused	No
40–87	12 direct block pointers	Yes
88–91	1 single indirect block pointer	Yes
92–95	1 double indirect block pointer	Yes
96–99	1 triple indirect block pointer	Yes
100–103	Generation number (NFS)	No
104–107	Extended attribute block (File ACL)	No
108–111	Upper 32 bits of size / Directory ACL Yes /	No
112–115	Block address of fragment	No
116–116	Fragment index in block	No
117–117	Fragment size	No
118–119	Unused	No
120–121	Upper 16 bits of user ID	No
122–123	Upper 16 bits of group ID	No
124–127	Unused	No

UNIX filsystem forts.

Block/inode - bitmap



```
0 indicates USED                            1 indicates FREE
Logical Sectors   147456 -   147459 (Cyl=  72 Head= 1 Sector= 1) FREE
1000000000000000000000000000000000000000100000000000000000000000
0000000000000000000000000000000000000000000000000000000101111111
0111111101111111011111110111111101111111011111110111111101111111
0111111101111111011111110111111101111111011111110111111101111111
0111111101111111011111110111111101111111011111110111111101111111
0111111101111111011111110111111101111111011111110111111101111111
0111111101111111011111110111111101111111011111110111111101111111
0111111101111111011111110111111101111111011111110111111101111111
0111111101111111011111110111111101111111011111110111111101111111
0111111101111111011111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
```



- Inoden har pekare till block där data lagras
  - Vid stora filer lagrar dessa istället pekare till nya block, upp till 3 ggr.
- Det finns speciella filtyper som inte lagrar data
  - Pekare till hårdvara, symbolisk länk etc. allt är filer i UNIX!
- Raderad fil fungerar lite olika i ext2 och ext3 filsystem
  - ext2fs markerar inoder med block pekare som lediga i block bitmaps och markerar ”info inoden” som ”deleted” i inode bitmap - men låter block pekarna stå kvar i inoden
  - ext3fs nollställer även block pekarna i inoder med block pekare
- Det finns inga verktyg för att hantera journalen i journalbaserade filsystem ännu som tex. ext3?

Ext3 delete

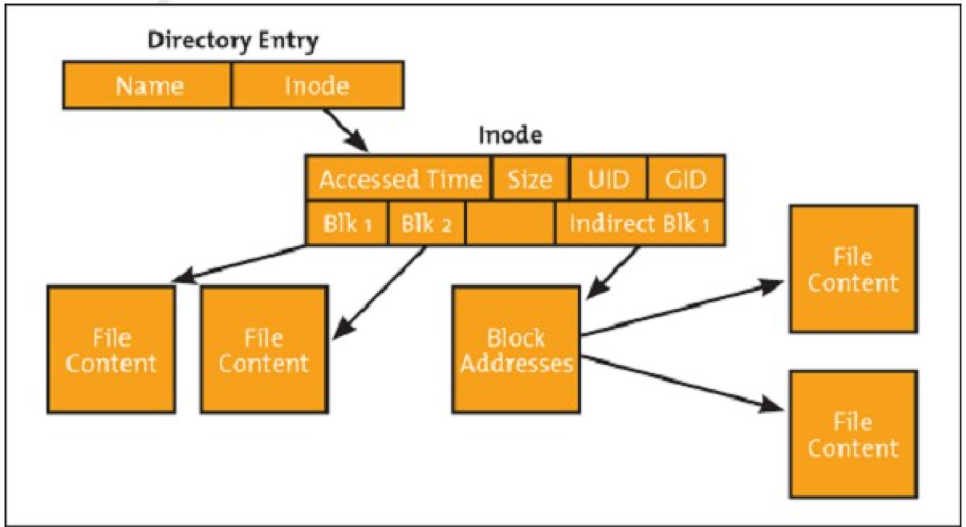


FIGURE 1 RELATIONSHIP BETWEEN THE DIRECTORY ENTRY, AN INODE, AND BLOCKS OF AN ALLOCATED FILE

Before deletion

HOWTO recover deleted files on an ext3 file system

http://www.xs4all.nl/~carlo17/howto/undelete_ext3.html

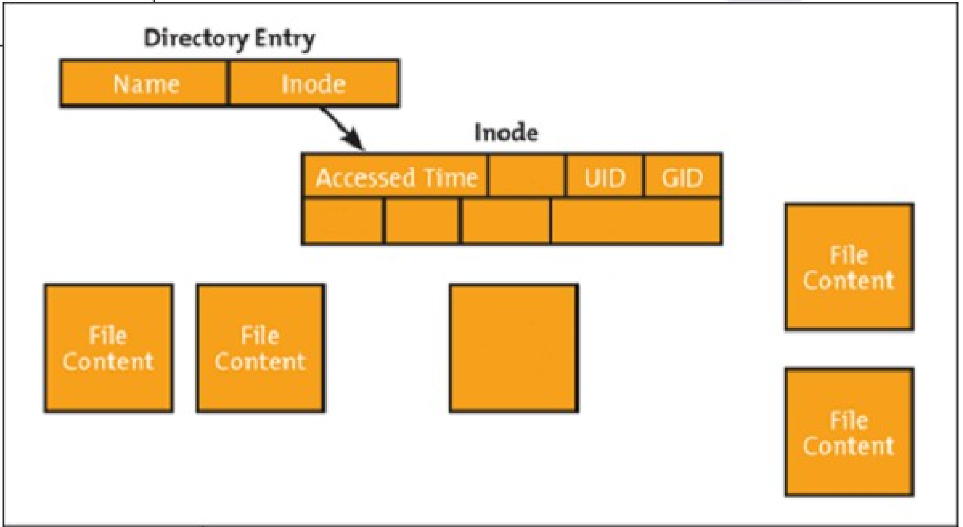


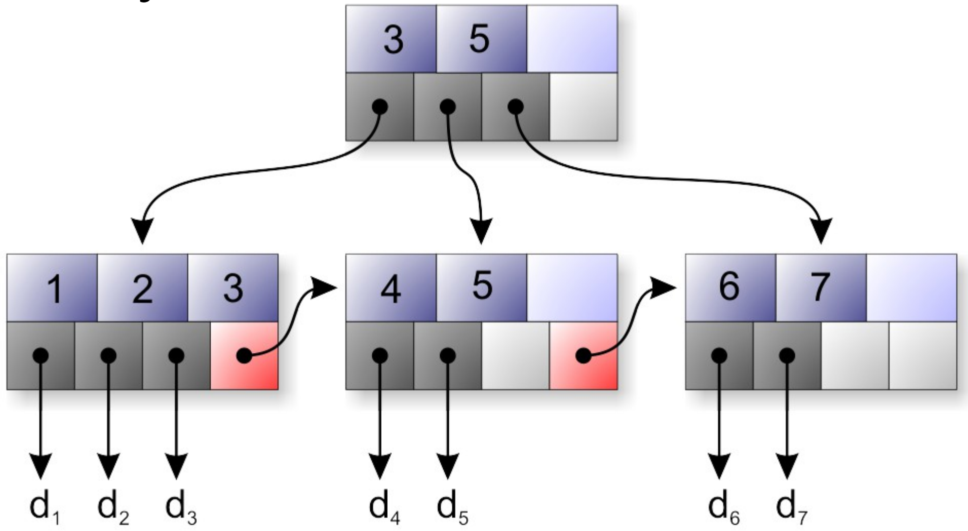
FIGURE 2 RELATIONSHIP BETWEEN THE DIRECTORY ENTRY, AN INODE, AND BLOCKS OF AN UNALLOCATED EXT3 FILE. THE LINKS BETWEEN THE INODE AND BLOCKS HAS BEEN CLEARED.

After deletion

Block pointers are zeroed out in the inode

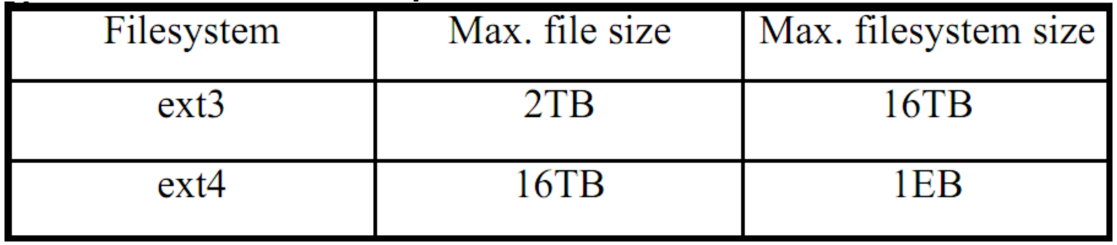
B-tree/B+ tree (not a binary tree)

- Representerar sorterad data som medger effektiv insättning, hämtning och borttagning av poster, samt indexering av metadata i filsystem och databaser
  - http://en.wikipedia.org/wiki/B%2B_tree
  - http://en.wikipedia.org/wiki/B-tree
- Används av NTFS, HFS, ReiserFS, XFS, JFS2, btrfs, ext4 mm.
- Ett enkelt B+ träd som länkar nycklarna 1-7 till datavärdena d1-d7
  - Den länkade listan (rött) medger snabb in-order traversering



Ext4 file system

- Ext4 is the successor of ext3 which is developed to solve performance issues and scalability bottleneck on ext3 and also provide backward compatibility with ext3
- Ext4 features
  - **Bigger file/filesystem size support (assuming 4 kB blocks):** because the block pointers are 48 bits instead of 32 bits

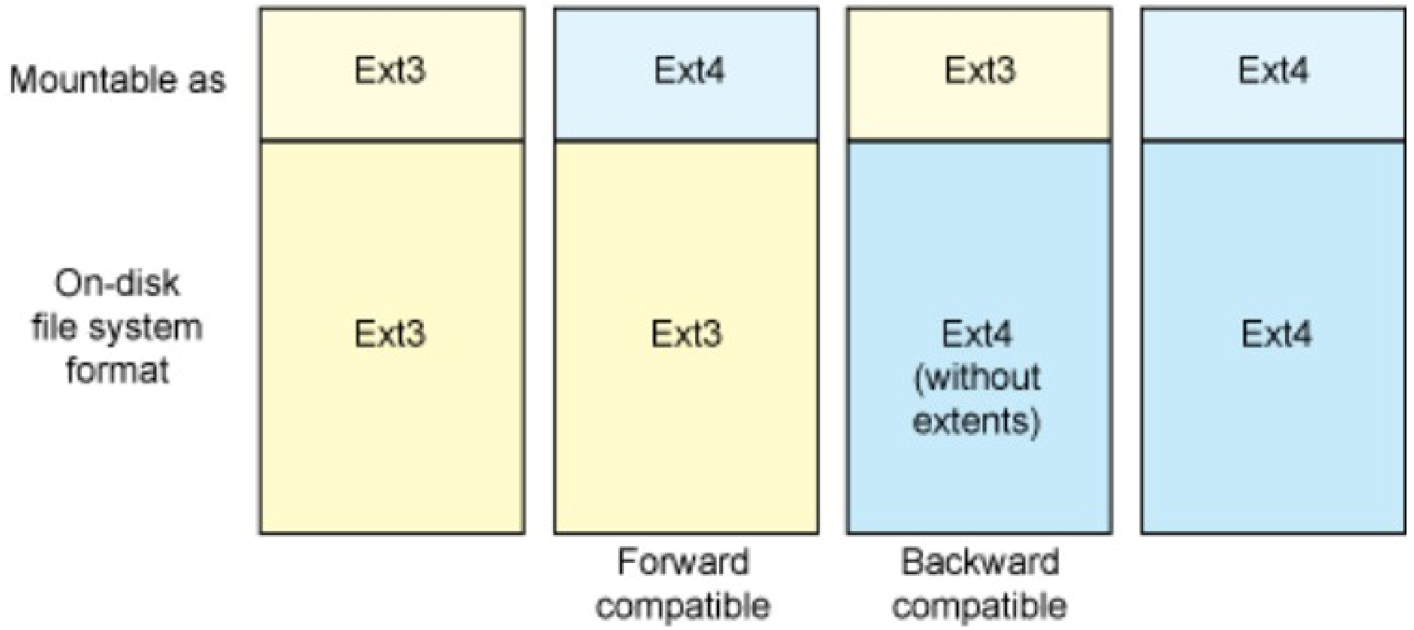


Filesystem	Max. file size	Max. filesystem size
ext3	2TB	16TB
ext4	16TB	1EB

  - **I/O performance improvement:** delayed allocation, multi block allocator extent map and persistent preallocation
  - **Fast fsck:** flex_bg and uninit_bg (file system feature flags)
  - **Reliability:** journal checksumming
  - **Maintenance:** online defrag
  - **Misc:** backward compatibility with ext2/ext3, nanosec timestamps, subdir scalability, etc.

Ext4 file system - compability

- When you want to migrate an ext3 file system to ext4, you can do so gradually
  - This means that old files that you have not moved can remain in the older ext3 format, while new files (or older files that have been copied) will occupy the new ext4 data structures
  - In this way, you can migrate an ext3 file system online to an ext4 file system



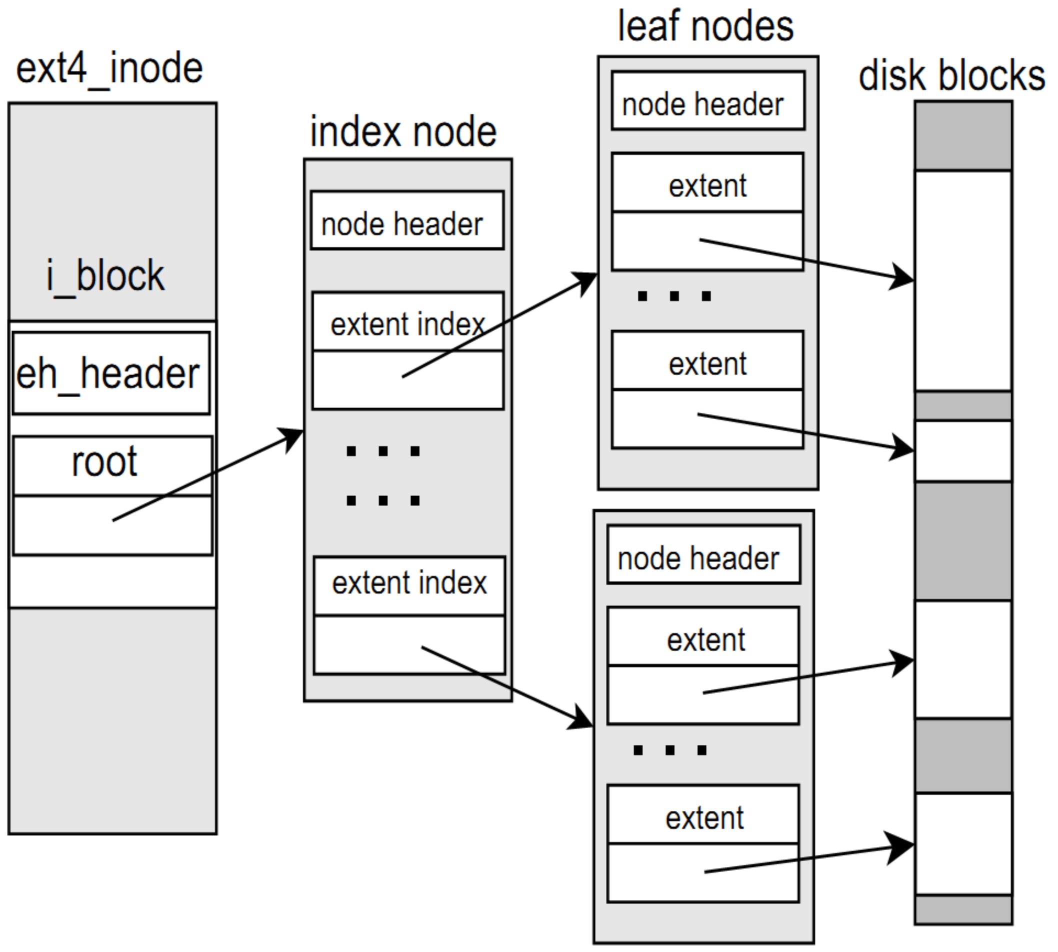
Ext4 file system – Extents 1

- One of the primary disadvantages of ext3 was its method of allocation. Files were allocated using a bit map of free space, which was not very fast nor very scalable.
- Ext3's format is very efficient for small files but horribly inefficient for large files. Ext4 replaces ext3's mechanism with extents to improve allocation and support a more efficient storage structure.
- An extent is simply a way to represent a contiguous sequence of blocks. In doing this, metadata shrinks, because instead of maintaining information about where a block is stored, the extent maintains information about where a long list of contiguous blocks is stored (thus reducing the overall metadata storage).
- Extents in ext4 adopt a layered approach to efficiently represent small files as well as extent trees to efficiently represent large files. For example, a single ext4 inode has sufficient space to reference four extents (where each extent represents a set of contiguous blocks).
- For large files (including those that are fragmented), an inode can reference an index node, each of which can reference a leaf node (referencing multiple extents). This **constant depth extent tree** provides a rich representation scheme for large, potentially sparse files. The nodes also include self-checking mechanisms to further protect against file system corruption.

Read more: http://www.kernel.org/doc/ols/2007/ols2007v2-pages-21-34.pdf

Ext4 file system – Extents 2

- Ext4 supports two block maps. Extent map is more efficient and can handle large file in comparison with the old indirect block map



Repetition end

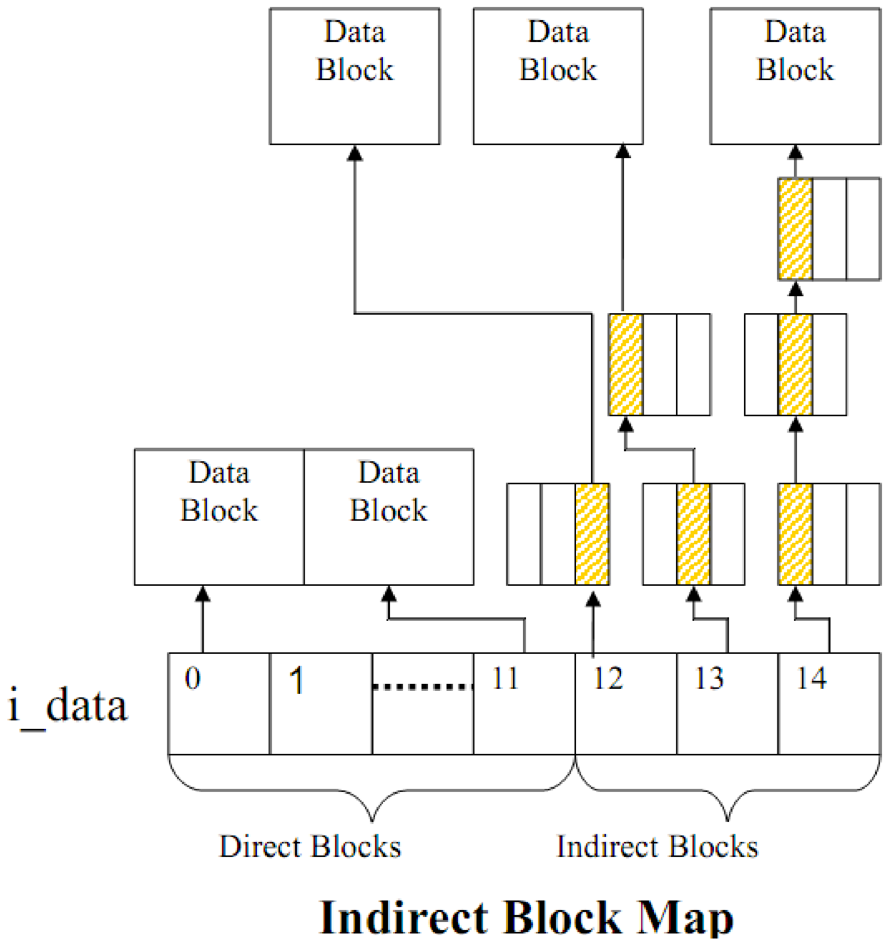
See forensic 1
http://users.du.se/~hjo/cs/dt1059/presentation/CF1_9_filesystems.pdf

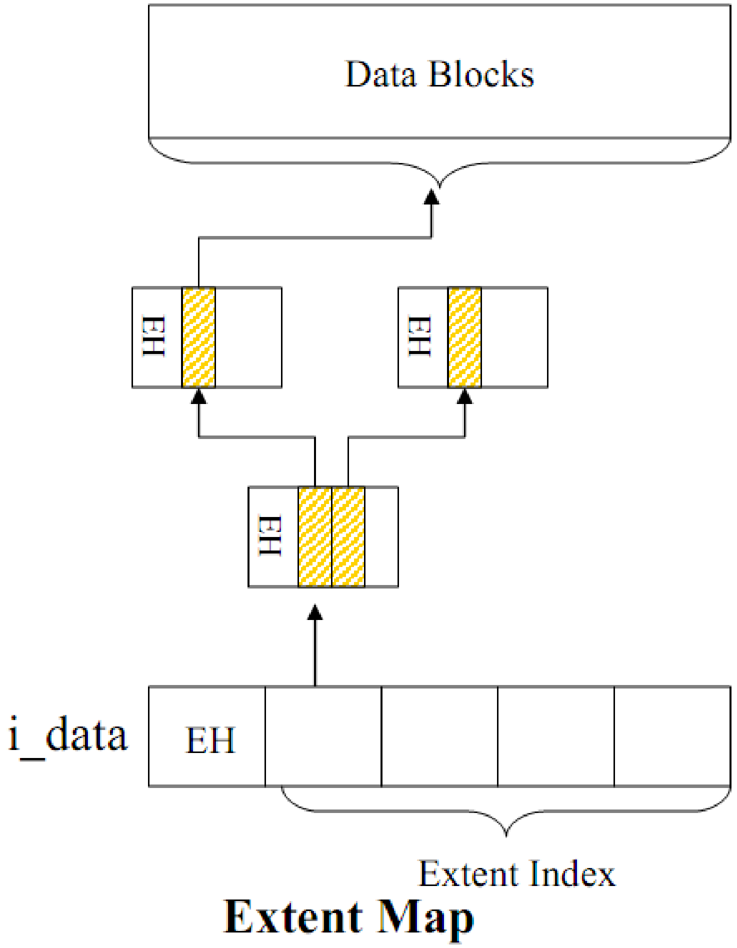
ext4 is important since most Android devices run it ==
Actually most of the computers in the world!

Slides from forensic 1 end

Ext4 file system – Extents 2 (alt.)

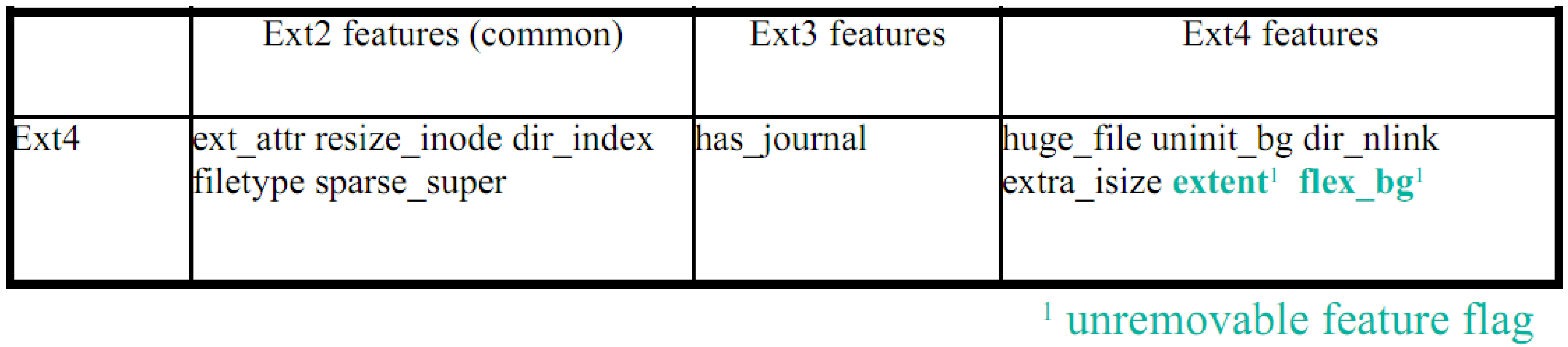
- Ext4 supports two block maps. Extent map is more efficient and can handle large file in comparison with indirect block map





Ext4 file system - timestamps

- Timestamps in the extended file system arena prior to ext4 were seconds based
- This was satisfactory in many settings, but as processors evolve with higher speeds and greater integration (multi-core processors) and Linux finds itself in other application domains such as high-performance computing, the seconds-based timestamp fails in its own simplicity
- Ext4 has essentially future-proofed timestamps by extending them into a nanosecond LSB. The time range has also been extended with two additional bits to increase the lifetime by another 500 years
- Feature flags of ext filesystems



	Ext2 features (common)	Ext3 features	Ext4 features
Ext4	ext_attr resize_inode dir_index filetype sparse_super	has_journal	huge_file uninit_bg dir_nlink extra_isize **extent**[1] **flex_bg**[1]

[1] unremovable feature flag

Ext4 file system - Performance

- Prevent file fragmentation and decrease CPU utilization.
- File-level preallocation
  - Certain applications, such as databases or content streaming, rely on files to be stored in contiguous blocks (to exploit sequential block read optimization of drives as well as to maximize Read command-to-block ratios).
- Delaying block allocation
  - Delayed allocation is used by write system call. It delays real block allocation until written data is flushed from memory to disk
- Multi-block allocation
  - A final optimization—again, contiguous block related—is the block allocator for ext4.
  - In ext3, the block allocator worked by allocating a single block at a time. When multiple blocks were necessary, it was possible to find contiguous data in non-contiguous blocks. Ext4 fixes this with a block allocator that allocates multiple blocks at a time, likely contiguous on disk. Like the previous optimizations, this optimization collects related data on the disk to optimize for sequential Read optimization.
  - The other aspect of multi-block allocation is the amount of processing required for allocating the blocks. Recall that ext3 performed allocation one block at a time. In the simplest units, that required a call to block allocation for each block. Allocating multiple blocks at a time requires many fewer calls to the block allocator, resulting in faster allocation and reduced processing.

Ext4 file system - Reliability

- Checksumming the file system journal
  - Ext4 supports multiple modes of journaling, depending upon the needs of the user. For example, ext4 supports a mode in which only metadata is journaled (Writeback mode), a mode in which metadata is journaled but data is written as the metadata is written from the journal (Ordered mode), and a mode in which both metadata and data are journaled (Journal mode—the most reliable mode). Note that Journal mode, although the best way to ensure a consistent file system, is also the slowest, because all data flows through the journal.
- Online defragmentation
  - Although ext4 incorporates features to reduce fragmentation within the file system (extents for sequential block allocation), some amount of fragmentation is impossible to avoid when a file system exists for a long period of time. For this reason, an online defragmentation tool exists to defragment both the file system and individual files for improved performance. The online defragmenter is a simple tool that copies files into a new ext4 inode that refers to contiguous extents.
  - The other aspect of online defragmentation is the reduced time required for a file system check (fsck). Ext4 marks unused groups of blocks within the inode table to allow the fsck process to skip them entirely to speed the check process. When the operating system decides to validate the file system because of internal corruption (which is inevitable as file systems increase in size and distribution), ext4's overall design means improved overall reliability.
- There is a trade-off between performance and reliability.
  - There is a known-issue on data loss. If a crash occurs when created or truncated file is closed, or a file is renamed to replace the previous file, data may be lost.
  - To avoid the issue, mount ext4 filesystem with “noauto_da_alloc” option.

BTRFS and ZFS

- Advantages of Btrfs or ZFS over Ext4 or other mature file-systems is support for sub-volumes, snapshots, built-in file-system compression, built-in RAID support, and various other modern features as automatic FS repair (scrubbing)
- Btrfs
  - Btrfs is a new copy on write (CoW) filesystem for Linux aimed at implementing advanced features while focusing on fault tolerance, repair and easy administration.
  - Jointly developed at Oracle, Red Hat, Fujitsu, Intel, SUSE, STRATO and many others, Btrfs is licensed under the GPL and open for contribution from anyone.
- ZFS
  - ZFS is an advanced filesystem created by Sun Microsystems (now owned by Oracle) and released for OpenSolaris in November 2005. Features of ZFS include: pooled storage (integrated volume management -- zpool), Copy-on-write, snapshots, data integrity verification and automatic repair (scrubbing), RAID-Z, a maximum 16 Exabyte file size, and a maximum 256 Zettabyte volume size. ZFS is licensed under the Common Development and Distribution License (CDDL).
  - Described as "The last word in filesystems" ZFS is stable, fast, secure, and future-proof. Being licensed under the CDDL, and thus incompatible with GPL, it is not possible for ZFS to be distributed along with the Linux Kernel. This requirement, however, does not prevent a native Linux kernel module from being developed and distributed by a third party, as is the case with zfsonlinux.org (ZOL).
  - ZOL is a project funded by the Lawrence Livermore National Laboratory to develop a native Linux kernel module for its massive storage requirements and super computers.

Read more about BTRFS and ZFS

- Comparisons – they got similar features
  - In short ZFS is more portable, require more RAM and is not integrated into the Linux kernel. Btrfs is Linux only, still under development and may not be production safe yet
    - http://discourse.ubuntu.com/t/zfs-vs-btrfs-experience/1648
  - Facebook is using Btrfs since mid 2104 for trial deployment
    - http://www.phoronix.com/scan.php?page=news_item&px=MTY0NDk
- Article from Linux Format in fronter
  - Linux Format - January 2015.pdf
- Wikipedia
  - http://en.wikipedia.org/wiki/Btrfs
  - http://en.wikipedia.org/wiki/ZFS
- Arch Linux Wiki guides
  - https://wiki.archlinux.org/index.php/ZFS
  - https://wiki.archlinux.org/index.php/Btrfs